Modeling and Intelligent Control for Spatial Processes and Spatially Distributed Systems

by

Efe C. Balta

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2021

Doctoral Committee:

Associate Professor Kira Barton, Co-Chair
Professor Dawn M. Tilbury, Co-Chair
Professor Ilya Kolmanovsky
Associate Professor Chinedum Okwudire

Efe C. Balta

baltaefe@umich.edu

ORCID iD: 0000-0001-8596-8739

Dedicated to my family.

ACKNOWLEDGEMENTS

Conducting research over the past years at the University of Michigan and working on this dissertation has been a humbling experience. I have had the opportunity to work and meet with an exceptional group of people over the years, for which I am forever grateful. The University of Michigan has been a home for me for many years. I feel fortunate to have been a part of this great society. I would like to thank each person who has been part of this journey, beyond grad school and research, including those not part of this brief acknowledgement. Writing this acknowledgement is quite a different and challenging experience compared to dissertation writing, partly due to its sentimental value but mostly because there are so many to thank for. At the end of this chapter in my life, all I can think of is how grateful and lucky I am to have had this opportunity.

To begin with, I would like to share my deepest and most sincere thanks and appreciation to my academic advisers, Prof. Kira Barton and Prof. Dawn Tilbury. They have given me the opportunity to pursue a Ph.D. and have supported me through all the ups and downs of grad school. I have learned so much from them and have been exceptionally lucky to have worked with them over the years. For their guidance, support, and kindness, I will be forever in debt. Thank you for everything.

I would like to thank the committee members Prof. Chinedum Okwudire and Prof. Ilya Kolmanovsky, for their guidance and discussions, which helped to improve the dissertation significantly. Thank you very much for your time and effort in this dissertation.

I thank and acknowledge our funding sources and collaborators who made this work possible. Specifically, I would like to thank the National Science Foundation and the National Institute for Standards and Technology for their funds and awards that enabled this research. Additionally, I would like to acknowledge Applied Dynamics International for our collaboration that helped novel implementations on experimental setups. I would like to also thank the Secure Cloud Manufacturing Multidisciplinary Design Project Team at the University of Michigan for their help with the experimental setups. Furthermore, a special thanks goes to Prof. John Lygeros and Dr. Alisa Rupenyan, who kindly hosted me for a semester at ETH Zurich, Switzerland, to work on novel concepts in industrial control.

I had the great pleasure of coauthoring articles with many of my colleagues. I would

like to thank all my collaborators and coauthors over the years. To list a notable few, many thanks to Ilya Kovalenko, Isaac Spiegel, Yassine Qamsane, Yikai Lin, Miguel Saez, Felipe Lopez, Yağıiz Çiçek, and Doruk Aksoy. I would like to especially acknowledge and thank Dr. James Moyne as a coauthor and a mentor. His guidance over the years has been a tremendous resource. Also, a special thanks to Prof. Atakan Altınkaynak for introducing me to research in undergrad and being a mentor and a coauthor ever since. I want to thank all my colleagues and friends at the Barton Research Group and Tilbury Lab for their great work environment and culture. I will miss you all very much.

I would like to thank my friends in Ann Arbor for making this place home. It is difficult to list everyone, but here is a brief list. Anıl, Ahmet, Yunus, Doğan, Doruk, Anıl, Duygu, Merve, Ilya, Isaac, Deema, Andrea, Suzanne, Shannon, Dom, Vyas, Andrew, Katie, Alex, Nur, Aykut, and Yeliz, thank you all for being there and all the good times over the years. Special thanks to the Turkish crew (including Dom and Vyas at this point) for years of fun, dinners, and barbeques. I would like to also give special thanks to Isaac and Ilya for their friendship and for always being there. I would like to share so many stories and memories about all these great people, but this would take too many pages. I cherish all my time with you all, and I cannot thank you enough for the past years. Our paths will undoubtedly cross again. I also thank my friends from home, Turkey, for their friendship and support over the years. Especially Turgut, Alp, and Halit, who have been supporting me from the very beginning, all the way back since high school and even before. Your friendship means so much to me, and I am very fortunate to have such a close group of friends around me. I would like to thank many others, including Aslı, Özge, Canberk, Yağız, Yiğit, Ozan, Ergin, Emre, Meryem, and my cousin Müjde for their friendship and support over the years. A last big special thanks goes to Yeliz Kaçamak. She has been not only sustainable emotional support but also a reviewer for many of my papers over the years. She has helped me and supported me through so many hurdles through grad school, and we have shared some of my best memories from grad school. Thank you for making me feel at home. I am forever grateful for everything you have done and all the kindness, love, and joy you have brought into my life.

After all these thanks and acknowledgements, it is difficult to describe an emotion even more profound and greater, but I will try my best. I would like to thank my family for their unconditional support, belief, and love, leading me to where I am today. Most importantly, Esma, Bülent, Emre, and Afife. My love and appreciation for having you in my life are immeasurable. I am blessed with having such an amazing family always looking out for me and supporting me through all the chapters in my life. To my brother, Emre, thank you for always inspiring me and pushing me to do more and be confident. To my aunt, Afife,

thank you for bringing all your positivity to my life every day. To my parents, Esma and Bülent, thank you for being my friends, my mentors, and my moral compass for all my life. I owe everything to you. I also want to share my appreciation and love for my grandparents Özcan and Nermin, who have always supported me throughout my life and education. I wish my grandmother Nermin was with us to share this joyous occasion together. I love you all from the bottom of my heart.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Dynamical systems are often characterized by their time-dependent evolution, named temporal dynamics. The space-dependent evolution of dynamical systems, named spatial dynamics, is another important domain of interest for many engineering applications. By studying both the spatial and temporal evolution, novel modeling and control applications may be developed for many industrial processes. One process of special interest is additive manufacturing, where a three-dimensional object is manufactured in a layer-wise fashion via a numerically controlled process. The material is printed over a spatial domain in each layer and subsequent layers are printed on top of each other. The spatial dynamics of the printing process over the layers is named the layer-to-layer spatial dynamics.

Additive manufacturing provides great flexibility in terms of material selection and design geometry for modern manufacturing applications, and has been hailed as a cornerstone technology for smart manufacturing, or Industry 4.0, applications in industry. However, due to the issues in reliability and repeatability, the applicability of additive manufacturing in industry has been limited. Layer-to-layer spatial dynamics represent the dynamics of the printed part. Through the layer-to-layer spatial dynamics, it is possible to represent the physical properties of the part such as dimensional properties of each layer in the form of a heightmap over a spatial domain. Thus, by considering the spatial dynamics, it is possible to develop models and controllers for the physical properties of a printed part. This dissertation develops control-oriented models to characterize the spatial dynamics and layer-to-layer closed-loop controllers to improve the performance of the printed parts in the layer-to-layer spatial domain.

In practice, additive manufacturing resources are often utilized as a fleet to improve the throughput and yield of a manufacturing system. An additive manufacturing fleet poses additional challenges in modeling, analysis, and control at a system-level. An additive manufacturing fleet is an instance of the more general class of spatially distributed systems, where the resources in the system (e.g., additive manufacturing machines, robots) are spatially distributed within the system. The goal is to efficiently model, analyze, and control spatially distributed systems by considering the system-level interactions of the resources. This dissertation develops a centralized system-level modeling and control framework for

additive manufacturing fleets.

Many monitoring and control applications rely on the availability of run-time, up-to-date representations of the physical resources (e.g., the spatial state of a process, connectivity and availability of resources in a fleet). Purpose-driven digital representations of the physical resources, known as digital twins, provide up-to-date digital representations of resources in run-time for analysis and control. This dissertation develops an extensible digital twin framework for cyber-physical manufacturing systems. The proposed digital twin framework is demonstrated through experimental case studies on abnormality detection, cyber-security, and spatial monitoring for additive manufacturing processes. The results and the contributions presented in this dissertation improve the performance and reliability of additive manufacturing processes and fleets for industrial applications, which in turn enables next-generation manufacturing systems with enhanced control and analysis capabilities through intelligent controllers and digital twins.

CHAPTER I

Introduction

Development of models and theories for characterizing and analyzing complex dynamical systems is a fundamental study in engineering. Dynamical systems in engineering evolve over a domain of interest (e.g., time, space, discrete-event, hybrid of multiple domains). These domains are leveraged in developing appropriate models for various purposes including fault detection, performance analysis, control, and coordination with other systems. While in general it is possible to develop a time-based or *temporal* representation of the dynamics of a system or process, considering the dynamical evolution over multiple domains including the temporal domain provides an improved and comprehensive characterization.

One such process that is analyzed in detail in this dissertation is Additive Manufacturing, also known as three-dimensional (3D) printing. While different types of additive manufacturing technologies exist, in general, an additive manufacturing process utilizes a digital representation (e.g., a computer-aided design (CAD) file) of the desired end-product to manufacture the three-dimensional object in a layer-wise fashion. The layer-wise manufacturing process provides great flexibility in terms of the geometrical complexity of the product design, as well as the material selection. Additionally, the process of producing a near-finished part geometry directly from a digital representation enables a high level of customization while reducing the need for customized tooling in manufacturing, hence reducing tooling and setup costs. Due to their flexibility and efficiency, additive manufacturing technologies have been increasingly adopted in modern manufacturing systems in industry. At each layer in an additive manufacturing process, the build material is formed into a desired solid shape by utilizing various methods, such as extrusion, sintering, and material curing [36, 129, 188]. Then, the next layer is formed on top of the current one to manufacture a three-dimensional geometry by a layer-to-layer process.

The evolution of each layer on top of the previous one results in an inherent *spatial dynamical system* that evolves with the addition of each new layer. Additionally, the layer-

to-layer nature of the process and the temporal dynamics of the material interactions within a layer and between layers define the temporal dynamic evolution of the spatial dynamics, resulting in a *spatiotemporal dynamical system*, where spatial and temporal dynamics interact. Due to the complex material interactions and physics in the process, additive manufacturing processes are often difficult to model, analyze, and control efficiently. An important research challenge is to develop effective methods to address the aforementioned difficulties for additive manufacturing processes in a formal, repeatable, and scalable fashion. Developing efficient methods for analysis and control will enable further utilization of additive manufacturing in industry by enabling enhanced process reliability, repeatability, yield, and efficiency. By studying the spatial dynamics of the process in addition to the temporal dynamics, it is possible to develop methods to characterize, analyze, and control not only the mechanical properties of an additive manufacturing process but also the properties of the printed part (i.e., the end product). In this dissertation, control-oriented modeling, control, system-level interactions, and online monitoring of spatial processes are studied and several solutions are proposed. We first define spatial processes and spatially distributed systems.

Spatial processes are a broad class of processes that arise in many engineering, social, and scientific applications. In contrast to temporal processes, where the evolution of process states are represented as a function of time, in spatial processes dynamics, constituent states are represented as a function of the spatial variables. Some exciting applications of spatial processes include geographical mapping and modeling [39, 88], population dynamics and ecological systems modeling [186], biological systems and disease mapping [158, 163], brain function mapping, and 3D imaging [72].

In engineering, spatial processes are often used in the context of mechanical modeling of thermal, fluidic, and mechanical systems. In most of the engineering applications, spatial processes are modeled as infinite-dimensional partial differential equations (PDE) of space and time. Although a variety of discretization and numerical approximation methods exist for many of these problems in the literature, in general, closed-form solutions do not exist. Additionally, PDE formulations of spatial processes are often computationally intractable for online monitoring, estimation, and control applications. For these purposes, spatial processes are often modeled and analyzed over a predefined discretization that reduces the infinite-dimensional modeling problem into a high-dimensional, but often tractable, approximation with multiple (possibly coupled) ordinary differential or difference equations (ODEs). By leveraging the discretized representations, it is possible to develop control-oriented models of spatial processes that can be used for online closed-loop control applications. Dynamic states of spatial processes are termed spatial dynamical states. Although

there exist control applications for spatial processes in the literature, important research challenges about characterizing the performance of spatial dynamical states of the process, in-situ sensing and estimation, online monitoring and verification, and control synthesis for spatial processes represent active research avenues.

Spatially distributed systems are composed of multiple (often distributed) spatial or temporal processes interacting over a spatial domain. Thus, spatially distributed systems provide a system-level architecture, where the individual processes interact with each other over a network, which is distributed over a spatial domain (e.g., multiple manufacturing processes in a manufacturing system, spanned over the spatial domain of the plant floor). The domain spanned by spatially distributed systems are larger than a single spatial process and may include multiple spatial or temporal processes that are coupled with one another. Scaling the modeling efforts to understand the interactions of multiple spatial processes with each other and developing efficient and scalable control applications for spatially distributed systems is an important research challenge. Note that the dynamics of the constituent processes within spatially distributed systems are not necessarily spatial in our discussions. Some relevant examples of spatially distributed systems in engineering are multi-agent robotic systems [173], manufacturing and supply chain systems [11, 185], and energy networks [85, 202]. While the dynamics of individual processes in these systems may be mainly temporal in nature, we are interested in the spatial characterization of process dynamics and interactions between the processes over a spatial domain.

This dissertation focuses on fundamental research challenges in the modeling, in-situ measurement, verification, and closed-loop control for spatial processes both at a single process and system scale. Single process scale in the context of this dissertation is a single additive manufacturing process in which we focus on the spatial dynamics of the system. Thus, within the context of spatial processes, this dissertation focuses on the subset of spatial processes that evolve over a discretized spatial domain, named discrete spatial processes. Additionally, we focus on spatial dynamics that evolve due to control inputs applied in a repetitive fashion, e.g., layer-to-layer material addition in an additive manufacturing process. Therefore, the contributions in this dissertation apply directly to this subset of spatial processes, and further extensions may be required to extend the contributions to a more general class of spatial processes (e.g., non-repetitive). The system scale in the context of this dissertation is a spatially distributed fleet of additive manufacturing systems in which the additive manufacturing systems interact with one another. Thus, within the large class of spatially distributed systems, this dissertation focuses on spatially distributed resources in a manufacturing system and their interactions through production scheduling and knowledge transfer via a centralized control framework to improve system-level throughput and

efficiency. Applications of the system-level modeling, analysis, and control applications to spatially distributed systems in other fields may require extensions in terms of specific dynamical models and control objectives. Here, we refer to additive manufacturing systems as cyber-physical systems that perform an additive manufacturing process, e.g., 3D printers. The class of spatial AM processes that are investigated are examples of discrete spatial processes, and they have a well-defined solution, which is formally defined in later chapters, under a given initial condition and spatial input.

In this dissertation, we are interested in the steady-state response of a spatial process to spatial inputs that are iterative in nature, e.g., iteratively printing layers in an AM process, or repeatedly manufacturing a specific geometry in an AM Fleet. In that case, the dynamics of the output of the process over a spatial discretization are used as a control-oriented model that maps the initial condition and the input to the spatial state at the next time-step or iteration. AM processes have inherent spatial dynamics that may be considered over a discretized spatial domain for each new layer. Additionally, the layer-to-layer evolution of the spatial process poses a unique set of discrete dynamics that are different than the usual time-discretized system dynamics. Note that the spatial processes of interest may not have a closed-form representation available and could be represented by multiple models that interact with one another in the form of meta-models or "live" models representing parts of virtual representations of a physical process.

As many processes exhibit spatio-temporal dynamics, the temporal aspect should be treated carefully when dealing with spatial processes. The connection between spatial and temporal dynamics may be through event-based or spatially specified references that also yield a temporal evolution. In such cases, it is often advantageous to translate the output dynamics of the system into a purely spatial or purely temporal type for analysis purposes. Appropriate translations for online analysis often require representations of the physical process that adapt to the changing conditions and input references. Digital twins are purpose-driven digital representations of a physical or digital process or entity. Note that digital twins are not necessarily models themselves, but they utilize various types of models, such as the previously mentioned ones, to digitally represent the twinned entity by incorporating online data analysis. By incorporating possibly multiple models of the twinned entity of interest, digital twins utilize online measurements to maintain an up-to-date representation, which may be used for monitoring, analysis, or control purposes. Digital twin technology has emerged as a prominent technological enabler for providing a unifying framework for online analysis and control of complex processes such as spatio-temporal AM processes. This dissertation illustrates several applications that utilize digital twins to effectively implement the developed engineering solutions for monitoring, control,

and analysis.

1.1 Contributions

The four primary contributions of this dissertation are summarized in this section. Each primary contribution corresponds to a chapter in the dissertation. The four primary contributions are numbered as *C1-C4* and referenced accordingly throughout the dissertation.

C1 - A unifying control-oriented modeling framework for spatial AM processes: To model and analyze spatial processes for closed-loop control purposes, we need a unifying modeling formalism to represent process dynamics and quantify performance in the spatial domain. While many modeling approaches exist in the literature, there is no unifying modeling framework to capture the layer-to-layer spatial dynamics of an AM process to enable the development of layer-to-layer performance metrics that characterize part functionality. *The first main contribution of this dissertation* is the development of a modeling framework and a notion of layer-to-layer stability to characterize the performance of the layer-to-layer spatial AM processes. This is a core development that enables many other future contributions in the dissertation. The spatial modeling framework and layer-to-layer stability metrics are presented in Chapter III.

C2- Novel layer-to-layer control methods that utilize spatial models of AM processes: Existing approaches in the literature often employ assumptions on many of the system theoretic aspects of AM process control, which may result in restrictions in practical applications. Additionally, AM processes have favorable dynamical properties that are not fully exploited in the current literature for robust and scalable controller development in an algorithmic fashion. *The second contribution* is the introduction of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints. Novel approaches leveraging the positivity and monotonicity of spatial height dynamics are presented for controllers that are scalable to large spatial domains. The layer-to-layer controllers and their applications are presented in Chapter IV.

C3 - A system-level control approach for spatially distributed AM processes: While AM processes are often employed in the form of a fleet in practice, little attention in the literature has been given to the application domain of AM Fleets. Since AM produces customized parts that may differ between consecutive runs, an effective system-level controller should be able to efficiently schedule incoming jobs, maintain fleet efficiency, and ensure high quality at a low operation cost. We present a novel closed-loop system-level scheduling controller for AM Fleets, utilizing the centralized system-level control frame-

work. Furthermore, while extensive engineering efforts are put into improving process performance and parameter tuning, the knowledge gained from one process is often not transferred to other processes in a systematic fashion. A structured approach to characterize and transfer the knowledge gained from one process to improve another one in the fleet has exceptional potential to improve the utilization of advanced control methods in AM Fleets and improve the utilization of AM processes in industry overall. *The third contribution* of this dissertation is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets. A system-level centralized control framework and its applications in scheduling and knowledge transfer are presented in Chapter V.

C4 - An extensible digital twin framework for monitoring and analysis of spatio-temporal processes: AM processes produce high-volume data that is spatio-temporal and at varying sampling rates. Monitoring and analysis of an AM process requires a framework that is extensible and adaptable for various data types, while enabling different analysis approaches ranging from data-driven machine learning methods to rule-based decision makers that leverage subject matter expertise. Digital twin technology is utilized to present a comprehensive framework for monitoring and analysis of AM processes for applications of performance monitoring, anomaly detection, and cybersecurity. An effective analysis framework is able to synthesize the spatial and temporal data and is capable of implementing various analysis methods such as machine learning classifiers and various other online analysis methods. Therefore, *the fourth main contribution* is a digital twin framework that is flexible and extensible to incorporate various models and data structures for run-time analysis of cyber-physical manufacturing systems. We demonstrate the digital twin framework on an extensive cybersecurity application implemented on an AM process and provide further implementations for analyzing and performance monitoring on spatio-temporal process data. The digital twin-based process monitoring and analysis framework is presented in Chapter VI.

1.2 Overview

This dissertation focuses on developing modeling and intelligent control methods for spatial processes and spatially distributed systems with applications to Additive Manufacturing and system-level applications in an industrial setting, named as Additive Manufacturing Fleets. Chapter II provides background information and limitations of the existing literature. Chapter III develops control-oriented models for the layer-to-layer spatial dynamics of spatial additive manufacturing processes and introduces the concept of layer-to-

layer stability to characterize the performance of the manufactured products. Chapter IV builds on the previous chapters to develop layer-to-layer closed-loop control methods for high-performance spatial process control. Chapter V presents a centralized framework for modeling, control, and run-time decision-making for additive manufacturing fleets. Chapter VI provides insights on process monitoring for general cyber-physical manufacturing resources, and presents experimental results on performance monitoring, anomaly detection, and cybersecurity for additive manufacturing processes. Finally, Chapter VII provides concluding remarks, highlights the contributions that apply to related fields, and provides several future research areas. Figure 1.1 illustrates a graphical outline of the focus areas for Chapters III-VI in this dissertation.

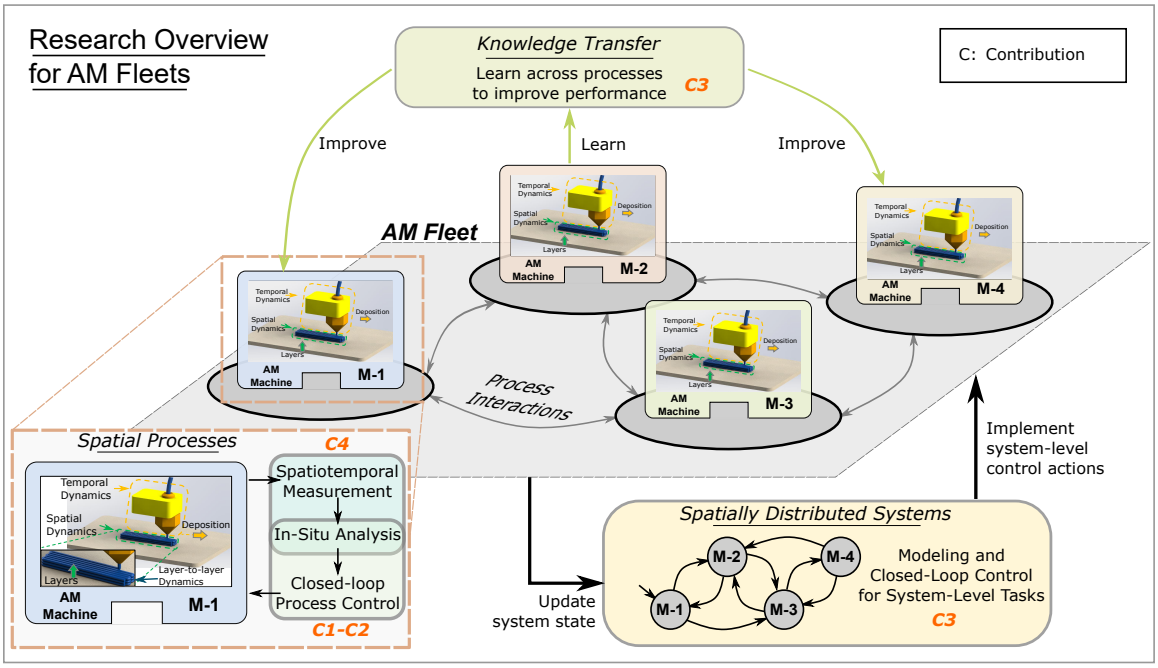


Figure 1.1: A pictorial representation of the research overview for AM processes in an AM Fleets as spatial processes in a spatially distributed system. At the machine level, we want to monitor, analyze, and control process dynamics (spatial and temporal) for improved reliability and quality. At a system level, we model the AM machines as spatially distributed processes that interact to accomplish system-level tasks such as optimizing throughput and yield. Additionally, we propose a knowledge transfer methodology to learn from a process in the AM Fleet and use that knowledge to improve the other processes for better process and system performance. The contributions *C1-C4* are defined in this Chapter.

CHAPTER II

Background

The general class of spatial systems that are investigated in this dissertation is termed *discrete spatial processes*, where the state of the process is considered to evolve over a spatial domain due to inputs that are applied over the spatial domain. As an example, if the spatial state of the process represents the spatial height map of an additive manufacturing process up to a current layer, then the spatial input is the material input at each layer, and the spatial dynamics of the process prescribes the layer-to-layer evolution of the spatial height map as a function of the state and input of each layer. In this chapter, a brief literature review for the class of discrete spatial systems at the single process-level and spatially distributed systems at the system-level is provided for introductory purposes. More detailed literature reviews specific to each chapter are given at the beginning of each chapter.

2.1 Discrete Spatial Process Modeling and Control

Here we discuss introductory concepts and literature gaps for modeling and control of discrete spatial processes. Discrete spatial processes emerge in many areas of science and engineering [19, 29, 103, 120, 147, 164]. While many of the proposed models are used for modeling and behavioral analysis of spatial processes, control applications for spatial processes have been limited to a subset of engineering applications such as bioengineering, geostatistics, and manufacturing applications. Here, we specifically focus on the literature regarding discrete spatial process modeling and control problems in the context of manufacturing processes. Spatiotemporal schemes to analyze the quality of welding processes are proposed in [91,117,156]. Spatial models and controllers for machining processes have been successfully utilized in the literature [61, 84, 113].

Additive Manufacturing (AM) is an important research area for spatial modeling and control that has attracted researchers due to the unique research challenges posed in the

spatial dynamics of the process. AM revolutionized manufacturing systems in the context of smart manufacturing, also called Industry 4.0, due to its high flexibility and high customizability. However, in practical applications, AM processes often have poor reliability and repeatability due to a lack of appropriate closed-loop controllers on the spatial dynamics, which results in an open-loop layer-to-layer process with minimal or no error correction. As a result, the application of AM in industry is hindered as most processes need fine tuning of process parameters for repeatable and reliable operation in an industrial setting. Therefore, developing high-performing AM controllers will increase the utilization and adoption of AM in the industry, which in turn will enable the next generation of efficient, reliable, and customizable manufacturing systems.

Spatial dynamics in the context of AM constitute the spatial characteristics of the process and include the change of material volume and location in space, deposited material interactions with the build plate, and the geometry of the deposited material and the printed part [4,13,65,83,90,94,123,150,170,184,200]. While all spatial controllers in the previous literature capture the spatial dynamics of AM processes, no unified modeling framework is adopted, instead many different models serving similar purposes are developed for various control applications. Additionally, the spatial dynamic behavior of the proposed models in the presence of spatial disturbances is often not explicitly modeled, and a performance measure to characterize the spatial dynamic states of a layer-to-layer AM process is not provided in the current literature. This is an important gap in the analysis of spatial dynamics of AM processes. While variations of well-known Lyapunov stability are provided for many AM spatial control applications [3,4,21,93,170], a similar measure for the layer-to-layer spatial dynamics to quantify the performance of the printed part, and therefore the process itself, has not been proposed. Further discussions on control-oriented modeling and proposed models with performance metrics are given in Chapter III.

Most of the previously mentioned work in the literature aims to control a single layer in the process rather than focusing on the layer-to-layer dynamics of the spatial process. There has been little effort in the literature to control the layer-to-layer process dynamics by considering notions of reachability, stabilizability, and controllability. Many of the existing works adopt assumptions on the process dynamics such as identical initial conditions and uniform deposition patterns which may be limiting for certain practical applications [4, 93, 149, 170]. Additionally, spatial dynamics of AM processes exhibit certain favorable dynamical properties, such as monotonicity of certain dynamical aspects, that may be exploited for efficient and algorithmic controller development that leverages well-developed theory from applied numerical methods literature and monotone operator theory [6, 68, 152, 168]. Scalable control methods that can deal with model uncertainty, layer-

to-layer varying spatial deposition paths, and constraints are needed for high-performance AM process modeling and control. However, leveraging properties of a layer-to-layer AM process to enable controllers that can be scaled to large spatial domains in a computationally efficient manner has not been proposed in the literature. Further discussions on the closed-loop control applications and proposed methods are given in Chapter IV.

2.2 Modeling and Control for Spatially Distributed Systems with Spatial Processes

Moving on to spatially distributed systems, we are interested in analyzing not only an individual process, but a network or processes that are spatially distributed in a domain. Most of the existing literature in the area of spatially distributed systems considers the individual processes that are part of the spatially distributed system as temporal processes [135]. In application areas such as multi-agent systems, the constituent temporal processes are often abstracted as discrete-event systems or timed automata [5, 64, 173]. Models of spatially distributed systems with spatial processes are also utilized in other scientific research fields. There are examples in biological systems where interactions of spatial processes or tissue behavior are modeled as spatially distributed systems [72, 78]. In the context of manufacturing systems, spatially distributed processes are analyzed with their abstracted dynamics that are often temporal and the spatial interactions between the spatial processes are not considered. There is a research gap in the control literature on utilizing spatially distributed spatial processes and their interactions to build system-level closed loop controllers to accomplish system-level tasks while rejecting disturbances in the system. A few example of such systems in engineering are interconnected manufacturing processes, supply chains, and energy networks. As spatial processes are inherently data-rich, exploring the interactions of spatially distributed systems with spatial processes may reveal innovative modeling and control approaches that are complementary to the existing literature in distributed systems modeling and control.

Learning applications for spatially distributed spatial processes have been limited with a few recent exceptions [63, 124, 190]. Learning for spatial processes aims to improve the performance of a process through the information learned on a similar but different spatially distributed spatial process. In semiconductor manufacturing processes, the idea of using process knowledge (subject matter expertise) to improve spatially distributed processes has been utilized in chamber matching [32, 174]. However, these developments are limited to the application domain and a general architecture to learn across spatial processes has not yet been proposed. These cross-domain relationships have been studied in the context of

transfer learning [148]. In the context of this dissertation, we name all such efforts to realize cross-domain learning as *knowledge transfer*. Leveraging knowledge transfer applications to improve the performance of spatially distributed spatial processes is a promising research area to propose a general approach and address this gap. In the context of AM fleets, knowledge transfer is crucial since parameter tuning is a labor-intensive task, which is required for high-performance applications in the industry. However, there is often no systematic methodology to transfer the knowledge gained by tuning a set of parameters to a similar process. This is an important gap that needs to be addressed for implementing scalable and efficient AM fleets. Chapter V further discusses the centralized control and knowledge transfer aspects and provides solutions that build on the previous chapters.

2.3 Discrete Spatial Process Monitoring and Analysis

Process monitoring through sensory information is the first step to characterize the performance of a process. Performance characterization in this context is the study of evaluating if the process conforms to certain specifications in terms of predefined performance metrics, commonly termed as key performance indicators. The purpose of process monitoring and analysis in the context of spatiotemporal processes may include but is not limited to abnormality, anomaly, and fault detection, performance degradation analysis, cybersecurity analysis, and quality control. Model-based and data-driven approaches are utilized for online process monitoring. Model-based approaches utilize phenomenological models of a process to define an expected behavior or a metric that represents a desired behavior. Data-driven approaches use data from previous and current measurements to derive a model that represents the data by means of an appropriate fitting measure (e.g., linear regression, support vector machines, artificial neural nets, etc.). An online monitoring system utilizes a model of the process to evaluate a desired metric for the measured process and reports the results to a higher-level decision-maker.

A powerful framework for online monitoring analysis in the literature utilizes *formal methods*, where online measurements of a process are compared to predefined logical specifications defined in a formal mathematical language to monitor whether the process conforms to the specifications. Applications of formal methods-based online monitoring have been successfully demonstrated in many safety-critical systems for in-situ monitoring and control [59, 76, 119, 144]. Furthermore, utilizing online monitoring methods with process data enables in-situ verification of a process with respect to given specifications, i.e., without the need for post-process analysis [60]. There have been recent developments in formal methods to specify spatio-temporal logical propositions for spatial processes [33, 49].

However, these tools have been used as model checkers in simulation rather than online monitoring tools with real-time systems.

Online schemes to monitor spatio-temporal processes in real time are proposed in [18, 85, 143]. While simulation studies clearly show the feasibility of the proposed approaches, applications in physical processes such as AM are often hindered by important challenges. Data collection and real-time analysis are a challenge in AM processes since the processes often have heterogeneous data streams that are available over multiple sampling rates [131]. The heterogeneous data streams may include process images from multiple cameras, profile measurements of the height of each layer, and additional sensors such as temperature, pressure, and vibration sensors that track the process outputs in real time. Additionally, since the process is spatio-temporal by nature, an effective monitoring solution should be able to synthesize the sensory data by considering the spatio-temporal dynamics of the process, which may be computationally expensive or difficult to model. Digital twin technology has been increasingly utilized in the literature to monitor AM processes and the interconnected dynamical evolution of process parameters [15, 46, 107, 130]. However, the existing frameworks present bespoke solutions that do not generalize to a comprehensive and extensible framework that can deal with the dynamics of the process and the available data streams. Additionally, since AM processes often utilize reference files that are prescribed in the spatial domain, an efficient monitoring framework should be able to transition between spatial and temporal domains to analyze process specifications in an in-situ fashion. Furthermore, due to the cyber-physical nature of AM processes in which a physical product is manufactured from a purely digital representation, cyber-attacks on an AM process may result in damage to physical systems. As a consequence, cybersecurity of AM processes is an important challenge to be addressed for widespread adoption of AM in industry [27, 45].

Currently, there exists no general framework to characterize the spatiotemporal behavior of an AM process with respect to a performance metric to develop efficient analysis tools for the purposes of performance monitoring, anomaly detection, and cybersecurity. By developing appropriate real-time measurement and estimation techniques, it is possible to leverage extensions of the existing literature to propose a generalized framework to characterize and verify spatial processes in terms of performance metrics to perform anomaly detection, cyber-attack detection, and performance analysis. Chapter VI addresses these research challenges and presents an extensible solution using digital twin technology.

CHAPTER III

Control-Oriented Modeling for Spatial Processes

Building on the insights and research directions identified in the introduction and background, in this chapter a control-oriented modeling framework for spatial processes is presented. The main contributions for this chapter are the development of a modeling framework and a notion of layer-to-layer stability to characterize the performance of the layer-to-layer spatial AM processes (*C1*). The linear model framework is named as a linear layer-wise spatially varying system that represents the layer-to-layer spatial dynamics of a generic AM process. The focus of the chapter is outlining the applications of closed-loop control for improving spatial process performance in Additive Manufacturing (AM) processes. Additionally, a concept of layer-to-layer stability is presented for spatial dynamics as a metric to characterize the performance of the spatial dynamics and in turn the printed part. Theoretical developments are illustrated with experimental applications that illustrate the concepts and their applications in practice. Contents of this chapter are included in in [13, 16].

3.1 Motivation and Problem Statement

An important open research area in AM is closed-loop process control [20, 155]. AM process dynamics can be analyzed in two domains. Temporal dynamics constitute the transient response of the deposition process and include material pre-process (e.g. heating), volumetric flow of material through a deposition nozzle [43, 93, 178], and the motion of the deposition system [21, 67, 69]. Spatial dynamics constitute the spatial characteristics of the process and include the change of material volume and location as a function of space, deposited material interactions with the build plate, and the geometry of the deposited material and the printed part [4, 13, 83, 94, 150, 170]. Although most of the temporal dynamics can be modeled using existing tools in robotics, physics, and kinematics, the spatial dynamics of AM processes pose research challenges that require novel modeling and control tools.

In current practice, most AM processes lack closed-loop spatial dynamical control, which results in mid-process failures and reliability issues that restrict the widespread use of AM processes. The lack of closed-loop spatial dynamical control is partly due to a lack of appropriate real-time topography feedback, and also due to a lack of control-oriented models that are suitable for control applications. By developing spatial dynamical modeling methods and corresponding analysis tools, it will be possible to develop efficient closed-loop controllers for high-performance AM processes to ensure reliability and quality.

Spatial dynamics are crucial to ensure that an AM printed part conforms to the design specifications. Layer-to-layer spatial dynamics entail the interaction of deposited materials at adjacent layers. Material characteristics and spatial evolution involve complex physical phenomena that have often been analyzed via numerical simulations [51, 58, 107, 199]. However, these simulation tools are generally not suitable for closed-loop control applications. A control-oriented layer-to-layer spatial dynamical model may represent the height evolution of the AM process over a spatial discretization. By utilizing a discretized representation, it is possible to develop state-space models of the layer-to-layer spatial dynamics. Therefore, a model for control of the AM spatial dynamics should describe the layer-to-layer spatial dynamics and capture the material interactions between layers. Additionally, stability properties of the spatial dynamics and a notion of layer-to-layer stability can be utilized to describe and quantify the performance of AM spatial dynamics (and subsequently an AM printed part) over the layer domain.

Therefore, to enable closed-loop, layer-to-layer control of AM spatial dynamics, appropriate control-oriented models should be developed [13]. Defining the appropriate models and layer-to-layer stability measures will provide a detailed analytical framework for spatial process dynamics in high-performance AM applications and enable closed-loop control. Within this context, two problems are of interest in this chapter: (III-P1) how to develop a mathematical framework to define control-oriented models for layer-to-layer AM spatial dynamics, and (III-P2) how to provide layer-to-layer stability measures to analyze the performance of spatial dynamics under known spatial disturbances.

The mathematical framework in this chapter involves spatial dynamical systems that describe AM processes, given as

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k), \tag{3.1}$$

where $k \in \mathbb{Z}$ is the layer index, $\boldsymbol{x}_k \in \mathbb{R}^n$ is the (spatial) state of the system, and $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control input (related to the material input). Due to the explicit dependence on the layer k, the dynamics of the system may vary between layers. As (3.1) describes a spatially

additive process, $\{\boldsymbol{x}_k\}_{k=k_0}^{k_n}$ is strictly increasing and can be lower-bounded by a positive-semi definite function. In the remainder of the chapter, we address (III-P1) and (III-P2): a linear layer-wise spatially varying (LLSV) model for the general non-linear system in (3.1) is developed to address (III-P1) and layer-to-layer stability properties of the layer-to-layer spatial trajectories $\{\boldsymbol{x}_k\}_{k=k_0}^{k_n}$ are provided to address (III-P2).

Computational models have been developed for the layer-to-layer spatial dynamics of AM processes [51, 58, 71, 199]. While most of these models have high accuracy, the tools used for evaluating such models are computationally expensive. Additionally, computational models are often very complex and do not allow for closed-form representations to build closed-loop control applications.

Control-oriented models have also been proposed to model the spatial dynamics of AM processes. In [83], liquid drop deposition and spreading dynamics for an ink-jet AM process are presented. In [94] a spatial modeling framework for electrohydrodynamic-jet printing (e-jet), a micro-AM process, is introduced and an efficient spatial iterative learning control algorithm is introduced. Drop spreading dynamics for the e-jet spatial deposition process are presented in [150] and various heightmap models for layer-to-layer dynamics at varying fidelities are presented in [149]. A task-basis controller model to ensure uniform deposition width in a micro-robotic deposition is given in [93]. Control models for the deposition height of metal AM processes are given in [65, 90, 170, 184, 200]. While controllers for AM spatial dynamics in the previous literature utilize difference or differential dynamical models of the spatial dynamics over a discretization, no unified modeling framework has been adopted. Many different models serving similar purposes are developed for various control applications. Additionally, the spatial dynamic behavior of the proposed models in the presence of spatial disturbances is often not explicitly modeled. Many of the models lack the capability to express the effect of deposition path directionality on the layer-to-layer dynamics, which is essential for extrusion-based processes. For model and process uncertainties, [4] presents an interval model to account for uncertainties that arise in most practical AM applications. Nevertheless, a performance measure to characterize the spatial dynamics is not provided in the current literature. This is an important gap for the analysis of the spatial dynamics of AM processes. While variations of the well-known Lyapunov stability are provided for many AM spatial control applications [3,4,21,93,170], a similar measure for the layer-to-layer spatial dynamics to quantify the performance of a printed part has not yet been proposed.

In this chapter we present an LLSV model that builds on existing models such as [94] to provide a framework that is able to represent existing spatial models and is extensible to provide control-oriented models with additional capabilities such as uncertainty models,

path directionality, various cross-sectional geometries, and spatial performance metrics for closed-loop controller designs. A spatial modeling framework specific to fused deposition modeling (FDM), where directionality of deposition path changes the spatial dynamics, is presented, and initial results on the layer-to-layer stability of FDM spatial dynamics are given here. We formalize a novel class of LLSV systems and provide the formal definitions and comprehensive analysis tools for layer-to-layer stability of LLSV systems under known spatial disturbances. Similarly, layer-to-layer stability is a novel analysis tool to characterize the spatial dynamics of layer-to-layer processes and quantify the performance of the printed part with respect to desired physical attributes such as optical, mechanical, or electrical properties.

The main contribution for this chapter is the development of a modeling framework and a notion of layer-to-layer stability to characterize the performance of the layer-to-layer spatial AM processes (*C1*). Within this main contribution, specific contributions for the chapter are itemized as the following [16].

(*C1*-1) A novel linear spatial dynamic modeling framework for AM processes and a switched affine system representation.

(*C1*-2) Formal definitions of layer-wise regularity and layer-to-layer stability measures in the context of well-known Lyapunov stability.

(*C1*-3) A formal analysis of robustness margins for layer-to-layer stability measures under spatial disturbances to characterize probabilistic layer-to-layer stability results.

(*C1*-4) Comparison of theoretical versus experimental robustness margins with an experimental study for FDM.

The rest of the chapter is organized as follows. Section 3.2 provides the preliminary definitions, notations, and assumptions. Section 3.3 presents the LLSV model by introducing its constituents based on their contributions to the layer-to-layer spatial dynamics. Section 3.4 provides the definitions of layer-to-layer stability and a theoretical framework for robustness margins in the presence of known spatial Gaussian noise in the system. Section 3.5 presents a case study on FDM, and a comparison of the theoretical results to experimental data. Section 3.6 gives concluding remarks for the chapter.

3.2 Preliminaries

3.2.1 Notations Used in the Chapter

$\mathcal{F}_E$ denotes a fixed inertial frame on the substrate (build plate), defined in $\mathbb{R}^3$, with the orthogonal directions of unit length $(\hat{\boldsymbol{\imath}}_E, \hat{\boldsymbol{\jmath}}_E, \hat{\boldsymbol{k}}_E)$. 3D physical vectors are denoted with boldface letters and a vector arrow, e.g. $\vec{\mathbf{r}}$. Vectors are denoted with boldface letters, e.g. $\boldsymbol{x} \in \mathbb{R}^n$ denotes a vector in n dimensional space. The norms $||\cdot||_1, ||\cdot||_2, ||\cdot||_\infty$ are the ℓ_1, ℓ_2, and ℓ_∞ norms and the induced matrix norms, respectively. A sequence of vectors is denoted by $\{\boldsymbol{x}_i\}_{i=n_0}^{n_f}$. Letter h is reserved for functions of height, the magnitude of the equivalent physical vector in the $\hat{\boldsymbol{k}}_E$ direction. Lowercase letter k denotes the layer index throughout the chapter. Similarly variables indexed with k denote layer-dependent variables. Vector $\boldsymbol{e}_i^n$ denotes the unit vector for the i^{th} dimension of an n-dimensional space (e.g., $\boldsymbol{e}_1^2 = [1, 0]^T$). Sets are denoted with capital letters e.g., A, with cardinality $|A|$.

Matrices are denoted with capital boldface letters, e.g. $\boldsymbol{A} \in \mathbb{R}^{n\times m}$. The element at the i^{th} row and j^{th} column of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}[i, j]$. The spectral radius of $\boldsymbol{A}$ is denoted by $\rho(\boldsymbol{A})$. Vectorization operation is denoted with $vec(\cdot)$ and its inverse, matricization, is denoted with $vec^{-1}(\cdot, n, m)$. A function $\varphi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$ function ([104], Section 4.4) if it is strictly increasing, continuous, and $\varphi(0) = 0$.

3.2.2 Definitions

Definition 3.1. (Power-series bounded matrix) A matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is power-series bounded if $\limsup_{r\to\infty} ||\boldsymbol{M}^r||_\infty = \bar{m} < \infty$ exists, or equivalently if $\rho(\boldsymbol{M}) \leq 1$ and the eigenvalues on the unit circle are of index 1 [108].

Proposition 3.2. *(Corollary to Gelfand's formula) For a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, the following property holds for $r \geq 1$, $r \in \mathbb{Z}_{>0}$.*

$$\rho(\boldsymbol{M}) \leq ||\boldsymbol{M}^r||^{1/r}.$$

Proof. Follows from the proof of Corollary 5.6.14 in [95], which includes the proposition statement given here. □

The following definitions are used to represent the physical AM process by formal mathematical notation. An AM process is defined in a finite volume in $\mathbb{R}^3$, called *volume of interest* (VOI). The VOI is defined by a rectangular cuboid

$$\mathcal{V} = \{\xi \in \mathbb{R}^3 | \xi \in [0, i_E^{max}] \times [0, j_E^{max}] \times [0, k_E^{max}]\},$$

where $(\cdot)_E^{max}$ denotes the upper limit in each direction in $\mathcal{F}_E$. A *layer* is the material deposition on a two-dimensional cross-section of the VOI, with the normal of the cross-section aligned with $\hat{\boldsymbol{k}}_E$ direction.

The material deposition follows a predefined path $p(k,\gamma) : \mathbb{Z}_{>0} \times [\gamma_0, \gamma_f] \to L_k$, where L_k is the 2D deposition plane in $\mathcal{V}$ for layer k, $\gamma \in \mathbb{R}$ is a path parameter, $p(\cdot,\gamma)$ is a Lipschitz continuous mapping, $p(k,\gamma_0)$ is the initial point and $p(k,\gamma_f)$ is the final point of the path and $\gamma_0 < \gamma_f$.

Remark 3.3. *In practice, the temporal execution of the deposition path of an AM process may include discontinuous jumps between deposition points in the geometry. Here we define the predefined path $p(k,\gamma)$ to represent the spatial deposition geometry for a single layer, which may be viewed as the overall spatial representation of the deposition path. Thus, as long as the deposition path is connected, we are able to define $p(k,\gamma)$. We do not treat the cases with spatially disconnected deposition paths in this chapter for simplicity of presentation.*

The *AM process* is defined as the sequential material deposition in a VOI, starting with an initial layer $k = 1$ and continuing in predefined increments in the $\hat{\boldsymbol{k}}_E$ direction. The predefined increment between layers is called the *layer height*, denoted by h_ℓ. In this chapter, paths of all layers k are predefined and fixed for the AM process to simplify further discussions and formulations. Furthermore, h_ℓ is uniform and fixed for all the layers in the AM process.

The dynamics of the AM process can be analyzed in two domains, as described in [13]. Two important attributes of the spatial dynamics of the AM process are the *in-layer* and the *layer-to-layer* attributes. In-layer attributes are related to the deposition of material within a single layer. Layer-to-layer attributes (i.e. layer-to-layer spatial dynamics) relate the material deposition on one layer to a subsequent layer. In other words, layer-to-layer spatial dynamics describe the height evolution of the printed part across multiple layers.

3.2.3 Assumptions

In the presented spatial dynamical model, the dynamics of material flow are assumed to be well-known for the duration of the process. A list of standing assumptions for this chapter is as follows.

(A1) Temporal dynamics of the AM process are stable and in steady-state.

(A2) Material deposition within layer k follows a predefined spatial deposition path $p(k,\gamma)$ accurately (i.e., within some precision that ensures the material is deposited at the desired spatial location) in the spatial domain, for all layers ($\forall k$).

(A3) Spatial dynamics (and consequently the spatial dynamical state) of the AM process are measurable. The spatial dynamics are measurable at the end of the deposition for a layer, including any layer-wise post-process (e.g., material curing, mechanical shape modification, etc.).

(A4) Spatial dynamics are observed as a result of material input in the AM process, and each layer k in the process has a predefined uniform layer height h_ℓ.

Assumption (A1) ensures that the temporal dynamics are stable for the analysis of spatial dynamics in this chapter. Assumption (A2) states that the deposition system follows a predefined path correctly and the disturbances in the process do not alter the actual deposition path in the process. Assumptions (A3)-(A4) ensure that the spatial dynamics are a result of material deposition in the process. Note that we only require the spatial state to be measurable at the end of the material deposition process within a layer including any additional layer-wise process treatments that may be necessary (see e.g., [2, 4, 90, 170]). While layer-wise post-processes such as material curing may influence the deposited spatial dynamics, we consider these influences as part of the spatial dynamics and do not consider the individual effects of material deposition versus process treatment in this dissertation.

3.3 Formulation of the Proposed Model

This section provides the first contribution of the chapter as a formulation of the proposed linear layer-wise spatially varying model and the model representation as a switched affine system. The spatial dynamics of AM processes are represented on a discretization of a domain of interest. First, a discretization scheme for a given VOI is presented and a matrix representation denoting the height evolution over the discretization is given. Then, discretization of a continuously defined deposition path and the local path frame are introduced. Using the discretization scheme, the linear layer-wise spatially varying system model is introduced and a simplified reformulation of the spatial dynamics is given. An uncertainty model in the form of spatial noise is provided at the end of the section.

3.3.1 Discretization of the Volume of Interest

To define the spatial dynamic state of the AM process, it is desirable to define a suitable discretization of the VOI $\mathcal{V}$. We consider the layer-wise deposition as always aligned with $\hat{\boldsymbol{k}}_E$, and define the deposition plane $L_k = \{\xi \in \mathbb{R}^2 \mid [\xi^T, h_k]^T \in \mathcal{V}\}$, where h_k denotes the deposition height for layer k (e.g., $h_k = kh_\ell$). Let α_i, α_j denote the discretization size in the $\hat{\boldsymbol{\imath}}_E, \hat{\boldsymbol{\jmath}}_E$ directions respectively, shown in Fig. 3.1. Discretization in the $\hat{\boldsymbol{\imath}}_E$ direction is

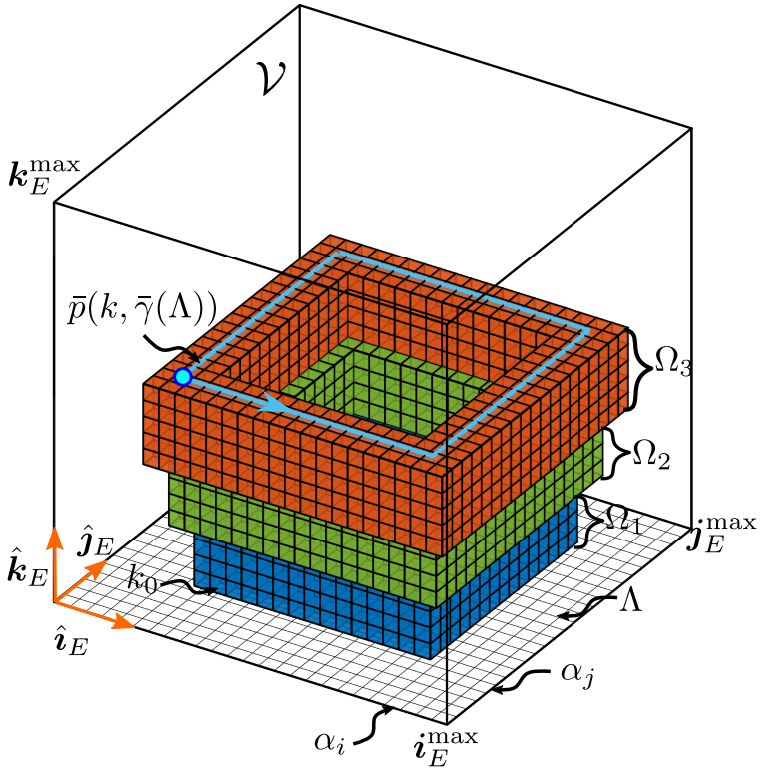


Figure 3.1: The conceptual setting to describe the LLSV system as an AM process. Within the VOI $\mathcal{V}$, the deposition of layers starting from the initial layer k_0 over the spatial discretization λ parametrized by α_i, α_j is shown. Three layer groups Ω_i are shown with five layers in each layer group and the deposition path $\bar{p}(k, \bar{\gamma}(\Lambda))$ for the final layer along with the deposition direction is highlighted in light blue with the start/end point of the path shown with the blue filled circle. The deposition path is aligned with the grid, which results in material deposition centered on the grid points.

defined by the ordered set $\Lambda_i = \{\xi \in \mathbb{R} | \xi = \alpha_i d, x \in [0, i_E^{max}], d \in \mathbb{Z}_{\geq 0}\}$ and similarly defined for the $\hat{\boldsymbol{j}}_E$ direction $\Lambda_j = \{\xi \in \mathbb{R} | \xi = \alpha_j d, x \in [0, j_E^{max}], d \in \mathbb{Z}_{\geq 0}\}$. Then, the discretization of the layer plane D is given as $\Lambda = \{\xi \in L | \xi \in (\Lambda_i \times \Lambda_j)\}$. Λ is assumed to be identical for all layers.

To denote the height at the spatial locations, the matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{n_i \times n_j}$ represents the locations of spatial grid Λ, where each element $\boldsymbol{\Lambda}[m, n]$ corresponds to a spatial location $\boldsymbol{\lambda}(m, n) = (\Lambda_i[m], \Lambda_j[n])$. A realization of the matrix $\boldsymbol{\Lambda}$ at a given layer k denotes the height at the discretized locations in that layer. Overloading the notation, let $h(\boldsymbol{\Lambda}[i, j], k)$ denote the height of the spatial location $\boldsymbol{\Lambda}[i, j]$ up to layer k. Thus the spatial height matrix

$\boldsymbol{H}(\boldsymbol{\Lambda}, k) \in \mathbb{R}^{n_i \times n_j}$ with the discretization Λ for layer k is denoted with

$$\boldsymbol{H}(\boldsymbol{\Lambda}, k) = \begin{bmatrix} h(\boldsymbol{\Lambda}[0,0],k) & \dots & h(\boldsymbol{\Lambda}[0,n_j-1],k) \\ \vdots & \ddots & \vdots \\ h(\boldsymbol{\Lambda}[n_i-1,0],k) & \dots & h(\boldsymbol{\Lambda}[n_i-1,n_j-1],k) \end{bmatrix}. \tag{3.2}$$

Now we can define the spatial dynamic state for layer k as

$$\boldsymbol{x}_k = vec(\boldsymbol{H}(\boldsymbol{\Lambda}, k)). \tag{3.3}$$

Using the underlying discretization Λ, it is possible to represent the spatial dynamic state as a vector $\boldsymbol{x}_k \in \mathbb{R}^{n_g}$, where $n_g = n_i n_j$, or as a matrix by using the $vec^{-1}(\boldsymbol{x}_k, n_i, n_j)$ operation. Therefore, $\boldsymbol{x}_k$ represents the total height evolution (in the $\hat{\boldsymbol{k}}_E$ direction) in the process up to layer k. The number $n_g \in \mathbb{Z}$ denotes the size of the discretization Λ and will be used in place of $n_i n_j$ throughout the rest of the chapter. The discretization Λ represents a finite number of spatial locations in D. Each spatial location can be viewed as a node of a graph, with n_g nodes in total. We define the graph $G = (\Lambda, \mathcal{E})$, with the edges $\mathcal{E}$ in the graph connecting each spatial location $\lambda \in \Lambda$ to its neighboring spatial locations (including diagonals) within Λ. A conceptual representation of G is given in Fig. 3.1 as the bottom grid indicated as Λ. Each intersection in the grid represents a location $\lambda \in \Lambda$ and the grid shows the edges $\mathcal{E}$ between the locations with the diagonal edges between locations omitted in the figure for visual simplicity. The graph G is a simple graph with self-loops, thus its adjacency matrix is positive semi-definite with possible nonzero diagonal entries.

3.3.2 Discretization of the Deposition Path on a Single Layer

Given the spatial discretization Λ, discretization schemes for the deposition path and the local path frames are given here. The deposition path $p(k, \gamma)$ is continuous with respect to the parameter γ. Define $\gamma(\tau) \in [\gamma_0, \gamma_f]$ as a continuously increasing parameter along the path parametrized by the variable τ, so that $\gamma(0) = \gamma_0$ and $\gamma(f) = \gamma_f$. An instantaneous direction vector is then defined as $\vec{\boldsymbol{v}}_\tau = p(k, \gamma(\tau + \epsilon_p)) - p(k, \gamma(\tau))$, on the interval $\tau \in [0, f - \epsilon_p)$, for small $\epsilon_p > 0$. Similarly the instantaneous normalized direction vector is given by $\hat{\boldsymbol{v}}_\tau$. While the choice of ϵ_p affects the direction that $\hat{\boldsymbol{v}}_\tau$ is pointing towards, the analysis of this effect will not be presented here with the understanding that suitable ϵ_p can always be selected, so that $\hat{\boldsymbol{v}}_\tau$ is a "tangent-like" vector. Finally, the local path frame $\mathcal{F}_P$ is defined such that $\hat{\boldsymbol{k}}_P$ is aligned with $\hat{\boldsymbol{k}}_E$, $\hat{\boldsymbol{\imath}}_P$ is aligned with $\vec{\boldsymbol{v}}_\tau$, and $\hat{\boldsymbol{\jmath}}_p = \hat{\boldsymbol{k}}_P \times \hat{\boldsymbol{\imath}}_P$ as expected, where $\times$ denotes the vector cross product.

In practice, the deposition path is defined by a sequence of spatial locations on Λ. To do so, the discretization sizes α_i, α_j must be chosen small enough to minimize the distortion on the continuous path $p(k, \gamma)$. To represent the deposition at the discretization Λ, define the discretized sequence of points on the path as $\bar{p}(k, \bar{\gamma}(\Lambda)) \triangleq \{\lambda_m\}_{m=1}^{n_p}$, where each λ_m is called a deposition location and denotes a spatial location $\boldsymbol{\lambda}(i, j) \in \Lambda$, the parameter $\bar{\gamma}(\Lambda)$ represents the discrete values of γ along the deposition path that aligns with Λ, and n_p is the number of points in the discretized deposition path. Choosing the parameter ϵ_p such that both $p(k, \gamma(\tau + \epsilon_p))$ and $p(k, \gamma(\tau))$ are aligned with Λ ensures that the local frame $\mathcal{F}_P$ is always well-defined with respect to the discretization. For example, if $\alpha_i = \alpha_j = \hat{\alpha} \in \mathbb{R}$, choosing $\epsilon_p = \hat{\alpha}$ ensures that a deposition path without diagonal movements $\bar{p}(k, \bar{\gamma}(\Lambda))$ is aligned with Λ as shown in Fig. 3.1 in light blue. We drop the dependency on the discretization and the layer index for the path whenever it is clear from the context for brevity.

Remark 3.4. *Note that while deposition paths that are not aligned with $\boldsymbol{\Lambda}$ are possible by defining the proper path variables, the spatial representation may become complicated depending on the cross-sectional geometry, the discretization size α_i, α_j, and the AM process itself. For the simplicity of discussions in this chapter, we focus on the cases where the deposition path aligns with the spatial grid with a sufficiently fine grid size.*

3.3.3 Linear Layer-wise Spatially Varying Systems

To model the layer-to-layer evolution of the spatial dynamic state at the spatial discretization $\boldsymbol{\Lambda}$, an LLSV model is presented in this section. LLSV is essentially a discrete linear parameter varying (LPV) model where the state of the system ($\boldsymbol{x}_k$) is the total height up to layer k (e.g. $\boldsymbol{H}(\boldsymbol{\Lambda}, k)$) and the parameter evolution is in terms of layer progression (e.g. k to $k+1$). The LLSV model is constructed as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k, \tag{3.4}$$

where $\boldsymbol{A}_k$ is the spatial register matrix, $\boldsymbol{B}_k$ is the input matrix and $\boldsymbol{u}_k$ is the spatial input vector. In this section, first, the spatial effect of the deposition input ($\boldsymbol{B}_k$) is modeled, then, the effect of the previous layer ($\boldsymbol{A}_k$) is given.

3.3.3.1 Effect of Material Deposition Input

Material deposition along the path $p(k, \gamma)$ results in height evolution on each spatial location with material input. A shape function to describe the local cross-sectional height

evolution as a function of distance from deposition location is defined by $c(p, \boldsymbol{\theta}, r) : L_k \times \mathbb{R}^{n_\theta} \times S \to \mathbb{R}$, where p is the path mapping, $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ is a parameter vector for a given geometry, and $r \in S \subset \mathbb{R}$ is the Euclidean distance from the deposition point in the $\hat{\boldsymbol{\jmath}}_P$ direction (in the local path frame $\mathcal{F}_P$). Note that since $c(p, \boldsymbol{\theta}, r)$ defines the height change in a finite interval, it has finite support for the compact domain S on which the cross-sectional geometry is defined, and it is zero elsewhere.

Remark 3.5. *Definition of the shape-function can be extended to two-dimensional height evolution for systems that have height evolution with radial symmetry, by defining the ball $S \triangleq \beta(p(k, \gamma), r) \subset \mathbb{R}^2$ as the domain.*

Some examples of $\boldsymbol{\theta}$ from literature shown with their discretization in Fig. 3.2 are:

- for a Gaussian bell-curve shape, $\boldsymbol{\theta}$ encodes the mean and covariance [83, 94, 150],
- for an ellipsoidal shape, $\boldsymbol{\theta}$ encodes the minor and major radii [1, 13, 51],
- for a rectangular shape, $\boldsymbol{\theta}$ encodes the height and width of the rectangle [43, 209].

While $c(p, \boldsymbol{\theta}, r)$ defines a continuous shape function at the cross-section of the deposition path, the analysis over the discretization Λ requires discretization of the shape function. Also, note that the shape parameters may depend on the material properties of the AM process and process-specific physical conditions.

Define $\tilde{\boldsymbol{c}}(\lambda_m) \in \mathbb{R}^{n_i \times n_j}$ as the discretized matrix representation of the shape function at the deposition location $\lambda_m \in \bar{p}$, for the path sequence $\bar{p}$ defined previously. $\tilde{\boldsymbol{c}}(\lambda_m)$ is evaluated by sampling the function $c(p, \boldsymbol{\theta}, r)$ centered at the deposition location λ_m over the discretization Λ. $\tilde{\boldsymbol{c}}(\lambda_m)$ can be viewed as the discretized spatial deposition impulse response of an AM process (i.e., cross-section of material spread in Λ due to an impulse deposition at deposition location λ_m). Note that by defining appropriate shape functions for diagonal depositions in the discretization $\boldsymbol{\Lambda}$ it is possible to extend the presented models to various deposition paths. We do not treat such cases in this chapter and identify them as part of future work.

Remark 3.6. *The discrete representation of the shape function is denoted with a matrix of the size of Λ ($\tilde{\boldsymbol{c}}(\lambda_m)$ centered at the deposition location λ_m and the rest zero padded for full dimension) for uniform notation.*

Using the discretized shape representation $\tilde{\boldsymbol{c}}(\lambda_m)$, define $\boldsymbol{c}_m = vec(\tilde{\boldsymbol{c}}(\lambda_m))$ as the *input-to-shape response* for a unit material input at the m^{th} deposition location in $\bar{p}(k, \bar{\gamma}(\Lambda))$. The vector $\boldsymbol{c}_m \in \mathbb{R}^{n_g}$ denotes the height distribution as a result of unit material input to

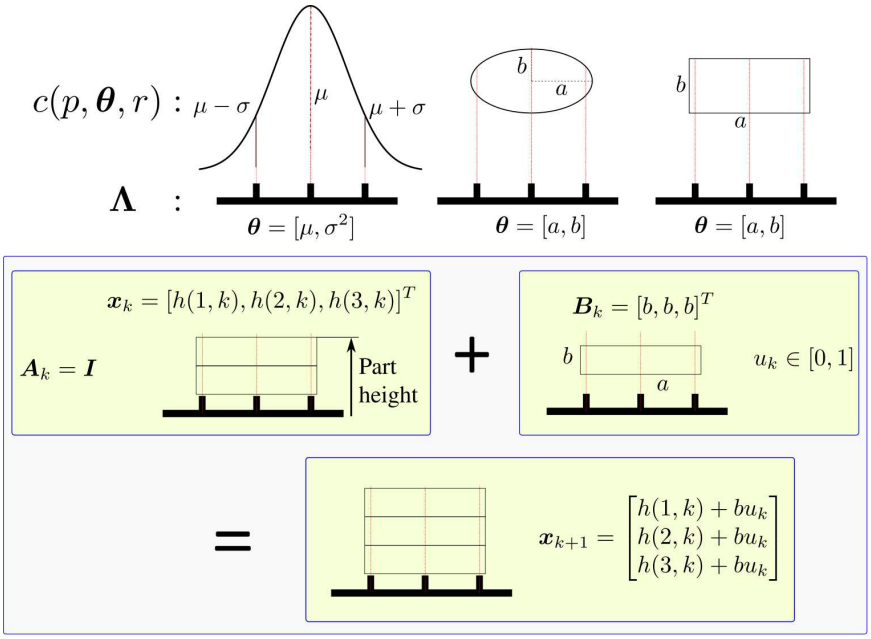


Figure 3.2: Three different shape functions, their corresponding shape parameters and their discretization on Λ (with discretization size α). These shape functions are used for characterizing the cross-sectional shape of deposited materials at each layer. A simple example of the layer-to-layer dynamics on a rectangular cross sectional shape is shown on the bottom.

the system. The spatial height map as a result of the unit input is denoted by the matrix $vec^{-1}(\boldsymbol{c}_m, n_i, n_j)$. Examples of the spatial height map representation are given in [4,13,94, 149,170,194]. For the linear layer-wise representation, define $\boldsymbol{B}_k \in \mathbb{R}^{n_g \times n_u}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$, where n_u is the number of input channels (spatial locations with material input). We can take $n_u = n_p$ without loss of generality and define

$$\boldsymbol{B}_k = [\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{n_u}], \tag{3.5}$$

$$\boldsymbol{u}_k = [u_1, u_2, \ldots, u_{n_u}]^T, \tag{3.6}$$

where $u_i \in [0, 1]$ is the normalized material input parameter to the AM process, which is defined as a physical input quantity such as pressure applied to the deposition system at a specific spatial location. A similarly layer-varying spatial height map model for e-jet printing is given in [149]. Note that $\boldsymbol{c}_i \boldsymbol{u}_i$ is the discretized cross-section shape scaled by the magnitude of the input $\boldsymbol{u_i}$, thus the name input-to-shape response. In this chapter, we utilize a linear input-to-shape response by noting that the nonlinear effects on the input dynamics may be approximated within the operating range of the input. Examples of linear input-to-shape responses include [4, 13, 83, 86, 90, 94, 127]. While nonlinear and state-

dependent relationships (e.g., $\boldsymbol{c}_i(\boldsymbol{x}_k)$) may outperform linear models [2, 51, 106, 149], they are not discussed here for simplicity. For control applications with layer-to-layer feedback available to the controller, an idealized shape function that is spatially invariant has been utilized in many of the works in the literature [4, 83, 94, 170].

3.3.3.2 Effect of the Previous Layer

Material deposition at layer $k+1$ is added on top of the previous layer k. Therefore, the effect of the previous layer on the height evolution of the subsequent layer must be captured in the spatial dynamics of the AM process.

A *spatial register matrix* $\boldsymbol{A}_k \in \mathbb{R}^{n_g \times n_g}$ is defined by the spatial height information relationship between different spatial locations in Λ across subsequent layers (i.e., from layer k to $k+1$). Due to the physical interpretation of the height relationship, the information can be scaled by at most one, in other words, a scaling factor $\kappa \in [0, 1]$ can be applied to any height relationship.

Each $\boldsymbol{x}_k[m] \in \boldsymbol{x}_k$ denotes the *height information* of a point in Λ (i.e., height at the spatial location m *up to layer* k). Additionally, define the mapping $M : \mathbb{Z}_{>0} \times \Lambda \to Z(k, m)$, where $Z(k, m) \subseteq \{1, \ldots, n_g\}$, to map the height relationship between spatial locations so that each point in $\boldsymbol{x}_k$ is scaled and mapped to another point in $\boldsymbol{x}_{k+1}$. This mapping is constructed based on the physical interactions of the materials deposited in subsequent layers. Formally, $M(k, m) = \{w \in \mathbb{Z}_{\geq 0} | w \in Z(k, m)\}$, i.e., $M(k, m)$ is the set of locations w on layer $k+1$ where the spatial height information is related to the height of $\boldsymbol{x}_k[m]$. Then the matrix $\boldsymbol{A}_k$ is constructed as the following sum over the mapping for each location in Λ.

$$\boldsymbol{A}_k = \sum_{m=1}^{n_g} \sum_{v \in M(k,m)} \kappa_v \boldsymbol{e}_v^{n_g} (\boldsymbol{e}_m^{n_g})^T, \tag{3.7}$$

where $\boldsymbol{e}_v^{n_g}$ is the v^{th} unit basis vector of $\mathbb{R}^{n_g}$ and $\kappa_v \in [0, 1]$ is the scaling factor based on the AM process, height evolution geometry, and material properties. The scaling κ_v also may be state-dependent (e.g. $\kappa_v(\boldsymbol{x}_k)$). Note that if M is a self-mapping (i.e. $M(k, m) = \{m\}$), and $\kappa_v = 1, \forall v$, then $\boldsymbol{A}_k = \boldsymbol{I}$. As an example of self-mapping, consider AM processes that involve material curing after the deposition of a layer so that after a deposited layer is cured and solidified as part of the processing of a single layer (see (A3)), its height information is additive (i.e., mapped to the next layer with $\kappa = 1$) [83, 194]. For additional layer-to-layer models with self-mapping see [86, 90, 127]. For extrusion-based processes (e.g., [2, 13, 93, 141]) or other deposition processes (i.e., remelting phenomenon [170]), the

height information in the previous layer may be scaled in relation to the next layer. In the case study, we provide a special case for the FDM process where we constrain $\kappa_v \in [0, 1)$ to model the nonzero L2L intersection behavior of the process [2, 13].

Based on (A4), height evolution in the process is due to the material input and is bounded between adjacent layers (e.g. $k-1$ and k). Each row of the spatial register matrix $\boldsymbol{A}_k[i, \cdot]$ relates the height information from previous layer $\boldsymbol{x}_k$ to the location in the subsequent layer $\boldsymbol{x}_{k+1}[i]$, thus the condition $||\boldsymbol{A}_k||_\infty \leq 1$ states that this relationship is bounded.

Remark 3.7. *Based on the definition in (3.7), the following induced matrix norm relationship holds for all LLSV systems.*

$$||\boldsymbol{A}_k||_1 \leq \vartheta \max_{m \in [1, n_g]} \Big\{ |M(k, m)| \Big\},$$

where, $\vartheta \in \mathbb{R}_{\geq 0}$ *is given by* $\vartheta = \max\{\kappa_v\} \leq 1$.

3.3.4 Switched Affine System Reformulation

Based on the assumptions (A2) and (A4), $\boldsymbol{u}_k$ and $\boldsymbol{B}_k$ are predefined for a specific geometry and process based on the fixed deposition paths that will be deposited at a layer. The model given in (3.4) can be reformulated into a simpler form as a switched affine system (SAS) with predefined switches. This representation allows for grouping similar layers together and creates a succinct formulation of the spatial dynamics with a predefined control input. Additionally, if the geometry of adjacent (e.g. $k-1$ and k) layers are identical, $\boldsymbol{u}_{k-1}$ and $\boldsymbol{u}_k$ are identical. In practice, an AM process typically includes multiple layers with geometries that are identical. Let the tuple (k, Ω_i) denote the layer k belonging to layer group Ω_i, where Ω_i is the set of all layers in the process that have identical inputs such that $\boldsymbol{u}_k = \boldsymbol{u}_{\bar{k}}, \forall (k, \bar{k}) \in \Omega_i$. The set of all such tuples for an AM process with n_Ω layer groups is defined as

$$\Omega = \{(k, \Omega_i) \mid k = 1, \ldots, n_\ell, i \in [1, n_\Omega]\}, \tag{3.8}$$

where each layer is mapped to one and only one layer group Ω_i. An AM process may contain multiple layer groups (see Fig. 3.1), so the map $\sigma(k) : k \to \Omega_i$ maps a layer k to its respective layer group for $(k, \Omega_i) \in \Omega$. A *switch* occurs when $\sigma(k)$ and $\sigma(k+1)$ map the subsequent layers to different Ω_i. As a result, the LLSV spatial dynamics can be

represented as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}(\sigma(k), \sigma(k-1))\boldsymbol{x}_k + \boldsymbol{\mu}(\sigma(k)), \tag{3.9}$$

where the switch of the spatial register matrix depends on the layer group of the current and previous layers ($i.e., k$ and $k-1$). Note that if $\sigma(k-1) = \sigma(k)$, then the dependency on $\sigma(k-1)$ is redundant and may be omitted for brevity. The layer group dependent input $\boldsymbol{\mu}(\sigma(k)) \in \mathbb{R}^{n_g}$ is defined by $\boldsymbol{\mu}(\sigma(k)) = \boldsymbol{B}_k \boldsymbol{u}_k$ for all $(k, \sigma(k)) \in \Omega$.

As many practical AM processes have layer groups (e.g., task groups in [81]), the representation in (3.9) allows one to design controllers independently for each layer group, which may result in simpler controller formulations due to the dynamical similarities within a layer group. In such cases, the controller for each layer group would switch whenever the layer group switches, and the control design should ensure stability during the switch. A layer-to-layer controller utilizing the idea of layer groups and switching between groups is presented in Chapter IV. Additionally, using layer groups, non-constant layer heights in the AM process may be grouped together to design individual closed-loop controllers for different layer heights.

3.3.5 Uncertainty in the LLSV Model

The AM process model has uncertainty due to material properties, discretization errors, environmental conditions, and un-modeled disturbances. In this section, the effect of uncertainty on the LLSV model is represented as a spatial noise distribution in the form of a Gaussian Process (GP). A spatial noise distribution is a multivariate Gaussian distribution in which each dimension of the multivariate distribution represents a spatial location $\boldsymbol{\lambda} \in \Lambda$. Based on the GP framework, it is possible to define the mean and covariance of the uncertainty on the discretization Λ. Let $\bar{\boldsymbol{\lambda}} = vec(\boldsymbol{\Lambda})$ denote the vector with the locations in Λ, $\boldsymbol{m}_{\bar{\boldsymbol{\lambda}}} \in \mathbb{R}^{n_g}$ denote the mean function, and $\boldsymbol{\Sigma}_{\bar{\boldsymbol{\lambda}}} \in \mathbb{R}^{n_g \times n_g}$ denote the covariance function for the spatial noise distribution. Then, the uncertainty as a spatial noise distribution is given by the GP $\boldsymbol{\nu}(\bar{\boldsymbol{\lambda}}) \sim \mathcal{N}(\boldsymbol{m}_{\bar{\boldsymbol{\lambda}}}, \boldsymbol{\Sigma}_{\bar{\boldsymbol{\lambda}}})$.

The GP is assumed to be *stationary with respect to the layer-to-layer spatial dynamics*, for an AM process. This means that the mean and covariance of the GP remains the same over the layer domain (and over Λ). This assumption is not restrictive since the uncertainties in the spatial dynamics are most likely to be functions of the space, thus, invariant to the height change in the AM process.

Define $\mathcal{I}(\cdot)$ as the element-wise indicator function for non-zero elements of a matrix. Then, $\mathcal{I}(\boldsymbol{\mu}(\sigma(k))) \in \{0, 1\}^{n_g}$ is a vector with ones in the locations with material deposition

for layer k and zeros elsewhere. Also define $\boldsymbol{\mathcal{I}}_k \triangleq diag(\mathcal{I}(\boldsymbol{\mu}(\sigma(k))))$ as a diagonal matrix of size $n_g \times n_g$. Then, the random vector $\boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k))) \sim \boldsymbol{\mathcal{I}}_k \mathcal{N}(\boldsymbol{m}_{\boldsymbol{\mu}(\sigma(k))}, \boldsymbol{\Sigma}_{\boldsymbol{\mu}(\sigma(k))})$ gives the spatial noise for a specific deposition geometry with input $\boldsymbol{\mu}(\sigma(k))$. The resulting LLSV model with the uncertainty term as a spatial noise distribution is given as the following.

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}(\sigma(k), \sigma(k-1))\boldsymbol{x}_k + \boldsymbol{\mu}(\sigma(k)) + \boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k))). \tag{3.10}$$

3.4 Layer-to-layer Stability

In this section, the layer-to-layer stability of the system in (3.9) (equivalently (3.4)) is investigated. Additionally, robustness margins for layer-to-layer stability are given for the uncertainty reformulation of the system in (3.10).

3.4.1 Layer-wise Regularity

In order to evaluate the layer-to-layer stability of the LLSV system, some additional measures for individual layers must be defined. A reference spatial state trajectory $\boldsymbol{x}^d$ for the AM process is defined based on the design of the desired end geometry, discretization Λ, and deposition path $\bar{p}(k, \bar{\gamma}(\Lambda))$ as $\boldsymbol{x}^d = \{\boldsymbol{x}_1^d, \ldots, \boldsymbol{x}_{n_\ell}^d\}$, where n_ℓ denotes the total number of layers. Then, define the maximal admissible height deviation from $\boldsymbol{x}_k^d$ as $\boldsymbol{\omega}_k(\boldsymbol{x}_k^d) \in \mathbb{R}_{\geq 0}^{n_g}$ as admissible bounds for the layer-wise AM process (i.e. tolerance). Note that $\boldsymbol{\omega}_k$ (dependence on $\boldsymbol{x}_k^d$ is omitted for brevity) is a layer-varying parameter so that the admissible bounds on the spatial state may be varied between layers based on the geometry or other considerations.

Definition 3.8 (Layer spatial conformance)**.** A layer k with the spatial dynamic state $\boldsymbol{x}_k$ is layer spatial conforming if $\hat{\boldsymbol{\omega}}_k^0(\boldsymbol{x}_k^d) \preceq \boldsymbol{x}_k^d - \boldsymbol{x}_k \preceq \hat{\boldsymbol{\omega}}_k^1(\boldsymbol{x}_k^d)$.

Here, $\preceq$ denotes element-wise less-than or equal-to. Layer spatial conformance denotes if the system trajectories are within the admissible bounds. The bounds $\hat{\boldsymbol{\omega}}_k^0(\boldsymbol{x}_k^d)$ and $\hat{\boldsymbol{\omega}}_k^1(\boldsymbol{x}_k^d)$ over the discretization Λ are design variables that are determined based on the prescribed layer height h_ℓ. By prescribing $\hat{\boldsymbol{\omega}}_k^0(\boldsymbol{x}_k^d)$ and $\hat{\boldsymbol{\omega}}_k^1(\boldsymbol{x}_k^d)$ independently, non-symmetric tolerance bounds may be prescribed. For the remainder of the chapter we utilize symmetric tolerances (i.e., $\hat{\boldsymbol{\omega}}_k^0(\boldsymbol{x}_k^d) = -\hat{\boldsymbol{\omega}}_k^1(\boldsymbol{x}_k^d)$) for simplicity. The following definition describes whether the trajectory $\{\boldsymbol{x}_k\}_{k=k_0}^{n_k}$ follows $\boldsymbol{x}^d$ within the admissible bounds.

Definition 3.9 (Layer-wise regularity [13])**.** An AM process is said to be layer-wise ω-regular at layer k if $||\boldsymbol{x}_k^d - \boldsymbol{x}_k||_2 \leq \omega$.

Layer-wise regularity states that the spatial trajectories of the LLSV system are within a tube of radius ω around $\boldsymbol{x}^d$. The numerical value of ω may be defined based on specific design considerations for part functionality (e.g. mechanical, electrical, or optical properties). An example of ω for dimensional performance is provided in the case study (Section 3.5). Using the definitions of spatial conformance and layer-wise regularity, layer-to-layer stability definitions are given next.

3.4.2 Layer-to-layer Stability Definitions

An important property of an AM process is its geometric stability. That is the ability of subsequent layers to be built on top of the existing layers. Define $\tilde{\boldsymbol{x}}_k = \boldsymbol{\mathcal{I}}_{k-1}\boldsymbol{x}_k$ as a vector with height information of the spatial locations with height change on layer k, defined by $\boldsymbol{\mu}_{k-1}$, and zeros elsewhere. The vector $\tilde{\boldsymbol{x}}_k$ denotes the height of the locations in Λ that had deposition for the current layer.

Definition 3.10 (Layer-to-layer geometric stability [13])**.** An LLSV system is said to be layer-to-layer geometrically stable (L2LGS) if $||\boldsymbol{x}_{k-1}||_\infty < \min_j\{|\tilde{\boldsymbol{x}}_k[j]| > 0\}$, for all $k \in (1, n_\ell]$.

The operator $\min_j\{|\tilde{\boldsymbol{x}}_k[j]| > 0\}$ denotes the magnitude of the minimum non-zero element in the vector $\tilde{\boldsymbol{x}}_k$, with element-wise absolute value operator $|\cdot|$. Therefore, in an L2LGS system, the minimum height at the spatial locations with deposition for layer k is greater than the maximum height in layer $k-1$. Intuitively, the L2LGS condition ensures that the current spatial state $\boldsymbol{x}_k$ is always "above" the preceding layer, providing a natural condition for geometrical stability of the printed part.

Remark 3.11. *The LLSV system with (A2) is layer-to-layer geometrically stable by design since the layer geometries and path planning for the AM process follow this stability condition to create reference trajectories $p(k, \gamma)$ for all $k \in (1, n_\ell]$, which in turn defines $\boldsymbol{x}^d$. Without loss of generality, the L2LGS condition holds for all LLSV systems analyzed in this chapter.*

Going back to the graph G interpretation of the discretization Λ, any spatial state $\boldsymbol{x}_k$ denotes the height on the nodes of G up to layer k. The layer-to-layer geometric stability condition requires the material deposition at the subsequent layer $(k+1)$ to be sufficiently supported from below (at layer k). Sufficient support is the least amount of material present at a certain location $\boldsymbol{\lambda}(m, n) \in \Lambda$ in layer k, so that the deposition on the subsequent layer $k+1$ is layer-to-layer geometrically stable according to Definition 3.10. Sufficient support depends on the specific AM technology, geometry, and material properties.

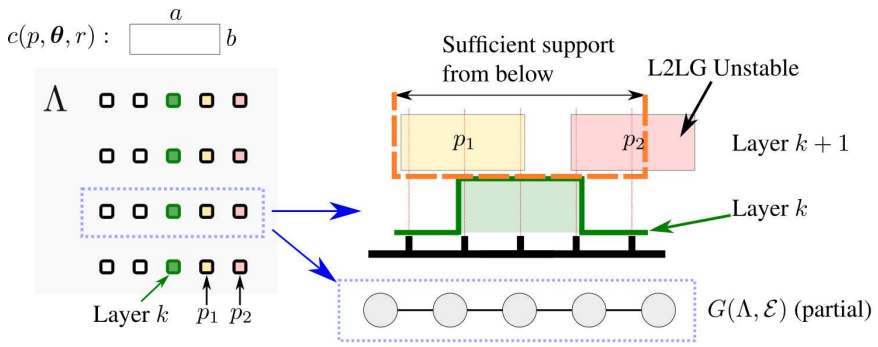


Figure 3.3: Left: Top view of the deposition process with a rectangular shape function. the green path labeled with Layer k is the currently deposited layer. The deposition path p_1 at layer $k+1$ is sufficiently supported from below, whereas p_2 at layer $k+1$ is not sufficiently supported from below. Right: Cross-sectional view of the deposition process at layers k and $k+1$ illustrates the sufficient support condition for the example. The partial graph G of the process is given at the bottom to illustrate the adjacency between the spatial locations within the dotted rectangle in Λ.

Consider the layer-to-layer dynamics illustrated in Fig. 3.3. Let $\boldsymbol{J}_k$ define the adjacency matrix of the graph G and the shape function of the process be a rectangular one. We may associate the region of sufficient support in Fig. 3.3 with the adjacency of the graph G. In the given case, material deposited in layer k at a location λ supports the material deposition on the spatial locations immediately adjacent to it at layer $k+1$. Note that the adjacency relationship is a function of the discretization size, shape function, and the physical properties of the material and process. Therefore, the deposition on layer k in the figure *provides sufficient support* for the deposition path p_1 on layer $k+1$, but the path p_2 becomes L2L geometrically unstable. Based on Remark 3.11, we assume that the sufficient support condition holds for all LLSV systems analyzed in this chapter.

The layer-to-layer stability of an LLSV system is then defined based on the definitions of regularity and stability.

Definition 3.12 (Layer-to-layer stability)**.** An LLSV system that is layer-to-layer geometrically stable is said to be

1. Layer-to-layer stable (L2LS) if for any given $\delta_c > 0$, $\exists \delta_s > 0$, such that $||\boldsymbol{x}^d_{k_0} - \boldsymbol{x}_{k_0}|| < \delta_s$ implies $||\boldsymbol{x}^d_k - \boldsymbol{x}_k|| < \delta_c$ for all $k \in [k_0, n_\ell]$

2. Layer-to-layer finite stable (L2LFS) if it is layer-to-layer stable and $\exists \xi \in (0,1)$ such that $||\boldsymbol{x}^d_{k+1} - \boldsymbol{x}_{k+1}|| \leq \xi ||\boldsymbol{x}^d_k - \boldsymbol{x}_k||$ for all $k \in [k_0, n_\ell - 1]$.

Note that L2LFS denotes finite convergence of the spatial trajectories to the desired state trajectories (within a predefined precision) with a convergence rate of at most ξ. The

actual ξ for a practical system determines if perfect tracking is feasible within finitely many layers in the AM process.

While L2LS defines a stability measure for the process, we are often interested in understanding if the spatial dynamics are layer-to-layer stable with respect to the layer-wise ω-regularity.

Definition 3.13. (Layer-to-layer ω-stability) An LLSV system that is layer-to-layer stable is said to be *layer-to-layer ω-regular stable* if given $\delta_c = \omega > 0$, there exists a $\delta_s \in (0, \omega)$ so that all the spatial trajectories are layer-wise ω-regular.

Therefore, layer-to-layer stability of the spatial trajectories is a performance measure for the LLSV system to have geometric stability and stay close to a *desired spatial trajectory* $\boldsymbol{x}^d$, while L2L ω-regular stability implies that we can prescribe how "close" the system trajectory is to the desired trajectory.

Remark 3.14. *Note that L2L ω-regular stability implies that L2LS can be ensured for a desired stability bound $\omega > 0$ for all layers and thus is more stringent than the nominal L2LS. Characterizing the conditions under which an LLSV system is L2LS and L2L ω-regular stable is left for future work.*

Another important aspect is the relationship between the layer-to-layer stability and the stability of the error dynamics of the system, highlighted in the following remark.

Remark 3.15. *While L2LS provides a framework to characterize the system performance, it is not readily compatible with the usual stability analysis tools as* L2LS is defined with respect to a desired trajectory *rather than an equilibrium for the unforced system. Therefore, by defining the error dynamics of the LLSV system with respect to a desired spatial trajectory, we utilize the transformed system dynamics to draw parallels between the layer-to-layer stability framework and the well-known Lyapunov stability.*

Without loss of generality, consider the error dynamics of (3.9) for a single Ω_i, which we assume to be power-series bounded for the simplicity of discussion. Define the error state $\boldsymbol{\eta}_k = \boldsymbol{x}_k^d - \boldsymbol{x}_k$ and the dynamics of the error as $\boldsymbol{\eta}_{k+1} = \boldsymbol{A}\boldsymbol{\eta}_k$, where we use the shorthand $\boldsymbol{A}$ for $\boldsymbol{A}_k, k \in \Omega_i$. The following lemma is given as the main result for layer-to-layer geometric stability of an LLSV system.

Lemma 3.16. *An LLSV system without noise, given in (3.9), with the error state as $\boldsymbol{\eta}_k = \boldsymbol{x}_k^d - \boldsymbol{x}_k$ is layer-to-layer geometrically stable if and only if there exists $c_0, c_1 > 0$ such that*

$$\begin{bmatrix} I & (1-c_0)I \\ -c_1 I & (1-c_1)I \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k \\ \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \end{bmatrix} \succeq 0, \quad \forall k \in [1, n_\ell - 1]$$

Proof. Define $\boldsymbol{\pi}_1 = \bar{\boldsymbol{x}}_k^d - c_0\bar{\boldsymbol{x}}_k$ and $\boldsymbol{\pi}_2 = \bar{\boldsymbol{x}}_k - c_1\bar{\boldsymbol{x}}_k^d$, where $\bar{\boldsymbol{x}}_k^d \coloneqq \boldsymbol{x}_k^d - \boldsymbol{x}_{k-1}^d$ and $\bar{\boldsymbol{x}}_k \coloneqq \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$. Then, decompose $\boldsymbol{\pi}_i = \boldsymbol{\pi}_i^+ - \boldsymbol{\pi}_i^-, \quad i = 1, 2$, where $\boldsymbol{\pi}_i^+ = \max\{0, \boldsymbol{\pi}_i\}$ and $\boldsymbol{\pi}_i^- = \max\{0, -\boldsymbol{\pi}_i\}$ with the element-wise $\max$ operator. Let $c_0 = m\left(\bar{\boldsymbol{x}}_k^d\right)/||\bar{\boldsymbol{x}}_k||_1$, where $m(\cdot)$ denotes the smallest non-zero element of a vector. Similarly, let $c_1 = m\left(\bar{\boldsymbol{x}}_k\right)/||\bar{\boldsymbol{x}}_k^d||_1$.

Now, observe that with the chosen scaling factors c_0, c_1, non-zero elements of $\boldsymbol{\pi}_1^+$ denote the spatial locations at layer k that either has desired deposition or voids (deposition is desired, but there is none). The non-zero elements of $\boldsymbol{\pi}_1^-$ denote the spatial locations with extra deposition (deposition is not desired at these locations). Similarly, non-zero elements of $\boldsymbol{\pi}_2^+$ denote desired and extra depositions, and non-zero elements of $\boldsymbol{\pi}_2^-$ denote voids.

L2LGS dictates that deposition at a layer must be exactly at the desired locations that are prescribed by the L2LGS reference $\boldsymbol{x}_k^d$ for each k. Thus a necessary condition for L2LGS is $||\boldsymbol{\pi}_1^-|| + ||\boldsymbol{\pi}_2^-|| = 0$, which is possible only if $\boldsymbol{\pi}_1^- = \boldsymbol{\pi}_2^- = \mathbf{0}$. In that case, we have $\boldsymbol{\pi}_i = \boldsymbol{\pi}_i^+ \succeq 0$ that results in $\bar{\boldsymbol{x}}_k^d - c_0\bar{\boldsymbol{x}}_k = \boldsymbol{\pi}_1^+ \succeq 0$ which is equivalently $\boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k + (1 - c_0)I\bar{\boldsymbol{x}}_k \succeq 0$. Similarly we derive $-c_1 I\left(\boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k\right) + (1 - c_1)I\bar{\boldsymbol{x}}_k \succeq 0$. Combining the last two inequalities gives the desired result.

To show the reverse direction, suppose there exists a L2LGS $\bar{\boldsymbol{x}}_k$ with voids. Then it must be that $\bar{\boldsymbol{x}}_k - c_1\bar{\boldsymbol{x}}_k^d = \boldsymbol{\pi}_2 \succeq 0$. Due to the void, there exists at least one direction j in the vector $\bar{\boldsymbol{x}}_k$ which is zero, but is non-zero in $\bar{\boldsymbol{x}}_k^d$. This means that there exists no $c_1 > 0$ that can make $\bar{\boldsymbol{x}}_k[j] - c_1\bar{\boldsymbol{x}}_k^d[j] \geq 0$, therefore $\boldsymbol{\pi}_2^- \neq 0$, which results in a contradiction with $\boldsymbol{\pi}_1^- = \boldsymbol{\pi}_2^- = \mathbf{0}$. A similar analysis follows for extra depositions but is omitted here for brevity, which concludes the proof. □

Then a formal relationship between the stability of the error dynamics and the layer-to-layer stability is given as follows.

Lemma 3.17. *The equilibrium point $\boldsymbol{\eta}_k = \mathbf{0}$ at layer k for an LLSV system given in (3.4) with the error state $\boldsymbol{\eta}_k = \boldsymbol{x}_k^d - \boldsymbol{x}_k$ satisfying Lemma 3.16 is said to be:*

1. *Layer-to-layer stable if and only if the equilibrium $\boldsymbol{\eta}_k = \mathbf{0}$ is stable in the sense of Lyapunov [104], meaning that for a given $\delta_c > 0$, $\exists \delta_s > 0$ such that $||\boldsymbol{\eta}_{k_0}|| < \delta_s \implies ||\boldsymbol{\eta}_k|| < \delta_c, \forall k > k_0$.*

2. *Layer-to-layer ω-regular stable if for $\delta_c = \omega$, $\exists \delta_s \in (0, \omega)$, with ω as the layer-wise regularity bound $\forall k$.*

3. *L2LFS if and only if it is linearly convergent to zero, i.e., $\exists \xi \in (0, 1)$ such that $||\boldsymbol{\eta}_{k+1}|| \leq \xi||\boldsymbol{\eta}_k||$.*

4. *Asymptotically stable, meaning the equilibrium is Lyapunov stable and $\{||\boldsymbol{\eta}_k||\}_{k=k_0}^{\infty} \to 0$ as $k \to \infty$, if it is L2LFS.*

Proof. The proof follows immediately from combining Lemma 3.16 with Definition 3.12. □

Note that the equilibrium $\boldsymbol{\eta}_k = \mathbf{0}$ at layer k implies that $\boldsymbol{x}_k = \boldsymbol{x}_k^d$. Given the monotone increasing nature of the desired spatial states $\boldsymbol{x}_k^d$, the equilibrium $\boldsymbol{\eta}_k = \mathbf{0}$ is not an unforced equilibrium since the absence of an input at layer k (e.g., $\boldsymbol{u}_k = \mathbf{0}$) implies $\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^d = \boldsymbol{\eta}_k \neq \mathbf{0}$. Therefore, the layer-to-layer stability and equilibrium stability notions are considered under the prescribed inputs to the system instead of the usual Lyapunov stability of an unforced system.

Recall that LLSV systems are strictly increasing due to the physical AM process that has additive spatial input at each layer. This property can be denoted as $\varphi_1(||\boldsymbol{x}_{k_0}||) \leq \{||\boldsymbol{x}_k||\}_{k=k_0}^{n_\ell}$, where $\varphi_1 \in \mathcal{K}$.

The findings in this section are summarized in the following:

Theorem 3.18. *For an LLSV system in (3.4) with the error state $\boldsymbol{\eta}_k = \boldsymbol{x}_k^d - \boldsymbol{x}_k$ satisfying Lemma 3.16, the following are equivalent:*

1. *The LLSV system is layer-to-layer stable.*
2. *The equilibrium $\boldsymbol{\eta}_k = \mathbf{0}$ is stable in the sense of Lyapunov.*

Additionally, the following are equivalent.

1. *The LLSV system is layer-to-layer finite stable.*
2. *The sequence $\{||\boldsymbol{\eta}_k||\}_{k=k_0}^{\infty}$ is linearly convergent to zero.*

Proofs for Theorem 4 were provided throughout the section.

3.4.3 Robustness to Uncertainty

The main theorems of this chapter are given in this Section. The robustness to layer-to-layer stability measures presented in this section serve as analysis tools to analyze if a given LLSV system will be layer-to-layer unstable in future layers. We present results on characterizing the layer-to-layer stability of a process with a given spatial disturbance, by considering the error dynamics of the process under predefined inputs.

To define the robustness of the layer-to-layer stability bound under the uncertainties in the model given in Eq. 3.10, the following lemmas are needed.

Lemma 3.19. *The expected value of $||\boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k)))||_2$ is upper bounded by the following relationship.*

$$\mathbb{E}\{||\boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k)))||_2\} \leq \left(\boldsymbol{tr}(\boldsymbol{\Sigma}_{\sigma(k)} + \boldsymbol{m}_{\sigma(k)}\boldsymbol{m}_{\sigma(k)}^T)\right)^{1/2}.$$

Proof. Noting that the vector $\boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k)))$ consists of random variables from the distribution $\mathcal{N}(\boldsymbol{m}_{\sigma(k)}, \boldsymbol{\Sigma}_{\sigma(k)})$, the expectation of the quadratic form of $\boldsymbol{z} \triangleq \boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k)))$ is $\mathbb{E}\{\boldsymbol{z}^T\boldsymbol{z}\} = \boldsymbol{tr}\left(\mathbb{E}\{\boldsymbol{z}\boldsymbol{z}^T\}\right)$. Then, the expectation is:

$$\begin{aligned}\mathbb{E}\{||\boldsymbol{z}||_2\} = \mathbb{E}\{(||\boldsymbol{z}||_2^2)^{1/2}\} &\leq \left(\mathbb{E}(||\boldsymbol{z}||_2^2)\right)^{1/2} \\ &= \left(\boldsymbol{tr}(\boldsymbol{v}(\boldsymbol{z}) + \mathbb{E}\{\boldsymbol{z}\}\mathbb{E}\{\boldsymbol{z}\}^T)\right)^{1/2} \\ &= \left(\boldsymbol{tr}(\boldsymbol{v}(\boldsymbol{z}) + \boldsymbol{m}\boldsymbol{m}^T)\right)^{1/2},\end{aligned}$$

with $\boldsymbol{v}(\cdot)$ denoting the variance matrix, which uses Jensen's inequality and the fact that square-root is a concave function to derive the required result. □

Lemma 3.20. *For the induced norm of the register matrix of an LLSV system, the following inequality holds for some $\vartheta' \in (0, 1]$.*

$$||\boldsymbol{A}_k||_2 \leq \sqrt{\vartheta' \max_{m\in[1,n_g]} \left\{|M(k,m)|\right\}},\ \forall k \in [1, n_\ell].$$

Proof. The proof follows immediately by applying Hölder's inequality with the conditions given in Remark 3.7, $M(k, m)$ defined in Section 3.3.3.2, and $\vartheta' = (max\{\kappa_v\}||\boldsymbol{A}||_\infty) \leq 1$. □

Theorem 3.21. *In a single layer group with $\sigma(k') \to \Omega_i,\ \forall k' \in [k_0, k_f]$ that is $\tilde{\omega}$-regular at layer k_0, and the state register matrix $\tilde{\boldsymbol{A}} \triangleq \boldsymbol{A}(\sigma(k), \sigma(k-1))$ defined over the range of layers k', the LLSV system in (3.10) is layer-to-layer $\tilde{\omega}$-regular stable at layer k_f in expectation if $\forall\zeta \in [1, k_f - k_0]$, we have*

$$||\tilde{\boldsymbol{A}}^\zeta||_2||\boldsymbol{\eta}_{k_0}||_2 + \left(\boldsymbol{tr}(\boldsymbol{\Sigma}' + \boldsymbol{m}'\boldsymbol{m}'^T)\right)^{1/2} \leq \tilde{\omega},$$

where $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}_{\sigma(k)} + \tilde{\boldsymbol{A}}\boldsymbol{\Sigma}_{\sigma(k)}\tilde{\boldsymbol{A}}^T + \ldots + \tilde{\boldsymbol{A}}^{\zeta-1}\boldsymbol{\Sigma}_{\sigma(k)}\left(\tilde{\boldsymbol{A}}^{\zeta-1}\right)^T$, $\boldsymbol{m}' = P_{\zeta-1}(\tilde{A})\boldsymbol{m}_{\sigma(k)}$, $P_{\zeta-1}(\tilde{A})$ is a matrix polynomial up to power $\zeta - 1$, and $\tilde{\omega} > 0$ is the layer-to-layer stability bound (δ_c). Furthermore if $\boldsymbol{m}_{\sigma(k)} = \boldsymbol{0}$ then

$$||\tilde{\boldsymbol{A}}^\zeta||_2||\boldsymbol{\eta}_{k_0}||_2 + ||\boldsymbol{L}||_F \leq \tilde{\omega}, \tag{3.11}$$

where $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{\Sigma}'$.

Proof. The proof is given for $\zeta = k_f - k_0$ since the same analysis follows $\forall\zeta \in [1, k_f - k_0]$.

Define $\tilde{\boldsymbol{\nu}} \triangleq \boldsymbol{\nu}(\boldsymbol{\mu}(\sigma(k)))$. The error dynamics of the LLSV system are given by

$$\boldsymbol{\eta}_{k_f} = \tilde{\boldsymbol{A}}^{k_f-k_0}\boldsymbol{\eta}_{k_0} + \textstyle\sum_{i=k_0}^{k_f-1} \tilde{\boldsymbol{A}}^{k_f-i-1}\tilde{\boldsymbol{\nu}}.$$

The sum in this equation is a matrix polynomial up to degree $\zeta - 1$ given as

$$\textstyle\sum_{i=k_0}^{k_f-1} \tilde{\boldsymbol{A}}^{k_f-i-1}\tilde{\boldsymbol{\nu}} = I\tilde{\boldsymbol{\nu}} + \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\nu}} + \ldots + \tilde{\boldsymbol{A}}^{\zeta-1}\tilde{\boldsymbol{\nu}},$$

which has the expected value $\mathbb{E}\{\sum_{i=k_0}^{k_f-1} \tilde{\boldsymbol{A}}^{k_f-i-1}\tilde{\boldsymbol{\nu}}\} = \boldsymbol{m}' = P_{\zeta-1}(\tilde{A})\boldsymbol{m}_{\sigma(k)}$, where $P_n(\boldsymbol{A}) = I + \boldsymbol{A} + \ldots \boldsymbol{A}^n$, and covariance $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}_{\sigma(k)} + \tilde{\boldsymbol{A}}\boldsymbol{\Sigma}_{\sigma(k)}\tilde{\boldsymbol{A}}^T + \ldots + \tilde{\boldsymbol{A}}^{\zeta-1}\boldsymbol{\Sigma}_{\sigma(k)}\left(\tilde{\boldsymbol{A}}^{\zeta-1}\right)^T$. Therefore we have $\sum_{i=k_0}^{k_f} \tilde{\boldsymbol{A}}^{k_f-i-1}\tilde{\boldsymbol{\nu}} \sim \mathcal{N}(\boldsymbol{m}', \boldsymbol{\Sigma}')$. Taking ℓ_2 norms of both sides and their expectations yield the following inequalities

$$\begin{aligned} ||\boldsymbol{\eta}_{k_f}||_2 &\leq ||\tilde{\boldsymbol{A}}^{k_f-k_0}||_2||\boldsymbol{\eta}_{k_0}||_2 + ||\textstyle\sum_{i=k_0}^{k_f} \tilde{\boldsymbol{A}}^{k_f-i-1}\tilde{\boldsymbol{\nu}}||_2 \\ \mathbb{E}\{||\boldsymbol{\eta}_{k_f}||_2\} &\leq ||\tilde{\boldsymbol{A}}^{\zeta}||_2||\boldsymbol{\eta}_{k_0}||_2 + \left(\boldsymbol{tr}(\boldsymbol{\Sigma}' + \boldsymbol{m}'\boldsymbol{m}'^T)\right)^{1/2}. \end{aligned}$$

Since this defines an upper bound on the error after ζ layers, if this bound is greater than $\tilde{\omega}$, layer-to-layer $\tilde{\omega}$-regular stability cannot be guaranteed, thus the given bound. So if the bound in the theorem holds for all layers in the analysis (i.e. $\forall \zeta \in [1, k_f - k_0]$), the system is layer-to-layer $\tilde{\omega}$-regular stable at layer k_f, in expectation. This relationship concludes the proof of the first part. Now suppose $\boldsymbol{m}_{\sigma(k)} = \boldsymbol{0}$ and $\boldsymbol{\Sigma}' = \boldsymbol{L}\boldsymbol{L}^T$ (this decomposition always exists since $\boldsymbol{\Sigma}' \succeq 0$), then,

$$\begin{aligned} \mathbb{E}\{||\boldsymbol{\eta}_{k_f}||_2\} &\leq ||\tilde{\boldsymbol{A}}^{\zeta}||_2||\boldsymbol{\eta}_{k_0}||_2 + \left(\boldsymbol{tr}(\boldsymbol{L}\boldsymbol{L}^T)\right)^{1/2} \\ &\leq ||\tilde{\boldsymbol{A}}^{\zeta}||_2||\boldsymbol{\eta}_{k_0}||_2 + ||\boldsymbol{L}||_F, \end{aligned}$$

which concludes the proof. □

Based on the magnitude of ζ, the bound on the error in the LLSV system is affected by the uncertainty in the system, while the effect of the initial error norm is decreasing with the increasing horizon. Using Lemma 3.20, the value of $||\tilde{\boldsymbol{A}}^{\zeta}||_2$ can be approximated. Note that if $||\tilde{\boldsymbol{A}}||_2 < 1$, we can analyze $\zeta = k_f - k_0$ (instead of $[1, k_f - k_0]$) in Theorem 3.21 without loss of generality (since $||\tilde{\boldsymbol{A}}||_\infty < 1$). The analysis for the case where the spatial register matrix is power-series convergent is given next.

Corollary 3.22. *If in addition to Theorem 3.21 we have $\rho(\tilde{\boldsymbol{A}}) < 1$ and $\varsigma = ||\tilde{\boldsymbol{A}}||_2 < 1$, then for large enough ζ and invertible $(I - \boldsymbol{A})$, the error dynamics η_k of the LLSV system*

are bounded in expectation by

$$\begin{aligned}\mathbb{E}\{||\boldsymbol{\eta}_{k_f}||_2\} &\leq \left(\boldsymbol{tr}(\boldsymbol{\Sigma}') + ||\boldsymbol{m}''||_2^2\right)^{1/2} \\ &\leq \left(\boldsymbol{tr}(\boldsymbol{\Sigma}') + \varsigma'||\boldsymbol{m}_{\sigma(k)}||_2^2\right)^{1/2},\end{aligned}$$

where $\varsigma' = (1-\varsigma)^{-1}$, $\boldsymbol{\Sigma}'$ *as previously given, and* $\boldsymbol{m}'' = \left(I - \tilde{\boldsymbol{A}}\right)^{-1} \boldsymbol{m}_{\sigma(k)}$.

Proof. If $\rho(\tilde{\boldsymbol{A}}) < 1$ and $\varsigma = ||\tilde{\boldsymbol{A}}||_2 < 1$, then there exists a $c > 0$ where $\tilde{\boldsymbol{A}}^c \simeq 0$ and thus $\varsigma^c \simeq 0$. By choosing $\zeta \geq c$ (assuming that this is feasible in the physical process), the sum $P_{\zeta-1}(\tilde{A})$ is equal to $\left(I - \tilde{\boldsymbol{A}}\right)^{-1}$, and $||\tilde{\boldsymbol{A}}^{\zeta}||_2 = 0$. The second upper bound follows from the geometric sum of the ς and the fact that $||\tilde{\boldsymbol{A}}^{\zeta}|| \leq ||\tilde{\boldsymbol{A}}||^{\zeta}$. □

Additionally, the following corollary to Theorem 3.21 provides a probabilistic bound for estimating the L2LS of an LLSV system of the form (3.10).

Corollary 3.23. *For the system given in Theorem 3.21, if the covariance matrix has the form* $\boldsymbol{\Sigma}' = \sigma_s^2 \boldsymbol{I}$, *probability for the* ℓ_2*-norm squared of error dynamics* $\left(||\boldsymbol{\eta}_{k_f}||_2^2\right)$ *being* $\tilde{\omega}^2$*-regular at layer* k_f *is given by*

$$\mathbb{P}\left(||\boldsymbol{\eta}_{k_f}||_2^2 \leq \tilde{\omega}^2\right) = \mathbb{P}\left(\Gamma \leq \tilde{\omega}^2/\sigma_s^2\right),$$

where $\Gamma \sim \mathcal{X}_n^2(\psi)$, *is a random variable from an* n_g *degrees of freedom non-central chi-squared distribution with the non-centrality parameter* $\psi = \sigma_s^{-2}\sum_{j=1}^{n_g}\left(\boldsymbol{m}^*[j]\right)^2$ *and* $\boldsymbol{m}^* = \tilde{\boldsymbol{A}}^{\zeta}\boldsymbol{\eta}_{k_0} + \boldsymbol{m}'$, $\zeta = k_f - k_0$.

Proof. Following the proof of Theorem 3.21, the error at layer k has the multivariate normal distribution $\boldsymbol{\eta}_k \sim \mathcal{N}\left(\tilde{\boldsymbol{A}}^{\zeta}\boldsymbol{\eta}_{k_0} + \boldsymbol{m}', \boldsymbol{\Sigma}'\right)$. The ℓ_2-norm squared of a random variable from this multivariate normal distribution is distributed as a non-central chi-squared distribution with n_g degrees of freedom. For the covariance matrix of the form $\boldsymbol{\Sigma}' = \sigma_s^2\boldsymbol{I}$, the non-centrality parameter of the distribution is given by $\psi = \sigma_s^{-2}\sum_{j=1}^{n_g}\left(\boldsymbol{m}^*[j]\right)^2$. Therefore the probability of the ℓ_2-norm squared value of the error being within the stability bound squared $\tilde{\omega}^2$ is given by the cumulative distribution function of $\mathcal{X}_{n_g}^2(\psi)$. □

Theorem 3.21 is stated for a single layer group and it is a known fact that a switched system with stable subsystems may become unstable under certain switching conditions [35]. To circumvent this issue and ensure the stability of the LLSV system for all Ω_i, the following observations are made. The trajectory of the error dynamics given by $\boldsymbol{\eta}_{k+1} = \boldsymbol{A}_k\boldsymbol{\eta}_k$ is bounded under arbitrary switching if the joint spectral radius [167] of all $\boldsymbol{A}(\sigma(k), \sigma(k-1))$

in the LLSV system are power-series bounded, and the system is layer-to-layer geometrically stable. However, this is a much stronger condition than what we need in practice since the dynamics undergo a known switching sequence rather than arbitrary switching. Therefore, the stability of the switches may be analyzed off-line. Further conditions for finite convergence of the switched dynamics are given in [87] (pp. 170-173).

3.5 Case Studies on Fused Deposition Modeling

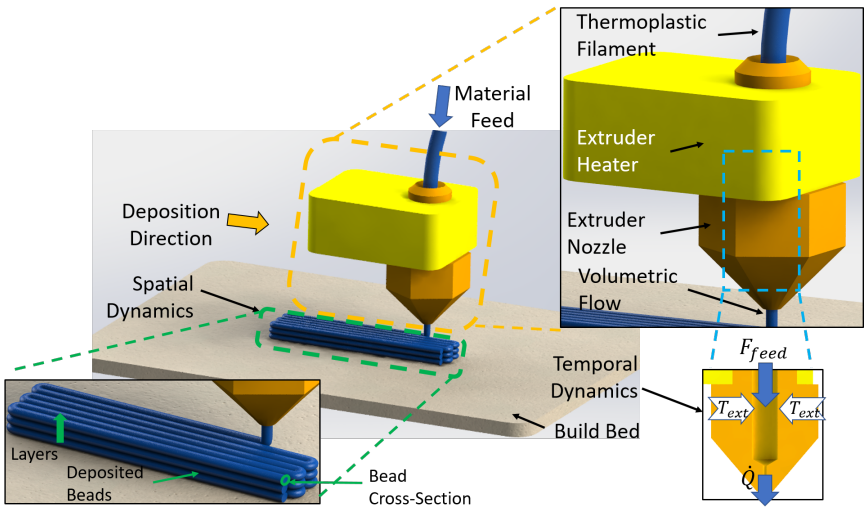


Figure 3.4: Description of fused deposition modeling [13]. F_{feed} is the material feed force for the extrusion process in the nozzle. T_{ext} is the heat supplied by the extruder heater. $\dot{Q}$ is the volumetric flow through the nozzle.

This section presents case studies for a fused deposition modeling (FDM) process modeled as an LLSV system. Definitions of layer-to-layer stability and details of the LLSV model for a specific geometry are given. Leveraging the models and the experimental setup, theoretical developments on the layer-to-layer stability in the earlier sections are compared against the experimental measurement. A schematic of an FDM process is shown in Fig. 3.4. FDM is an AM process in which a thermoplastic material is extruded through a heated nozzle in a numerically controlled deposition system. After a layer of material is deposited, either the deposition system or the build platform changes its height to accommodate the next deposition layer until all the layers of a 3D object are deposited. Refer to [141] and references therein for a survey of the FDM process.

This case study aims to demonstrate the practical use of the LLSV model and the layer-to-layer stability concepts presented in this chapter as a measure of dimensional performance. An experimental setup is used to measure the spatial state of an FDM with induced

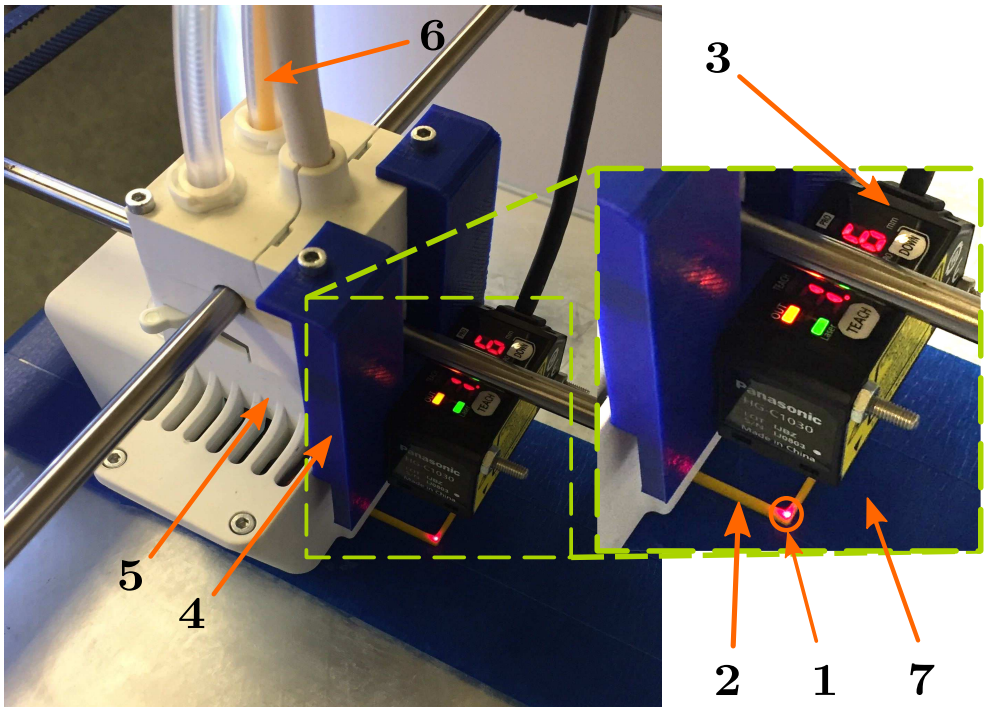


Figure 3.5: Experimental setup. **1**: laser measurement point, **2**: square shell build geometry, **3**: laser distance measurement sensor, **4**: mounting piece for the sensor, **5**: extruder head of the FDM printer, **6**: PLA filament used in the experiment, **7**: heated build plate with the painter's tape to mitigate glare.

spatial noise. The theoretical bounds derived in Theorem 3.21 are compared to experimental results.

An important issue in FDM is under-extrusion or over-extrusion of material during the deposition process, which leads to dimensional inaccuracy of the resulting printed layers and layer-to-layer stability issues. The issues in the extrusion rate may occur due to directional changes in the deposition path, (e.g., cornering), slippage of the filament material in the feeder, or insufficient heating of the deposited material. In this study, we induce spatial noise to the process to simulate the aforementioned disturbances on material extrusion. Additionally, the effect of the induced spatial disturbance increases as we move away from the center of the build plate to simulate location dependent disturbances (e.g., warping issues on the build plate).

3.5.1 Experimental setup

The experimental setup is shown in Fig. 3.5. To enable a spatial height measurement between layers, a Panasonic HG-C1030 laser point distance measurement sensor is mounted on the extruder of an Ultimaker 3 FDM printer. The laser has a $50\mu m$ spot diameter and a

Table 3.1: Experimental parameters for the case study

Variable	Value
Material	PLA @ 220°C
Build bed	Glass @ 60°C
Feedrate	$20mm/sec$
Shell width	$20.4mm$
Shell height	$5.33 \pm 0.05mm$
Layer height	$h_\ell = 0.27mm$
Design discretization	$\alpha_i = \alpha_j = 0.16mm$
Grid size	$n_i = n_j = 128$
Number of layers	$n_\ell = 20$
L2L intersection	$\bar{d} = 30.6\mu m$
Experimental discretization	$\bar{\alpha}_i = \bar{\alpha}_j = 0.2mm$
Experimental grid	$\bar{n}_i = \bar{n}_j = 103$
Extrusion per unit length	$p_e = 0.0025mm$

$10\mu m$ repeatability. An Arduino Mega connected to MATLAB on a Windows 10 machine with i7-4700 CPU is used for collecting the sensor measurements at a rate of 480 Hz.

For the case study, a square shell geometry (1 bead thickness) is additively manufactured using FDM. The geometry of the shell and its assembly are shown in Fig. 3.6. A manufacturing scenario in which the shell geometry needs to fit inside a square slot of $20.4mm$ and depth of $5.33mm$ is considered with the given dimensional tolerances shown in Fig. 3.6. Based on these dimensional specifications, layer-wise regularity and layer-to-layer ω-regular stability limits are evaluated in the following section. The experimental parameters for the case study are given in Table 3.1. By ensuring layer-to-layer ω-regular stability, we can ensure that the FDM printed shell will conform to the design specifications and deliver the desired performance in terms of dimensional accuracy. Similarly, we will conclude that layer-to-layer ω-regular unstable parts will not conform to the design specifications, and thus will be scrap. Thus, we utilize L2LS as an in-situ tool for analyzing printed part performance in this case study, which is a novel approach to understand the printed part performance in AM.

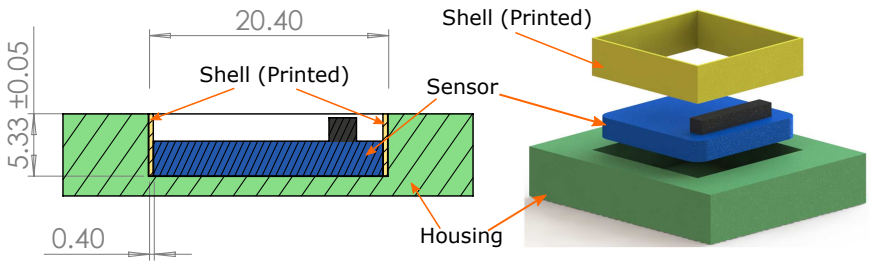


Figure 3.6: Technical drawing of the assembly of the square shell geometry (left), and the exploded view of the assembly (right). The conceptual sensor that fits inside the shell and the housing. A cross-sectional view is shown in the figure and the assembly is symmetric about the axis of the cross-sectional cut.

3.5.2 Linear Layer-wise Spatially Varying Systems for FDM

The spatial deposition path $p(k, \gamma)$ for the square shell geometry is identical for all layers $\forall k$. Consequently, there is only a single layer group $\sigma(k) = \Omega_1, \ \forall k \in [1, n_\ell]$. In FDM, deposited beads of subsequent layers form an intersection where they partially bond to create a sound structure [141]. As a result, the height evolution of the build between subsequent layers is less than the spatial input to the system, which results in a $\kappa_v < 1$ (in (3.7)). Additionally, due to inconsistencies (caused by transient dynamics of the fluid flow) in the material flow at the beginning of the process and the interaction of the material with the build plate, initial layer height is observed to be less than expected during the experiments.

3.5.2.1 Spatial register matrix for FDM

To understand the effect of height intersection, ex-situ measurements are performed on the printed parts. Experiment specimens cut across the cross-section are measured using a microscope, Fig. 3.7. Green ellipsoids are fit to each cross-section in the image and their intersection amount is calculated. In practice, FDM deposition is adjusted such that the extruder nozzle presses onto the deposited material in a layer [2]. In this case study, the deposition height is adjusted to have a gap between the nozzle tip and the deposited bead to better understand the effect of spatial noise on the size of a deposited bead. Through this study, the mean value of layer-to-layer intersection between the beads of successive layers is found to be $\bar{d} = 30.6 \mu m$.

Using the laser distance measurement sensor, the profile of the surface of a printed part with 180 layers is measured and no significant change in the intersection amount is observed.. As a result, the intersection amount is modeled as a constant amount across

layers. The scaling factor κ_v in (3.7) is given as

$$\kappa_v(\boldsymbol{x}_k[m]) = (1 - \bar{d}/\boldsymbol{x}_k[m]).$$

Since $\bar{d} << \boldsymbol{x}_k, \forall k$, the value of $\kappa_v(\boldsymbol{x}_k)$ is strictly less than one (e.g. $\kappa_v(\boldsymbol{x}_k) \in (0, 1)$).

The height information at spatial locations $\boldsymbol{\lambda}(1, m)$, $m \in [2, 125]$ in layer k are mapped to $\boldsymbol{\lambda}(1, m)$ and the neighboring spatial locations at $\boldsymbol{\lambda}(0, m)$ and $\boldsymbol{\lambda}(2, m)$ in layer $k + 1$. The spatial deposition points on the corners have their height information mapped to their surrounding points. The mapping $M(k, m)$ is created by utilizing the relationship given in the above examples for all the deposition points in the process for all layers.

$$\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k) = \sum_{m=0}^{n_g-1} \sum_{v \in M(k,m)} (1 - \frac{\bar{d}}{\boldsymbol{x}_k[m]}) \boldsymbol{e}_v^{n_g} (\boldsymbol{e}_m^{n_g})^T. \tag{3.12}$$

Since $\kappa_v(\boldsymbol{x}_k) \in (0, 1)$, it is straightforward to show that $||\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k)||_\infty < 1$ and $\rho(\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k)) < 1$. Furthermore, the spectral radius $\rho(\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k)), k \in [1, n_\ell]$ is upper bounded by $\kappa_v(\boldsymbol{x}_{n_\ell}) < 1$ for the last layer n_ℓ since the spatial state trajectory is nondecreasing.

Further details of model creation to represent a bead cross-section at three locations in Λ are provided in Appendix A. For simplicity of analysis, we consider the deposited bead cross sections only along the spatial locations on the deposition path $p(k, \bar{\gamma}(\Lambda))$. We employ the mapping $M(k, m) = \{m\}$ with $\kappa_v(\boldsymbol{x}_k) \in (0, 1)$ described as above so that $\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k) = (1 - \bar{d}/\boldsymbol{x}_k)\boldsymbol{I}$, where the division is element-wise.

3.5.2.2 The effect of noise

To illustrate the layer-to-layer stability concept, known spatial disturbances are induced on the deposition process and the results are experimentally measured. A quadratic positive semi-definite spatial noise function is used in the case study. The function is given as

$$\boldsymbol{v}(\boldsymbol{\xi}, \mu) = \mu/\upsilon^2(\boldsymbol{\xi}_1^2 + \boldsymbol{\xi}_2^2), \tag{3.13}$$

where $\upsilon \in \mathbb{R}$ is a scale correction factor (e.g. for a spatial area of $20mm \times 20mm$ around the origin, $\upsilon = 10$), $\boldsymbol{\xi} \in \mathbb{R}^2$ is the spatial variable and μ is the amplitude of the noise function. As the spatial location of the deposition moves away from the center of the build plate, increased noise is expected due to errors in the flatness of the build plate.

The induced noise is added as a disturbance to the extrusion command (i.e., adding noise terms to the Extrusion (E) axis references) in the G-Code for the FDM machine. Due

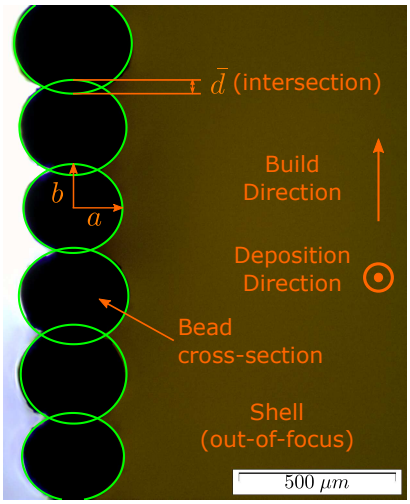


Figure 3.7: Cross-sectional cutout of one of the deposited square shell specimen under a microscope. Green ellipsoids are fit to the cross-sections to study the bead intersection between the subsequent layers.

to the on-line computational capacity available in the FDM machine, the design discretization of $0.16mm$ is down-sampled 1.25 times so that $\bar{\alpha}_i = \bar{\alpha}_j = 0.2mm$. As a result, the discretization used for G-Code generation has $\bar{n}_i = \bar{n}_j = 103$. The G-Code discretization is denoted by $\bar{\Lambda}$ and the deposition path is denoted by $\bar{p}(k, \bar{\gamma}(\bar{\Lambda}))$, with the vector $\bar{\boldsymbol{p}}$ as the vector with locations of deposition. Note that $diag(\boldsymbol{\mathcal{I}}_k) = \mathcal{I}(\sigma(k)) = \bar{\boldsymbol{p}}, \forall k$.

The spatial noise over the deposition path is given by the distribution $\mathcal{N}(\boldsymbol{v}'(\boldsymbol{\xi}, \mu), \sigma_e^2 \boldsymbol{I})$ where, $\boldsymbol{v}'(\boldsymbol{\xi}, \mu) = \boldsymbol{v}(\bar{p}(k, \bar{\gamma}(\bar{\Lambda})) - \bar{\boldsymbol{\lambda}}(52, 52), \mu)$, $\boldsymbol{\xi}_{(\cdot)} \in \bar{\Lambda}$, $\bar{\boldsymbol{\lambda}}(52, 52)$ is picked as the center of the square deposition discretization in $\bar{\Lambda}$. The spatial noise is then given as:

$$\boldsymbol{\nu}(\mu) \sim \boldsymbol{\mathcal{I}}_k \mathcal{N}(\boldsymbol{v}'(\boldsymbol{\xi}, \mu), \sigma_e^2 \boldsymbol{I}), \quad \forall k \in [1, n_\ell] \tag{3.14}$$

A standard deviation of $\sigma_e = 6.62 \times 10^{-4} mm$ is applied to all of the points, which corresponds to 0.25 of the unit filament extrusion length ($p_e = 0.0025mm$ between each point in $\bar{p}(k, \bar{\gamma}(\bar{\Lambda}))$).

The input to the system is defined as $\boldsymbol{u}_k = \boldsymbol{\mathcal{I}}_k \mathbf{0.0025}$. The spatial noise $\boldsymbol{\nu}(\mu)$ is added to the input. To derive a linear model for the expected effect of induced Gaussian noise, a first order approximation of the nonlinear input dynamics around the nominal layer height is derived as a function of the noise amplitude variable μ as $\bar{b}(\boldsymbol{u}_k + \mathbb{E}\{\boldsymbol{\nu}(\mu)\})|_{\boldsymbol{u}_k = p_e} \simeq 0.267 + 1.0962\mu$ with a fit residual corresponding to less than $2.5\mu m$. Using the linear approximation, the LLSV dynamics are

$$\boldsymbol{x}_k = \boldsymbol{A}(\sigma(k), \boldsymbol{x}_k)\boldsymbol{x}_k + \boldsymbol{\mathcal{I}}_k \left(\mathbf{0.267} + \mathbf{1.0962}\boldsymbol{\nu}(\mu)\right). \tag{3.15}$$

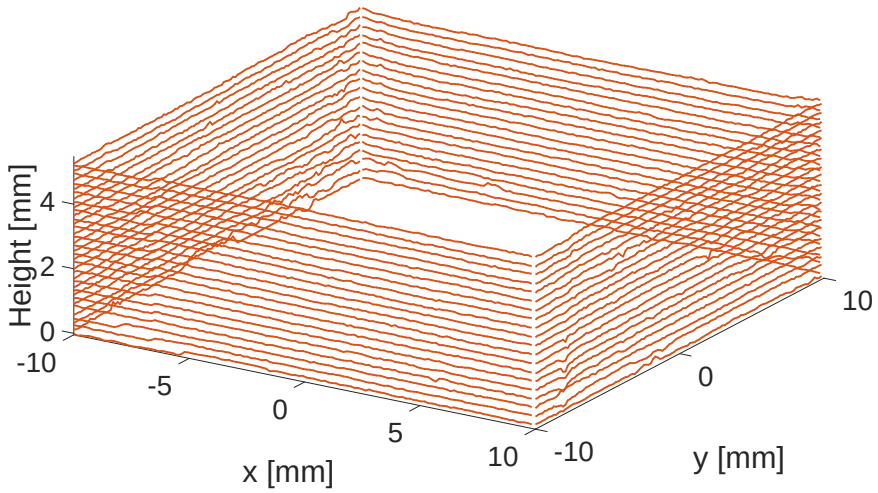


Figure 3.8: Measurement data for one of the nominal prints over $\bar{\Lambda}$. At each layer, the deposition starts from the point $(10, 10)$ and follows the corners $(-10, 10)$, $(-10, -10)$, and $(10, -10)$.

Without loss of generality, we employ a $\boldsymbol{B}$ matrix for the noise input as $\boldsymbol{B} = \boldsymbol{\mathcal{I}}_k 1.0962\boldsymbol{I}$ and denote the transformed input noise as $\bar{\boldsymbol{\nu}}(\mu) \sim \mathcal{N}(\boldsymbol{B}\boldsymbol{v}'(\boldsymbol{\xi}, \mu), \boldsymbol{B}\sigma_e^2\boldsymbol{I}\boldsymbol{B}^T)$.

3.5.2.3 Definition of L2L stability bounds

Based on the dimensional tolerances shown in Fig. 3.6, the upper and lower deviation limit of $0.05mm$ for the printed design is given. To ensure that the printed part fits within the tolerances, layer-to-layer stability bounds are chosen as $\hat{\boldsymbol{\omega}}_k = \mathbf{0.05}, \forall k \in [1, n_\ell]$, and $\tilde{\omega} = ||\mathbf{0.05}||_2$ for the layer-wise regularity conditions and to ensure layer-to-layer ω-regular stability for all layers in the process (when δ_c in Definition 3.12 is chosen as $\tilde{\omega}$).

3.5.2.4 Experimental Procedure

Layer height for layers $k \in [1, n_\ell]$ is identically prescribed as $h_\ell = 0.27mm$. A set of experiments is conducted for the deposition process without any induced spatial noise on the system. Then, experiments with induced spatial noise are conducted (dynamics as in (3.15)). The value of μ is varied between $1.5p_e$ to $3.9p_e$ in $0.2p_e$ increments for a total of 13 levels of μ. For each spatial noise function, four experiments are conducted and in-situ measurements of all the deposited layers are performed with the experimental setup. The bead-center heights are measured by centering the laser sensor on the deposited beads and scanning along the deposition path in the experiments. A total of 57 parts are printed for the experiments and data for a total of 1140 layers are collected. It is important to note that there is no closed-loop control implemented for any of the experiments as it is beyond the

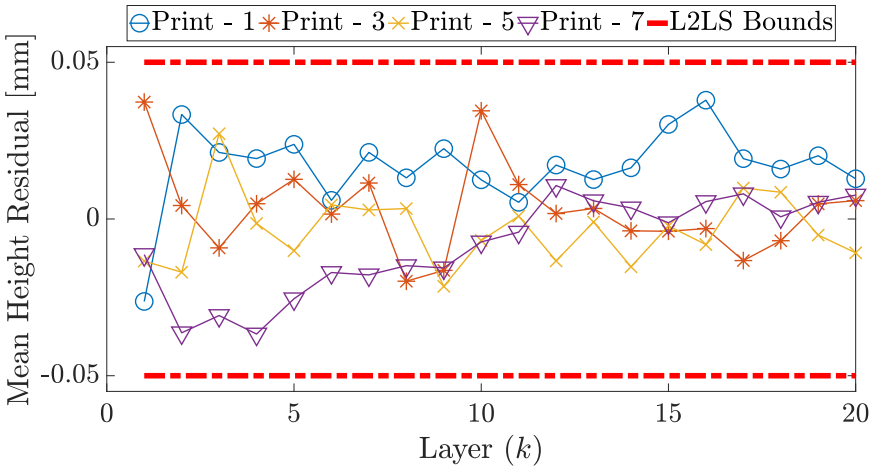


Figure 3.9: Residual of the mean layer height for nominal deposition case without induced spatial noise. Mean values from four nominal prints are shown all 20 layers in the deposition process. The layer-to-layer ω-regular stability bounds for the experiment are shown with red dashed lines.

scope of the presented work.

3.5.3 Results

3.5.3.1 Nominal case without induced noise

The mean layer height of the seven nominal experiments is taken as the desired layer height profile $\boldsymbol{x}^d$. The mean height residuals between the desired height profile and four of the nominal prints are shown in Fig. 3.9. The measurements for the nominal case have an average (for each layer over seven experiments) standard deviation of $0.0175mm$. Note that this value is the statistical standard deviation and does not reflect the actual resolution of the measurement system ($10\mu m$). The desired height profile $\boldsymbol{x}^d$ is defined as the mean of nominal trajectories to mitigate the effect of inherent disturbances and noise in the experimental process. Experimental height measurement data for one of the nominal parts over the discretization $\bar{\Lambda}$ is shown in Fig 3.8. At each layer, the deposition starts from the point $(10, 10)$ and follows the corners $(-10, 10)$, $(-10, -10)$, and $(10, -10)$.

3.5.3.2 Experimental results with induced noise

Experimental results for seven different spatial noise functions are given in Fig. 3.10. Four parts are printed for each spatial noise corresponding to different values of μ. The mean layer height for each of the layers is compared to the desired height profile defined by the nominal case to evaluate the mean height residuals. One standard deviation of the

mean values from the four printed parts of each μ value is shown with the filled colors around the mean values. The layer-to-layer stability bounds are shown with red dashed lines in Fig. 3.10. The results presented in Fig. 3.10 show that the layer-to-layer stability bound is violated between the noise levels $\mu = 2.1p_e$ and $\mu = 2.3p_e$ and closer to $\mu = 2.3p_e$ which corresponds to $\mu^{exp} = 0.0057mm$. We use the approximate value of $\mu^{exp} \approx 2.25p_e$ for further discussions.

3.5.3.3 Outlook for control synthesis

Here, the results in Theorem 3.21 are compared to the experimental bound given above to validate the theoretical framework proposed in this chapter. For the discretization $\bar{\Lambda}$, $\bar{p}(k, \bar{\gamma}(\bar{\Lambda}))$ has a total of 400 deposition points per layer. By construction in the previous sections, we have $\rho(\boldsymbol{A}(\sigma(k), \boldsymbol{x}_k)) < 1$ (taken as 0.99 for analysis) and $\tilde{\omega} = ||\mathbf{0.05}||_2 = 1$. The spatial dynamics of the deposition process are only considered at the measurement points along the $\bar{p}(k, \gamma, \bar{\Lambda})$. It is straightforward to show the system is layer-to-layer geometrically stable as the spatial trajectories are building on top of one another to build a sound structure.

Suppose we want to synthesize a controller to ensure layer-to-layer ω-regular stability for the remainder of the process at layer $k_0 = 1, \zeta = 19$. For an initial error η_{k_0} upper bounded by $0.015mm$ ($i.e.$ $\boldsymbol{\eta_{k_0}} \preceq \mathbf{0.015}$), Theorem 3.21 is satisfied for noise levels $\mu \in (0, 0.0014mm]$ with a probability of 0.97 according to Corollary 3.23. In comparison to the experimental bound $\mu^{exp} = 0.0057mm$, the theoretical bound $\mu^{thr} = 0.0014mm$ is a conservative underapproximation of the actual robustness bound. This means that *a controller that stabilizes the AM process according to the theoretical bounds given in Theorem 3.21, albeit being conservative, would indeed layer-to-layer stabilize the AM process under disturbances (with the indicated probability).* This is an important finding that has not been previously presented in the literature and forms a basis for the theory of layer-to-layer stability for LLSV systems.

The gap between the noise levels for the predicted bound and the experimental bound is $0.0043mm$ which is below the measurement resolution in the experimental procedure. The discrepancy between the theoretical and experimental value is also due to the upper approximations in the formulation, which represent the worst-case effect of the noise on the system and the uncertainties in the formulation of the spatial register matrix and the resolution of the first-order approximation for the effect of input.

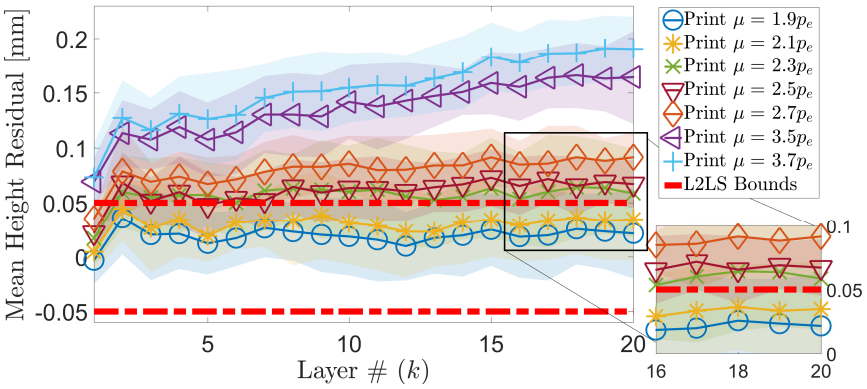


Figure 3.10: Plots of the mean height residuals and their one standard deviation for four different noise values. The layer-to-layer ω-regular stability bounds for the experiment are shown with red dashed lines.

3.5.3.4 Outlook for decision making

Here, we provide additional insights on how the theoretical findings in this chapter (i.e., Theorem 3.21) can be utilized as a decision-making mechanism for an autonomous FDM process. Since layer-to-layer ω-regular unstable parts will not conform to the design specifications, it may be desirable to stop the FDM process once we predict that the resulting part will become scrap. After the current layer k_0 is measured, we utilize a binary classifier based on Theorem 3.21 to predict if the process will be layer-to-layer ω-regular stable in ζ layers, in expectation. While the bound μ^{thr} is given for a projection of 19 layers, adjusting the horizon length ζ results in a more accurate prediction of the experimental robustness bounds. For a controller that stops the AM process as the prediction of layer-to-layer ω-regular instability (utilizing the LLSV model) is above a certain confidence level, adjusting the value of ζ would characterize the accuracy of the decision. To illustrate this concept, we present a receiver operating characteristic (ROC) curve for a binary classifier that predicts layer-to-layer ω-regular instability in ζ layers using Theorem 3.21.

Figure 3.11 shows the ROC curve for predicting if the process will be layer-to-layer ω-regular unstable, parametrized by ζ. We have the same analysis setup with $\boldsymbol{\eta}_1 \preceq \mathbf{0.015}$ and $\tilde{\omega} = ||\mathbf{0.05}||_2$. To evaluate the true positive and false positive rates, we utilize the experimental data with 13 noise levels, out of which 9 are layer-to-layer unstable. We utilize the classifier to predict layer-to-layer ω-regular instability and plot the results for the values of $\zeta \in [1, 19]$. We observe that a value of $\zeta = 5$ provides a true positive rate of 1 with a low false-positive rate (0.25). Similarly, a value of $\zeta = 4$ provides a false positive rate of 0 with a high true positive rate (0.89). Thus, a decision-maker should be operated at one of these levels based on the desired operating characteristics. This result also confirms the over-conservative nature of the theorem as high values of ζ cause the classifier to predict

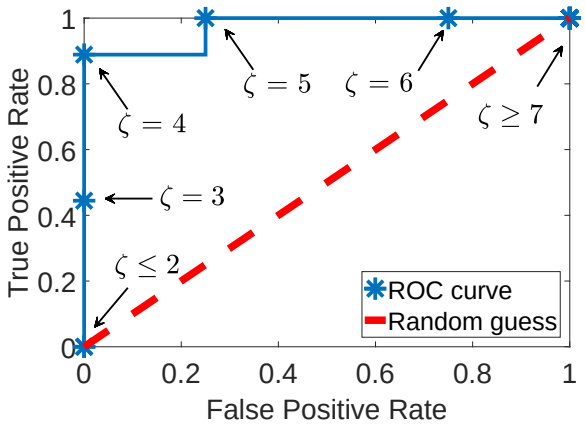


Figure 3.11: ROC curve for the decision-maker utilizing a classifier based on Theorem 3.21 to predict if the FDM process measured at layer $k = 1$ will be layer-to-layer ω-regular unstable. The curve is parametrized by the horizon $\zeta \in [1, 19]$ for the remaining layers in the process.

that the process will go unstable. A perfect process model would be able to completely identify layer-to-layer ω-regular stable processes from unstable ones. Based on the LLSV model derived for a specific process, the ROC curve can be experimentally constructed and implemented for a practical decision-maker. A closed-loop controller may utilize the ROC curve to tune the prediction horizon of a controller on the spatial dynamics to ensure L2L stability of the printed device for all layers, which in turn would provide guarantees on the part functionality.

The fact that theoretical developments are more accurate with shorter horizons can be explained by the accuracy of the first-order approximation around the operating point. Additionally, since the spatial register matrix used here is also an approximation, the accuracy of the projections deteriorate for long horizon lengths. By evaluating higher fidelity models of the AM processes, high accuracy decision-makers for longer prediction horizons (ζ) may be developed.

3.6 Chapter Conclusions

Linear layer-wise spatially varying (LLSV) models provide a general framework for modeling the spatial dynamics of AM processes where the layer-to-layer height evolution of an AM process is modeled over a discretization of interest. The main contribution for this chapter is the development of a modeling framework and a notion of layer-to-layer stability to characterize the performance of the layer-to-layer spatial AM processes (*C1*). Also, a theoretical bound for robustness to layer-to-layer stability is given and the provided bound is compared with experimental results. This theoretical framework provides a new

analysis tool for the performance of the layer-to-layer spatial dynamics of AM processes. For the experimental validation, the proposed modeling framework is demonstrated on an FDM machine and an LLSV model of the process is developed. The theoretical robustness bound for layer-to-layer stability gives a conservative approximation of the actual stability bound. Therefore, layer-to-layer stability and the robustness bounds proposed in this chapter may be leveraged for control development of LLSV systems. This chapter presents crucial findings on the stability of the spatial dynamics through the notion of layer-to-layer stability. Stronger stability conditions such as layer-to-layer finite stability can be used for designing controllers that guarantee monotonic stability of the spatial dynamics over the layer domain. Dimensional inaccuracy of FDM processes is a critical issue in practice. The experimental study in this chapter studies the effect of spatial disturbances to characterize layer-to-layer stability, which in turn characterizes the geometric performance of the printed part. Other important issues that are not considered with the L2L stability framework presented in this chapter include build bed adhesion issues and stepper motor inaccuracies. Addressing other such critical challenges for FDM processes are subjects for future research.

The unifying control-oriented spatial modeling framework developed in this chapter provides a foundation to develop closed-loop layer-to-layer control applications for AM processes. The next chapter utilizes the modeling framework developed in this chapter to develop a novel closed-loop controller for layer-to-layer dynamics. Additionally, the L2L stability concepts are analyzed in the context of stabilizability for an AM process to realize some of the applications outlined in this chapter in a closed-loop controlled AM process.

CHAPTER IV

Control for Spatial Processes

In this chapter, we leverage the layer-to-layer spatial dynamical model developed in Chapter III to develop closed-loop controllers for spatial AM processes. The main contribution of this chapter is the introduction of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints (*C2*). The chapter is divided into two main sections. The first section discusses a learning-based controller methodology to deal with model uncertainties in a real process. The presented method learns a control signal that minimizes the tracking error for a subsequent layer, given the measurements and the reference trajectory of the spatial process up to the current layer. The second section presents further analysis tools for layer-to-layer spatial dynamics, focusing on layer-to-layer stabilizability and finite stabilizability for a spatial process, by leveraging stability concepts presented in Chapter III. We formalize layer-to-layer stabilizing controllers for a layer-to-layer spatial process and present results on evaluating layer-to-layer stabilizing control policies for additive manufacturing processes. Each section has an illustrative case study at the end to demonstrate the presented concepts on simulation studies of spatial processes with examples given from fused deposition modeling processes, similar to those presented in Chapter III. The contents of the main sections in this chapter are in preparation for journal submissions.

4.1 Layer-to-layer Learning Control for Additive Manufacturing Spatial Dynamics

Additive manufacturing (AM) processes are increasingly utilized in a wide range of applications in research and industry. The increasing availability of new technologies and materials enables AM to be used in high-performance production settings where printed part

specifications are stringent. Consequently, there is an important need for high-performance AM processes with in-situ sensing, verification, and closed-loop control [20]. Closed-loop control of AM process dynamics is an important research area with many recent works [4, 82, 94, 149, 170]. While closed-loop control has been studied, methods to ensure the layer-to-layer performance of an AM process via closed-loop control remain an important research challenge [2, 13, 83].

Due to the physics involved in many AM processes, in-situ measurements during the formation of a layer are often infeasible or unfavorable. Instead, a layer-to-layer (L2L) control approach is often utilized where a control action is evaluated for the next layer after the current layer is formed and measured by an in-situ measurement system [4,90,170,201], e.g., the layer-to-layer heightmap measurements presented in Chapter III. While achieving stability under certain process conditions, existing controllers often rely on the layer-wise invariant nature of the process and the reference to provide their performance results [4,94]. Compensating for layer-wise variations in the spatial dynamics (including deposition path and layer height) is an important challenge that has not been addressed for closed-loop control of the L2L dynamics in the literature.

Iterative Learning Control (ILC) is a control strategy that enables enhanced reference tracking in repetitive processes, where a reference signal is tracked by a dynamical process in each iteration. Due to its wide application opportunities and its robustness to model uncertainty and disturbances, ILC has been extensively studied in the literature in both temporal [21, 133] and spatial domains [4, 41, 94]. ILC has been used extensively in AM applications to track an iteration invariant deposition trajectory for each layer [4, 90, 94, 170]. However, in practice, AM processes have multiple layers that have varying deposition paths [13, 83]. Classical ILC developments fail to provide tracking performance guarantees under such iteration-varying references. Additionally, the existing ILC literature for AM assumes uniform initial conditions for an AM process. However, as a current layer is the initial condition for a subsequent layer, a uniform initial condition assumption does not hold in an L2L dynamics setting. In this chapter, we propose an ILC method that utilizes the process reference information for the next layer and the in-situ measurement of the current layer to compute a control action, which provides provable convergence in the L2L domain under certain assumptions. Due to the utilization of the information for the next layer, we name our approach as *layer preview iterative learning control* (LP-ILC).

Although there are frameworks for iteration-varying references and initial conditions (e.g., [47, 189]), these solutions do not utilize the knowledge of an upcoming reference change to improve controller performance. As spatial dynamics and more importantly references are predetermined for AM processes, we utilize this knowledge to develop our

reference preview-based approach.

The main contribution of this chapter is the introduction of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints (*C2*). The specific contributions of this section focus on constrained reference tracking.

(*C2*-1) Formulation of a layer-to-layer iterative learning controller that utilizes an iteration preview approach to track spatial references for layer-wise varying AM processes.

(*C2*-2) Formal performance analysis and guarantees for the proposed LP-ILC controller.

(*C2*-3) A simulation study to illustrate the utility of the proposed controller.

4.1.1 Background

We consider the linear layer-wise spatially varying (LLSV) system dynamics for an AM process (see Chapter III) given by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k \tag{4.1}$$

where $k = 1, 2, \ldots, n_\ell - 1$ is the layer index, n_ℓ is the total number of layers, $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$, $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ are the spatial state vector and the spatial control input, respectively. $\boldsymbol{A}_k \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{B}_k \in \mathbb{R}^{n_x \times n_u}$ are state and input matrices for the spatial process at layer k. We assume that the system in (4.1) is controllable for all layers. Here, $\boldsymbol{A}_k$ represents the L2L spatial effects of a previous layer on a subsequent layer, and $\boldsymbol{B}_k$ represents the spatial effects of an input to the process at layer k. Refer to Chapter III and Appendix A [13, 16] for further details of process model (4.1).

In this chapter, similar to Chapter III, we take the state $\boldsymbol{x}_k$ as the height of the printed part *up to layer* k. The state is directly measured by an in-situ spatial measurement system (e.g., [2, 16, 170]) after a layer is formed (printed). Therefore, we measure the state directly and omit additional output dynamics in (4.1). Additionally, since the height in an AM process is nonnegative, nondecreasing, and increasing with the material input, (4.1) denotes a positive system where given an initially nonnegative state variable, the state is always nonnegative.

We define *layer dynamics groups* as consecutive layers where the input dynamics ($\boldsymbol{B}$) are layer-wise spatially invariant, i.e., dynamics of the layers within the same layer group (see Chapter III). Note that while the layer groups in Chapter III prescribes a corresponding desired input sequence for the group, here we only refer the the grouping of the dynamics as the proposed controller defines the input for each layer instead, hence the name layer

dynamics group. For completeness, we provide the definition in the following. Formally, we denote $k \in \Omega_i$, where Ω_i is the layer dynamics group such that $\boldsymbol{B}_k = \boldsymbol{B}_m$ for all pairs $(k, m) \in \Omega_i$ in the same layer dynamics group. Additionally, if we have a layer $k \in \Omega_i$ and the subsequent layer $k+1 \in \Omega_j$ (with $i \neq j$), we name the layer $k+1$ a *dynamics switch layer* as it is the layer where we switch from the layer dynamics group Ω_i to Ω_j. In this context, the layer $k+2$ is named a *switch layer* since a change in $\boldsymbol{B}_{k+1}$ affects the state $\boldsymbol{x}_{k+2}$. Within a layer dynamics group, we have $\boldsymbol{A}_k = \boldsymbol{A}_m$ for all pairs $(k, m) \in \Omega_i$ except for the switch layer. Layer dynamics groups are common in AM processes, especially for geometries with large number of layers with small layer increments (i.e., layer thickness).

4.1.2 Preliminaries and Layer-to-Layer Reference Tracking Problem Formulation

The iterative task for system (4.1) is for the state $\boldsymbol{x}_k$ to track a known reference $\boldsymbol{r}_k \in \mathbb{R}^{n_i}$ with minimal error. In the remainder of this section, we use boldface variables to denote specific inputs, states, etc., for the L2L spatial process, whereas normal font variables denote the arguments of functions. The cost function for the controller is given as

$$\mathcal{J}_k(u) = ||\boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{B}_k u - \boldsymbol{r}_{k+1}||^2_{\boldsymbol{Q}} + ||u||^2_{\boldsymbol{S}}, \tag{4.2}$$

where $u \in \mathbb{R}^{n_u}$ is the argument of the cost function, and $\boldsymbol{Q} = q\boldsymbol{I}, \boldsymbol{S} = s\boldsymbol{I}, (q, s) \in \mathbb{R}_+$, which defines a performance metric based on the tracking error and the control input. Our goal is to choose a control input $\boldsymbol{u}_k$, such that the cost function (4.2) is minimized for each layer.

The reference $\boldsymbol{r}_k$ is a design variable and reflects the reference geometry to be printed by the L2L spatial AM process. Additionally, since $\boldsymbol{r}_k$ denotes the overall reference height for the L2L process *up to layer* k, it is strictly increasing and varying at each layer. By solving a series of optimization problems at each layer, our goal is to iteratively learn the best control signal for the next layer ($\boldsymbol{u}_{k+1}$) given the current input ($\boldsymbol{u}_k$), output measurement ($\boldsymbol{x}_k$), and reference for the next layer ($\boldsymbol{r}_{k+1}$). In this section, the problem of tracking an iteration varying reference $\boldsymbol{r}_k$ is named as the *iteration-varying reference problem*.

Let $\phi(\boldsymbol{x}_{k'}, \{\boldsymbol{u}_k\}_{k'}^{k'+N})$ denote the solution of the state equation (4.1) for the initial state $\boldsymbol{x}_{k'}$ and the control inputs $\{\boldsymbol{u}_k\}_{k'}^{k'+N} = \{\boldsymbol{u}_{k'}, \boldsymbol{u}_{k'+1}, \ldots, \boldsymbol{u}_{k'+N}\}$. Next, we define the reachability of a target from a given current state under zero disturbances.

Definition 4.1. (N Layer Reachable State) A desired state $\boldsymbol{r} \in \mathbb{R}^{n_x}$ is reachable from $\boldsymbol{x}_{k'}$ in N layers if $\exists \boldsymbol{u}_k \in \mathcal{U}, k = k', \ldots, k'+N$ such that $\phi(\boldsymbol{x}_{k'}, \{\boldsymbol{u}_k\}_{k'}^{k'+N}) = \boldsymbol{r}$.

Note that in many AM applications the spatial locations of material addition (i.e., deposition pattern) vary between layers. Therefore, an error at a previous layer may result

in unreachable desired states for the L2L process due to the deposition pattern of future layers. However, in many applications the goal is to track a total reference spatial height profile by the end of the process. We define an additional reachability condition to characterize the spatial height tracking goal. Let ι_k denote the spatial locations with a change in desired spatial height at layer k, given as

$$\iota_k = \iota(\boldsymbol{r}_k - \boldsymbol{r}_{k-1}), \tag{4.3}$$

where we abuse the notation to denote $\iota(\cdot)$ as an indicator function with $\iota(p) = 0$ if $p = 0$ and $\iota(p) = 1$ otherwise. Note that ι denotes the height change of the spatial locations between layers, and it is not related to the definition of layer dynamics groups.

Definition 4.2. (N Layer Reachable Height) A spatial height $\boldsymbol{r}_m^h = \boldsymbol{r}_m \circ \iota_m$ is reachable from $\boldsymbol{x}_{k'}$ in N layers if $\exists \boldsymbol{u}_k \in \mathcal{U}, k = k', \ldots, k'+N$ such that $\phi(\boldsymbol{x}_{k'}, \{\boldsymbol{u}_k\}_{k'}^{k'+N}) \circ \iota_m = \boldsymbol{r}_m^h$ with $m = k' + N$.

We assume that the spatial height for the last layer, $\boldsymbol{r}_{n_\ell}^h$, is reachable from the initial state of the process in n_ℓ layers under the input constraint set $\mathcal{U}$. This assumption is necessary for a well-posed control problem and states that we have adequate input authority to track the iteration varying reference in the L2L domain by the end of the AM process.

Let us first consider a single layer and the corresponding minimization problem, given as,

$$\boldsymbol{u}_k^* = \arg\min_{u \in \mathcal{U}} \{\mathcal{J}_k(u) \mid \theta_k\}, \tag{4.4}$$

where θ_k denotes the information about the measurement of the current and past layers $(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots)$ as well as the reference for the next layer $(\boldsymbol{r}_{k+1})$. Here, we denote $\boldsymbol{u}_k^*$ as the minimizer of the minimization problem (4.4). Additionally, (4.4) includes a closed and convex input constraint set $\mathcal{U}$, which is necessary since we have finite nonnegative actuation for an AM process due to the physical considerations and possibly other constraints on the input signal. Thus, our approach is to solve the iteration-varying reference problem by evaluating the solution of the single layer minimization problem (4.4) at each layer. The approach of solving the iteration-varying reference problem by minimizing (4.4) at each layer has similarities with real-time iteration approaches (see [62] and references therein) since both approaches aim to estimate the best control action by solving a series of time-varying optimization problems at run-time. Real-time iteration approaches commonly approximate related optimization problems along a prediction horizon in run-time to solve a receding horizon control problem. The main goal of real-time iteration is to evaluate an approximate

control input between the measurement instances of the system, which are then updates as measurements become available. The proposed LP-ILC controller utilizes the spatial dynamics and the upcoming spatial reference for the next layer to learn a control action for the next layer by leveraging the system measurements. Future work may explore methods similar to real-time iteration for adapting the LP-ILC controller to applications where computation-time is limited, or it is advantageous to compute an approximate control input while the spatial measurements are not yet available to the controller.

Due to the physics involved in material addition during printing of a new layer, the input dynamics in $\boldsymbol{B}_k$ are often hard to capture exactly. Thus, we denote the true dynamics of the input as $\boldsymbol{B}_k = \bar{\boldsymbol{B}}_k(\boldsymbol{I} + \boldsymbol{W}_k\Delta_k^B)$ with $||\Delta_k^B|| \leq 1$ as an unstructured norm-bounded uncertainty (in a suitable norm topology, which is further discussed in the following subsection) on the nominal model $\bar{\boldsymbol{B}}_k$. Furthermore, we assume the uncertainty in the layer-to-layer dynamics is only on the input dynamics $\boldsymbol{B}_k$ and not the state matrix $\boldsymbol{A}_k$. This assumption is justified by applications in the literature where the state matrix often represents the geometrical relationship of the spatial height between layers, i.e., an integrator for the spatial height in the L2L domain [13, 98, 149].

Due to the model mismatch in the input dynamics as mentioned above, evaluating $\boldsymbol{u}_k^*$ for the minimization (4.4) is not always possible. Thus, the control goal for the L2L process is to compute an approximate solution of (4.4), which in turn is used as the input $\boldsymbol{u}_{k+1}$ for the next layer, given the information θ_k at the current layer. At each layer k, we utilize measurements of the system and the nominal model of the input dynamics $\bar{\boldsymbol{B}}_k$ to iteratively learn a control signal that minimizes the cost function (4.2) for the next layer $k+1$. In the next subsection we present the proposed controller and analyze its convergence properties.

4.1.3 Proposed LP-ILC Formulation

As stated earlier, we are interested in successively minimizing (4.2) for each layer in the LLSV process (4.1). Since we leverage the information for a subsequent layer when evaluating our proposed controller, we name the following ILC law as the layer-preview ILC (LP-ILC) update.

$$\boldsymbol{u}_{k+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}_{k+1}}(\boldsymbol{u}_k - \alpha \boldsymbol{P}_{k+1}^{-1}\nabla \mathcal{J}_{k+1}(\boldsymbol{u}_k)), \tag{4.5}$$

where $\alpha > 0$ is a step-size that we specify later based on the process model and the preconditioner $\boldsymbol{P}_{k+1} = \bar{\boldsymbol{B}}_{k+1}^T\boldsymbol{Q}\bar{\boldsymbol{B}}_{k+1} + \boldsymbol{S}$ is evaluated using the model $\bar{\boldsymbol{B}}_k$. We denote the

projection onto the input space

$$\Pi_{\mathcal{U}}^{\boldsymbol{P}}(z) = \underset{z' \in \mathcal{U}}{\operatorname{argmin}} \, ||z' - z||_{\boldsymbol{P}}, \tag{4.6}$$

by the projection operator to the convex set $\mathcal{U}$, with $0 \in \mathcal{U}$, in the norm defined by a positive definite matrix $\boldsymbol{P}$. We assume that the preconditioner $\boldsymbol{P}_k$ is symmetric and positive definite (PD) for all k. By utilizing the model $\bar{\boldsymbol{B}}$, we approximate the gradient of the cost function (4.2)

$$\nabla \mathcal{J}_k(u) = \bar{\boldsymbol{B}}_k^T \boldsymbol{Q}(\boldsymbol{B}_k u - \boldsymbol{\eta}_k) + \boldsymbol{S}u, \tag{4.7}$$

where $u \in \mathbb{R}^{n_u}$ is the argument of the gradient function, $\boldsymbol{\eta}_k = \boldsymbol{r}_{k+1} - \boldsymbol{A}_k \boldsymbol{x}_k$ and we replace $\boldsymbol{B}_k^T$ with $\bar{\boldsymbol{B}}_k^T$ since the true model $\boldsymbol{B}_k$ is not available. The LP-ILC update (4.5) utilizes the gradient function (4.7) evaluated at the previous control input $\boldsymbol{u}_k$ (i.e., $\boldsymbol{u}_k$ in place of the argument u in (4.7)). Note that whenever layers k and $k+1$ belong to the same layer dynamics group, we have that $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k$. Consequently, we incorporate the measurements of the most recent layer in place of $\boldsymbol{B}_{k+1}\boldsymbol{u}_k$ term for evaluating the gradient at $\boldsymbol{u}_k$ in (4.7) whenever the previous layer and the current layer are in the same layer dynamics group. Since we do not have measurement information available when switching between layer dynamics groups, we utilize the model $\bar{\boldsymbol{B}}_k$ for the dynamics switch layers. Suppose we have $k \in \Omega_i$, then to approximate the value of the gradient (4.7) at the point $\boldsymbol{u}_k$ we use the following.

$$\nabla \mathcal{J}_{k+1}(\boldsymbol{u}_k) = \begin{cases} \bar{\boldsymbol{B}}_{k+1}^T \boldsymbol{Q}(\boldsymbol{\xi}_{k+1} - \boldsymbol{\eta}_{k+1}) + \boldsymbol{S}\boldsymbol{u}_k & \text{if } k+1 \in \Omega_i \\ \bar{\boldsymbol{B}}_{k+1}^T \boldsymbol{Q}(\bar{\boldsymbol{B}}_{k+1}\boldsymbol{u}_k - \boldsymbol{\eta}_{k+1}) + \boldsymbol{S}\boldsymbol{u}_k & \text{otherwise,} \end{cases} \tag{4.8}$$

where $\boldsymbol{\xi}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{A}_k \boldsymbol{x}_k = \boldsymbol{B}_k \boldsymbol{u}_k$ from the past measurements is used in place of $\boldsymbol{B}_{k+1}\boldsymbol{u}_k$, since $\boldsymbol{B}_k = \boldsymbol{B}_{k+1}$ within the layer dynamics group. We name the first expression in (4.8) as the *data gradient* and the second one as the *model gradient* in further discussions. The gradient evaluation (4.8) is used with (4.5) to evaluate the control input for the next layer. For an L2L spatial process with the LLSV dynamics (4.1), we assume the system matrices are available to the controller at all times. Availability of all system matrices is a practical assumption when the spatial deposition paths of all layers are designed prior to the printing process and the models of the spatial input dynamics are known for the process, which is often the case for most AM processes. We provide the following procedure for the implementation of the LP-ILC for an L2L AM process.

Note that it is possible to initialize the initial layer with a choice better than just a feasi-

Algorithm 1 Proposed LP-ILC loop.

Initialize: $k = 0, \boldsymbol{x}_0 = \boldsymbol{0}, \boldsymbol{u}_0 \in \mathcal{U}$
Print the initial layer with a feasible choice of $\boldsymbol{u}_0$.
while $k \neq n_\ell - 1$ (The last layer is not printed.) **do**
    $k \leftarrow k + 1$
    Measure the spatial state $\boldsymbol{x}_k$.
    Evaluate the spatial input $\boldsymbol{u}_k$ via (4.5) and (4.8).
    Print the layer with the input $\boldsymbol{u}_k$.
end while
Measure the last layer $\boldsymbol{x}_{n_\ell}$.

ble input sequence (e.g., input design based on past data or numerical analysis). Additional design methods and models may be utilized to improve the initial input sequence. Here, we do not make such distinctions and consider any initial control input that is feasible with respect to the input set $\mathcal{U}$.

4.1.4 Performance Analysis

In this section we study the convergence properties of (4.5) under the approximation rule (4.8). We start by stating initial results for the convergence properties within a single layer dynamics group using the data gradient and at switch layers with the model gradient.

We denote the fixed point of (4.5) with respect to the iteration (layer) index k as $\hat{\boldsymbol{u}}_k$. The fixed point is evaluated by "freezing" the matrices $\boldsymbol{A}, \boldsymbol{B}, \bar{\boldsymbol{B}}, \boldsymbol{P}$ as well as $\boldsymbol{\eta}$ for the given layer index and iterating the resulting controller update. Let us show this for evaluating the fixed point of (4.5) for the layer index $k+1$. In the following, we fix the matrices $\boldsymbol{A}_{k+1}, \boldsymbol{B}_{k+1}, \bar{\boldsymbol{B}}_{k+1}, \boldsymbol{P}_{k+1}$ and also the vector $\boldsymbol{\eta}_{k+1}$. Then we use the subscript j to denote the iterations of the update equation (4.5) with these fixed vectors and matrices.

$$\boldsymbol{u}_{j+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}_{k+1}}(\boldsymbol{u}_j - \alpha \boldsymbol{P}_{k+1}^{-1} \nabla \mathcal{J}_{k+1}(\boldsymbol{u}_j)), \tag{4.9}$$

where now the fixed point is over the *iteration* index j and the *layer* index $k+1$ with the corresponding data and matrices are fixed. Note that realizing the iteration (4.9) on a practical system would require printing the exact same layer with the exact same dynamical matrices and $\boldsymbol{\eta}_{k+1}$, which may not be practical. The iteration (4.9) is therefore a theoretical setup to analyze the fixed point of the controller update for a given layer index, if the same layer is iterated until convergence to an optimal input, while the dynamic matrices and $\boldsymbol{\eta}$ are kept constant for each iteration. Based on the iteration (4.9), whenever we have $\varphi = ||\boldsymbol{I} - \alpha \boldsymbol{P}_{k+1}^{-1}(\bar{\boldsymbol{B}}_{k+1}^T \boldsymbol{Q} \boldsymbol{B}_{k+1} + \boldsymbol{S})||_{\boldsymbol{P}_{k+1}} < 1$, we get the following monotonic convergence

condition

$$||\boldsymbol{u}_{j+1} - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}_{k+1}} \leq \varphi||\boldsymbol{u}_j - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}_{k+1}}, \tag{4.10}$$

towards the fixed point of (4.9), i.e., $\hat{\boldsymbol{u}}_{k+1}$. Notice that the fixed points $\hat{\boldsymbol{u}}_k$ for different layers are not necessarily identical for all k due to the changing spatial dynamics and height-to-go $\boldsymbol{\eta}_k$ between subsequent layers (also within a single layer group), thus next, we characterize the bound on the control input within a single layer dynamics group.

Proposition 4.3. *Within a single layer dynamics group, let* $\varphi = ||\boldsymbol{I} - \alpha\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B} + \boldsymbol{S})||_{\boldsymbol{P}}$ *and* $\gamma = ||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}W||_{\boldsymbol{P}}$*, where we drop the layer subscripts since the dynamics are identical within a layer dynamics group. If* $\gamma < 1$ *and* $\alpha \in (0, \frac{2}{1+\gamma})$*, then we have*

$$||\boldsymbol{u}_{k+1} - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}} \leq \varphi^{k+1}||\boldsymbol{u}_0 - \hat{\boldsymbol{u}}_0||_{\boldsymbol{P}} + \zeta\sum_{i=0}^{k}\varphi^{k-i+1}||\boldsymbol{\eta}_{i+1} - \boldsymbol{\eta}_i||_{\boldsymbol{P}},$$

with $\varphi < 1, \zeta = ||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}||_{\boldsymbol{P}}/(1-\varphi)$*, and* $\hat{\boldsymbol{u}}_k$ *is the fixed point of (4.5).*

Proof. Let $\Phi = \boldsymbol{I} - \alpha\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B} + \boldsymbol{S})$, which is constant within a layer dynamics group. Then we can rewrite (4.5) as

$$\boldsymbol{u}_{k+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\Phi\boldsymbol{u}_k + P^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{\eta}_k), \tag{4.11}$$

with the fixed point $\hat{\boldsymbol{u}}_{k+1}$. Furthermore we have that

$$\begin{aligned}
||\boldsymbol{u}_{k+1} - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}} &\leq ||\Pi_{\mathcal{U}}^{\boldsymbol{P}}(\Phi\boldsymbol{u}_k + \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{\eta}_k) - \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\Phi\hat{\boldsymbol{u}}_{k+1} + \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{\eta}_k)||_{\boldsymbol{P}} \\
&\leq \varphi||\boldsymbol{u}_k - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}} \\
&\leq \varphi(||\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}} + ||\hat{\boldsymbol{u}}_k - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}}),
\end{aligned} \tag{4.12}$$

where we used the 1-Lipschitz property of the orthogonal projection in the positive definite preconditioner weighted norm and used the shorthand φ with the triangle inequality. Additionally, we can rewrite

$$\begin{aligned}
||\Phi||_{\boldsymbol{P}} &= ||\boldsymbol{P}^{-1}((1-\alpha)(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}} + \boldsymbol{S}) - \alpha\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}\Delta^B)||_{\boldsymbol{P}}, \\
&= ||(1-\alpha)\boldsymbol{I} - \alpha\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}\Delta^B)||_{\boldsymbol{P}}, \\
&\leq |1-\alpha| + \alpha||\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}\Delta^B)||_{\boldsymbol{P}},
\end{aligned} \tag{4.13}$$

where we have the induced weighted matrix norm $||\mathcal{A}||_{\boldsymbol{P}} = ||\boldsymbol{P}^{1/2}\mathcal{A}\boldsymbol{P}^{-1/2}||_2$ for a square matrix $\mathcal{A}$, and we used the definition $\boldsymbol{P} = \bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}} + \boldsymbol{S}$ to get the last inequality. From

(4.13), we conclude that $\varphi < 1$ is satisfied whenever $||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}\Delta^B||_{\boldsymbol{P}} < 1$ and $\alpha \in (0, \frac{2}{1+\gamma})$ by inspecting the two cases for the absolute value $|1-\alpha|$. Since $||\Delta^B||_{\boldsymbol{P}} < 1$, we have $||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}||_{\boldsymbol{P}} < 1$ as a sufficient condition. The last term in (4.12) can be bounded by analyzing the distance between two consecutive fixed points within a layer dynamics group, given by

$$\begin{aligned}||\hat{\boldsymbol{u}}_{k+1} - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}} &\leq ||\Pi_{\mathcal{U}}^{\boldsymbol{P}}(\Phi\hat{\boldsymbol{u}}_{k+1} + \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{\eta}_{k+1}) - \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\Phi\hat{\boldsymbol{u}}_k + \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{\eta}_k)||_{\boldsymbol{P}} \\ &\leq \varphi||\hat{\boldsymbol{u}}_{k+1} - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}} + ||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}||_{\boldsymbol{P}}||\boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k||_{\boldsymbol{P}} \\ &\leq \zeta||\boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k||_{\boldsymbol{P}},\end{aligned}$$

where we used the 1-Lipschitz property of the orthogonal projection and $\zeta = ||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}||_{\boldsymbol{P}}/(1-\varphi)$. By propagating the derived bounds for $k+1$ layers within a layer dynamics group, we get the presented result. □

We take $\alpha = 1$ with $||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\bar{\boldsymbol{B}}\boldsymbol{W}||_{\boldsymbol{P}} < 1$ for the rest of the section and omit α from the formulations to lighten the notation unless stated otherwise. Note that Proposition 4.3 implies the variation of fixed points $||\hat{\boldsymbol{u}}_k - \hat{\boldsymbol{u}}_{k-1}||$ is bounded by the change of $\boldsymbol{\eta}_k$ between layers. We interpret $\boldsymbol{\eta}_k$ as the *height-to-go* for the controller in the upcoming layer. An immediate implication of Proposition 4.3 is that in a single layer dynamics group with identically fixed reference height increments, the control input reaches its fixed point in the L2L domain whenever $\boldsymbol{\eta}_k = \boldsymbol{\eta}_{k+1}$. Note that in practical applications, we consider the fixed point convergence at a desired resolution, i.e., $||\boldsymbol{\eta}_k - \boldsymbol{\eta}_{k+1}|| < \epsilon$, with small ϵ.

Next, we derive bounds on the input when the model gradient is used with the switch layers. Whenever the model gradient is used with the preconditioner $\boldsymbol{P}_{k+1}$, observe that we get (4.5) as

$$\bar{\boldsymbol{u}}_{k+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}_{k+1}}(\boldsymbol{P}_{k+1}^{-1}\bar{\boldsymbol{B}}_{k+1}^T\boldsymbol{Q}\boldsymbol{\eta}_{k+1}), \tag{4.14}$$

where we use $\bar{\boldsymbol{u}}_{k+1}$ to indicate the control input evaluated by utilizing the model gradient.

Proposition 4.4. *The LP-ILC iteration (4.5) using the model gradient in (4.8) has the bound* $||\bar{\boldsymbol{u}}_k - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}} \leq \varphi||\hat{\boldsymbol{u}}_k||_{\boldsymbol{P}}$*, where* $\varphi = ||\boldsymbol{I} - \alpha\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B} + \boldsymbol{S})||_{\boldsymbol{P}}$.

Proof. Follows immediately by combining the expression for the fixed point $\hat{\boldsymbol{u}}_k$ given in the proof of Proposition 4.3, and (4.14). □

Therefore, the bounds on $||\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}}$, i.e., the distance between the input evaluated by (4.5) under the rule (4.8) and the fixed point of (4.9) can be evaluated utilizing the presented propositions. For a given layer, an upper bound on $||\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}}$ is evaluated by combining

the bounds in Proposition 4.3 and Proposition 4.4 for the iterations where we use the data gradient and model gradient in (4.8), respectively.

Next, we study the error progression of the proposed controller in the L2L domain. Here we adopt a standard assumption for the reference trajectory to have the form $\boldsymbol{r}_{k+1} = \boldsymbol{A}_k\boldsymbol{r}_k + \boldsymbol{B}_k\boldsymbol{u}_k^r$, where the reference input is denoted as $\boldsymbol{u}_k^r \in \mathcal{U}$ and is unknown to the controller due to the model mismatch. This assumption simply states that the reference trajectory to be tracked by the L2L spatial dynamics can be represented as a trajectory generated by the process dynamics. Thereby, we ensure the feasibility of the reference tracking problem. As an example, a 3D printing process for a simple shell geometry with identical layer thickness may be represented as the following. The spatial reference trajectory for each layer is represented by $\boldsymbol{r}$, we use $\boldsymbol{A}_k = \boldsymbol{I}$ to represent the increasing height of the reference at each layer, and $\boldsymbol{B}_k\boldsymbol{u}^r$ as the uniform layer thickness to be printed at each layer in the spatial domain. Additionally, we denote the error as $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{r}_k$. The error dynamics are thus given as

$$\boldsymbol{e}_{k+1} = \boldsymbol{A}_k\boldsymbol{e}_k + \boldsymbol{B}_k(\boldsymbol{u}_k - \boldsymbol{u}_k^r). \tag{4.15}$$

Note that $\boldsymbol{u}_k^r$ is the optimal input if we have zero error, i.e., $\boldsymbol{x}_k = \boldsymbol{r}_k$.

We start by demonstrating the error tracking performance of the proposed controller for the case of no model mismatch, i.e., $\bar{\boldsymbol{B}}_k = \boldsymbol{B}_k$. Observe that in this case we get $\varphi = ||\boldsymbol{I} - \alpha\boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B} + \boldsymbol{S})|| = 0$ with $\alpha = 1$, since we have $\boldsymbol{P}_{k+1} = \bar{\boldsymbol{B}}_{k+1}^T\boldsymbol{Q}\bar{\boldsymbol{B}}_{k+1} + \boldsymbol{S}$. Then, it is clear that $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ by Proposition 4.3. Furthermore, by Proposition 4.4, we see that $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ holds for switch layers as well. The formulation of the input $\boldsymbol{u}_k$ in this setting is given similar to (4.14), by taking $\bar{\boldsymbol{B}}_k = \boldsymbol{B}_k$, which corresponds to the model gradient in the case of a perfect model, i.e., the case where the model and data gradients are identical. Note that in this case the fixed points $\hat{\boldsymbol{u}}_k$ correspond to the projections of optimal inputs that minimize the tracking error for each layer. Thus, the inputs $\boldsymbol{u}_k = \hat{\boldsymbol{u}}_k$ track the L2L reference bounded by an error ball that is characterized by the regularization term $\boldsymbol{S}$ for the case with no model mismatch, given that tracking control is feasible. Due to the reachable height assumption, we assume that the L2L input $\boldsymbol{u}_k$ can track the L2L reference at a layer $k(e_0)$ parametrized by the initial tracking error e_0. We focus on the case with $\boldsymbol{S} = 0$ to illustrate the case of perfect tracking with the proposed controller when we have a perfect model and strictly feasible control input, in the following.

Theorem 4.5. *(Perfect tracking) Suppose we have a perfect model, i.e., $\bar{\boldsymbol{B}}_k = \boldsymbol{B}_k$ and $\boldsymbol{u}^r \in \mathcal{U}$. Additionally, suppose there exists a layer $k(e_0)$ such that the input update strictly feasible (i.e., $\left(\boldsymbol{u}_k - \alpha\boldsymbol{P}_{k+1}^{-1}\nabla\mathcal{J}_{k+1}(\boldsymbol{u}_k)\right) \in \mathcal{U}$) for all $k > k(e_0)$, then within a layer dy-*

namics group we have

$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} - \boldsymbol{B}\boldsymbol{P}^{-1}\boldsymbol{B}^T\boldsymbol{Q})(\boldsymbol{A}\boldsymbol{e}_k - \boldsymbol{B}\boldsymbol{u}^r), \tag{4.16}$$

which is asymptotically convergent if $\rho((\boldsymbol{I} - \boldsymbol{B}\boldsymbol{P}^{-1}\boldsymbol{B}^T\boldsymbol{Q})\boldsymbol{A}) < 1$. *Furthermore, if we have* $\boldsymbol{S} = 0$*, then we get* $\limsup_{k\to\infty} ||\boldsymbol{e}_k|| = 0$.

Proof. As previously mentioned, with no model mismatch, we get $\varphi = 0$ with $\alpha = 1$. The input with the perfect model is evaluated by $\boldsymbol{u}_k = \Pi_{\mathcal{U}}^{\boldsymbol{P}_k}(\boldsymbol{v}_k)$, where we have the shorthand $\boldsymbol{v}_k = \boldsymbol{P}_k^{-1}\boldsymbol{B}_k^T\boldsymbol{Q}\boldsymbol{\eta}_k$. The strict feasibility assumption implies that $\boldsymbol{u}_k = \boldsymbol{v}_k$. Then, by noting that $\boldsymbol{\eta}_k = \boldsymbol{B}_k\boldsymbol{u}^r - \boldsymbol{A}_k\boldsymbol{e}_k$, we get (4.16) by using (4.15). Therefore if $\rho((\boldsymbol{I} - \boldsymbol{B}\boldsymbol{P}^{-1}\boldsymbol{B}^T\boldsymbol{Q})\boldsymbol{A}) < 1$ within a layer dynamics group, we get asymptotic convergence to a fixed point. Notice that, the fixed point may not necessarily be zero in this case. Note that with the perfect model, it can be shown that $\boldsymbol{B}_k\boldsymbol{P}_k^{-1}\boldsymbol{B}_k^T\boldsymbol{Q} \preceq \boldsymbol{I}$ with the equality holding if $\boldsymbol{S} = 0$. Therefore, whenever $\boldsymbol{S} = 0$, we have $\boldsymbol{B}_k\boldsymbol{P}_k^{-1}\boldsymbol{B}_k^T\boldsymbol{Q} = \boldsymbol{I}$ since $\boldsymbol{P}_k = \bar{\boldsymbol{B}}_k^T\boldsymbol{Q}\bar{\boldsymbol{B}}_k + \boldsymbol{S}$, thus we get $\limsup_{k\to\infty} ||\boldsymbol{e}_k|| = 0$. □

Note that additionally if we have $||(\boldsymbol{I} - \boldsymbol{B}_k\boldsymbol{P}_k^{-1}\boldsymbol{B}_k^T\boldsymbol{Q})\boldsymbol{A}_k||_{\boldsymbol{P}} < 1$, we get monotonic convergence in the weighted $\boldsymbol{P}$ norm. The results of Theorem 4.5 imply that an ideal controller (in the sense of zero tracking error) is one that tracks the reference at some layer $k(e_0)$ and then uses $\boldsymbol{u}^r$ for all future layers $k > k(e_0)$.

Next, we analyze a special case of the proposed controller that corresponds to a common model assumption used in the literature. To represent the L2L height evolution, discrete-time integrator dynamics are often utilized for the matrix $\boldsymbol{A}_k$, resulting in $\boldsymbol{A}_k = \boldsymbol{I}$. Notice that in this form (i.e., $\boldsymbol{A}_k = \boldsymbol{I}$) an optimal input $\boldsymbol{u}^r$ for the layer dynamics group results in fixed incremental layer height, thus this model choice represents a practical AM process. Additionally, note that in this setting, $\boldsymbol{e}_k$ can reach an equilibrium only if we have $\boldsymbol{u}_k = \boldsymbol{u}^r$. We assume that $\boldsymbol{u}^r \in \mathcal{U}$ as previously mentioned, as the problem is ill posed otherwise.

Integrator dynamics are well-known to be not asymptotically stable since $\rho(\boldsymbol{A}) = 1$ in this case. We first provide results on the boundedness under the assumption of bounded distance between the inputs and $\boldsymbol{u}_k^r$. Let us denote $\sup_{\boldsymbol{u}_k\in\mathcal{U}} ||\boldsymbol{u}_k - \boldsymbol{u}^r||_P \leq \sigma$. Due to the constraints on the input, we can always derive a bound for σ albeit conservatively in the case of high model uncertainty. The following proposition specializes the results of Proposition 4.3 in this setting.

Proposition 4.6. *Suppose that within a layer dynamics group, $\boldsymbol{A}_k = \boldsymbol{I}$. Then,*

$$||\boldsymbol{u}_{k+1} - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}} \leq \varphi^{k+1}||\boldsymbol{u}_0 - \hat{\boldsymbol{u}}_0||_{\boldsymbol{P}} + c_\sigma,$$

where $\varphi < 1$ and $c_\sigma = \frac{\sigma||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B}||_{\boldsymbol{P}}}{(1-\varphi)} \sum_{i=0}^{k} \varphi^{k-i+1}$.

Proof. The error dynamics are given as $\boldsymbol{e}_{k+1} = \boldsymbol{e}_k + \boldsymbol{B}_k(\boldsymbol{u}_k - \boldsymbol{u}_k^r)$. Then we have that $\boldsymbol{\eta}_{k+1} - \boldsymbol{\eta}_k = B(\boldsymbol{u}^r - \boldsymbol{u}_k)$. By following the proof of Proposition 4.3 we get the bound

$$||\boldsymbol{u}_{k+1} - \hat{\boldsymbol{u}}_{k+1}||_{\boldsymbol{P}} \leq \varphi(||\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k||_{\boldsymbol{P}} + \frac{||\boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{B}||_{\boldsymbol{P}}}{(1-\varphi)}||\boldsymbol{u}_k - \boldsymbol{u}^r||_{\boldsymbol{P}}).$$

Then, by using the bound σ and taking the sum of over the layers, we get the stated result. □

Next, we provide further remarks on the effect of input constraints on the error dynamics within a layer dynamics group. Suppose we have a single layer dynamics group with constant $\boldsymbol{u}^r := \boldsymbol{u}_k^r$ and the control input computed as $\boldsymbol{v}_{k+1} = \boldsymbol{u}_k - \alpha\boldsymbol{P}_{k+1}^{-1}\nabla\mathcal{J}_{k+1}(\boldsymbol{u}_k)$ is outside of the set $\mathcal{U}$, i.e., $\boldsymbol{v}_{k+1} \notin \mathcal{U}$. As discussed earlier, a constant reference input is possible only if $\boldsymbol{A}_k = \boldsymbol{I}$. Additionally, suppose $\mathcal{U} = [0, u^{\max}]^{n_u}$, i.e., a box constraint. Then, due to the projection, we get the control input $\boldsymbol{u}_{k+1}$ that is closest to $\boldsymbol{v}_{k+1}$ in the preconditioner norm. Additionally, since the references are reachable and the system is positive, we must have a reference $\boldsymbol{r}_{k'}$ at layer k' where we evaluate $\boldsymbol{u}_k = \boldsymbol{v}_k$, i.e., $\boldsymbol{v}_k \in \mathcal{U}$ in this setting, which corresponds to the case with the strict feasibility assumption mentioned previously. Then for a single layer dynamics group with constant $\boldsymbol{u}^r$, the dynamics of $\boldsymbol{w}_k := \boldsymbol{u}_k - \boldsymbol{u}^r$ results in the affine closed-loop dynamics of the input

$$\boldsymbol{w}_{k+1} = \boldsymbol{K}\boldsymbol{w}_k - \boldsymbol{E}\boldsymbol{e}_k - \boldsymbol{P}^{-1}\boldsymbol{S}\boldsymbol{u}^r, \tag{4.17}$$

where we use (4.15) to derive $\boldsymbol{K} = \boldsymbol{I} - \boldsymbol{P}^{-1}(\bar{\boldsymbol{B}}^T\boldsymbol{Q}(\boldsymbol{B} + \boldsymbol{A}\boldsymbol{B}) + \boldsymbol{S})$ and $\boldsymbol{E} = \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}\boldsymbol{A}^2$. Whenever $\boldsymbol{v}_k \notin \mathcal{U}$, we can bound the norm $||\boldsymbol{w}_k||$ by the right hand side of (4.17) using the properties of the projection operator. Box constraints to limit the nonnegative inputs to a feasible physical input capacity is a common choice of input constraint set.

To characterize the closed-loop system behavior, we next formulate the combined dynamics of the LP-ILC control update with the L2L spatial dynamical process. Utilizing (4.15) and (4.17) we denote the combined dynamics of the error and input within a layer dynamics group with constant $\boldsymbol{u}^r$ in the lifted variable $\boldsymbol{z}_k := [\boldsymbol{u}_k^T, \boldsymbol{e}_k^T]^T$. Suppose, again,

that we have $\boldsymbol{v}_k \in \mathcal{U}$, then we get

$$\boldsymbol{z}_{k+1} = \underbrace{\begin{bmatrix} \boldsymbol{K} & -\boldsymbol{E} \\ \boldsymbol{B} & \boldsymbol{A} \end{bmatrix}}_{\boldsymbol{Z}} \boldsymbol{z}_k - \begin{bmatrix} \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}(\boldsymbol{B}+\boldsymbol{A}\boldsymbol{B})\boldsymbol{u}^r \\ -\boldsymbol{B}\boldsymbol{u}^r \end{bmatrix}. \tag{4.18}$$

Therefore, the controller (4.5) with model mismatch is convergent in the L2L domain within a layer dynamics group if the spectral radius of $\boldsymbol{Z}$ is less than 1 (under the assumption of strict feasibility for layers $k > k(e_0)$). When the dynamics are known, we use (4.18) to tune the controller gains while ensuring convergence. Note that if we have the true dynamics and $\boldsymbol{S} = 0$, we recover the nominal convergence given in Theorem 4.5. Additionally, under the box constraints, we can rewrite (4.18) in terms of a saturated linear feedback. Then, utilizing equivalent representations given in [52], it is possible to characterize the region of asymptotic stability (RAS) for the saturated system. Since the RAS in this case is given for the saturated closed loop system, the assumption on the strict feasibility $k > k(e_0)$ is not required. Note that for high dimensional processes such as the L2L spatial AM process in discussion, computation and verification of an RAS may be computationally infeasible as the formulations in the literature for verifying the RAS for a set of initial conditions scale exponentially in the dimensionality of the problem (e.g., [52, 70]). We assume that the set $\mathcal{U}$ is within the RAS of the system with the linear region dynamics (4.18) (with respect to the saturation due to the projection). Under this assumption, since the system has an asymptotically stable equilibrium, the condition of feasibility for layers $k > k(e_0)$ holds whenever $\boldsymbol{z}_k$ is near its equilibrium, for some $k(e_0)$. Furthermore, stability of the saturated dynamics for (4.18) may be verified utilizing diagonal Lyapunov functions by extending the methods presented in e.g., [70, 96], and [122] Chapter 3. Note that while $\boldsymbol{u}_k = \boldsymbol{u}^r$ is always an equilibrium for $\boldsymbol{A}_k = \boldsymbol{I}$, the equilibrium of the error depends on the magnitude of $\boldsymbol{S}$.

A Projected Linear Dynamical System Representation

The closed loop dynamics for strictly feasible input sequences (i.e., without projection) given in (4.18) is in a linear affine form. We can rewrite the closed loop dynamics of (4.18) within a single layer group in terms of a partially projected linear system. Let $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{u}}^T, \tilde{\boldsymbol{e}}^T]^T$ denote the fixed point of (4.18) with the projection operator on the input dynamics. Note that $\tilde{\boldsymbol{u}} = \boldsymbol{u}^r$ within a single layer dynamics group. Additionally, suppose the input constraint set is polyhedral, i.e., $\mathcal{U} = \{u \in \mathbb{R}^{n_u} | Hu \leq b\}$, for some matrix H and

vector b of appropriate sizes. Then the projected input dynamics can be written as

$$\boldsymbol{u}_{k+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{c}_z), \tag{4.19}$$

where $\boldsymbol{F} = [\boldsymbol{K}, -\boldsymbol{E}]$, and $\boldsymbol{c}_z = \boldsymbol{P}^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{Q}(\boldsymbol{B} + \boldsymbol{A}\boldsymbol{B})\boldsymbol{u}^r$. Thus, at the fixed point we have $\tilde{\boldsymbol{u}} = \boldsymbol{u}^r = \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\boldsymbol{F}\tilde{\boldsymbol{z}} + \boldsymbol{c}_z)$, and since we have $\boldsymbol{u}^r \in \mathcal{U}$ by assumption, we get

$$\boldsymbol{u}^r = \boldsymbol{F}\tilde{\boldsymbol{z}} + \boldsymbol{c}_z. \tag{4.20}$$

Then by utilizing (4.19) and (4.20) we can write

$$\begin{aligned}
\boldsymbol{u}_{k+1} - \boldsymbol{u}^r &= \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{c}_z) - \boldsymbol{u}^r && (4.21)\\
&= \Pi_{\mathcal{U}}^{\boldsymbol{P}}(\boldsymbol{F}(\boldsymbol{z}_k - \tilde{\boldsymbol{z}}) + \boldsymbol{F}\tilde{\boldsymbol{z}} + \boldsymbol{c}_z) - \boldsymbol{u}^r && (4.22)\\
&= \Pi_{\tilde{\mathcal{U}}}^{\boldsymbol{P}}(\boldsymbol{F}(\boldsymbol{z}_k - \tilde{\boldsymbol{z}})) + \boldsymbol{F}\tilde{\boldsymbol{z}} + \boldsymbol{c}_z - \boldsymbol{u}^r && (4.23)\\
&= \Pi_{\tilde{\mathcal{U}}}^{\boldsymbol{P}}(\boldsymbol{F}(\boldsymbol{z}_k - \tilde{\boldsymbol{z}})), && (4.24)
\end{aligned}$$

where we have $\tilde{\mathcal{U}} = \{u \in \mathbb{R}^{n_u} | Hu \leq b - H\boldsymbol{u}^r\}$ as the modified constraint set, and we use (4.20) in the last step. In (4.23) we use the equivalence

$$\Pi_{\mathcal{U}}^{\boldsymbol{P}}(\boldsymbol{v} + \boldsymbol{u}^r) = \Pi_{\tilde{\mathcal{U}}}^{\boldsymbol{P}}(\boldsymbol{v}) + \boldsymbol{u}^r, \tag{4.25}$$

for any $\boldsymbol{v} \in \mathbb{R}^{n_u}$, which follows from the definition of the projection operator (4.6) with the sets $\mathcal{U}$ and $\tilde{\mathcal{U}}$. Then, we can rewrite the closed-loop dynamics as

$$\begin{aligned}
\boldsymbol{u}_{k+1} - \tilde{\boldsymbol{u}} &= \Pi_{\tilde{\mathcal{U}}}^{\boldsymbol{P}}\left(\boldsymbol{K}(\boldsymbol{u}_k - \tilde{\boldsymbol{u}}) - \boldsymbol{E}(\boldsymbol{e}_k - \tilde{\boldsymbol{e}})\right) && (4.26)\\
\boldsymbol{e}_{k+1} - \tilde{\boldsymbol{e}} &= \boldsymbol{B}(\boldsymbol{u}_k - \tilde{\boldsymbol{u}}) + \boldsymbol{A}(\boldsymbol{e}_k - \tilde{\boldsymbol{e}}), && (4.27)
\end{aligned}$$

which is a partially projected linear system in the state $[(\boldsymbol{u}_k - \tilde{\boldsymbol{u}})^T, (\boldsymbol{e}_k - \tilde{\boldsymbol{e}})^T]^T$. Note that the open loop dynamics of (4.26) has the state matrix $\boldsymbol{Z}$, thus $\rho(\boldsymbol{Z}) < 1$ is again a necessary condition for stability. In [122] Chapter 3, global asymptotic stability of discrete-time saturated linear systems is studied. An extension of the results from [122] for the partial projection in (4.26) can be utilized to provide sufficient conditions for global asymptotic stability of the partially projected linear dynamics of (4.26) within a single layer dynamics group. Similarly, stability conditions for partially saturated continuous-time linear systems given in [96] may be easily adapted to discrete-time arguments to state global asymptotic stability results.

For both affine and linear representations of the closed-loop dynamics, the input dynam-

ics matrix $\boldsymbol{B}$ is required to evaluate the true spectral radius of $\boldsymbol{Z}$. Since the true dynamics are unknown to the controller in the case of model mismatch, the spectral radius cannot be computed exactly. We use the model information $\bar{\boldsymbol{B}}_k(\boldsymbol{I} + \boldsymbol{W}_k)$ (without the unknown unstructured uncertainty Δ_k^B) to approximate the spectral radius. For a practical implementation, we choose $\boldsymbol{Q}, \boldsymbol{S}$ such that Proposition 4.3 and the spectral radius conditions hold. A conservatively high $\boldsymbol{S}$ gain may be initially chosen to account for the model mismatch. Then, we perform experimental runs with the process with reduced values of $\boldsymbol{S}$ until the control input becomes unstable within a single layer dynamics group.

4.1.5 Simulation Case Study for LP-ILC

We illustrate the proposed controller on a simulation case study for a fused deposition modeling (FDM) process. We first present the AM process and the model we use for the spatial dynamics. Then we present the controller on a layer-wise varying spatial deposition path.

We consider the FDM process conceptually illustrated in Fig. 4.1. In FDM, a thermoplastic material is deposited onto a build bed via a numerically controlled extruder head. Deposited material forms a bead geometry with an ellipsoidal cross-sectional shape [2]. After a layer is deposited, the deposition system moves up (or the build bed moves down) to accommodate the deposition of a new layer. The layer-to-layer deposition of layers forms a 3D geometry.

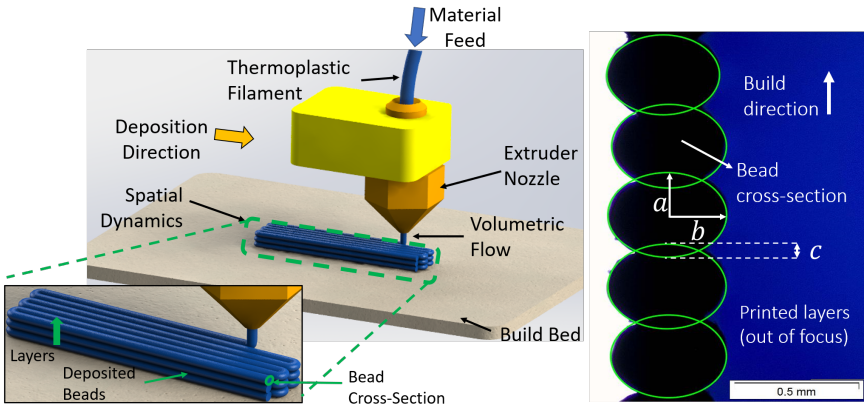


Figure 4.1: **Left:** Illustration of the FDM process and the different types of dynamics in the process [13]. **Right:** Microscope image of the cross-sectional geometry of FDM printed shell geometry from [2].

FDM is currently one of the most used AM processes due to its flexibility to utilize a wide range of materials and its low operational costs. We proposed a control-oriented

spatial dynamical model for the L2L dynamics of the FDM process in Chapter III [13, 16]. Additionally, we characterized the cross-sectional dimensional properties of deposited beads with an experimental verification in the coauthored work presented in [2]. Here we utilize a high-fidelity cross-sectional model from [2] with the spatial L2L dynamics model to present case studies on an inverted pyramid geometry illustrated in Fig. 4.2.

While we have demonstrated experimental studies in Chapter III for layer-to-layer measurements and stability analysis, implementation of layer-to-layer control on off-the-shelf machines poses a number of additional challenges. Most commercially available FDM machines run open-loop by executing a provided G-Code file line by line. Thus, to implement layer-to-layer control by adjusting the extrusion commands for each layer in closed-loop, either the control logic of the FDM machine should be modified to instead use closed-loop control inputs at each layer, or the G-Code commands that are executed by the machine must be adjusted in run time. Implementation of such approaches require labor-intensive experimentation and validation to reliably manipulate control actions in a layer-to-layer fashion. In this section, we instead focus on the controller development and performance through demonstrations on simulation studies to demonstrate the utility and benefit of an L2L control approach which would potentially be implemented on a real system by addressing the aforementioned challenges.

We present the model of the spatial dynamics for height evolution in the process in two parts.

4.1.5.1 Material input model

In the coauthored work in [2], a nonlinear model for the cross-sectional properties of the beads in an FDM process is given. The cross-sectional parameters a, b, c are illustrated on cross-sectional cuts of experimental specimen in Fig. 4.1. It is important to note that the model derived in [2] *does not consider any closed-loop deposition rate control*. The cross-sections of subsequent layers intersect each other at the layer-to-layer intersection zone, denoted by the parameter c. Therefore, we have the relationship $h_\ell = 2a - c$ for the cross-sectional height. We do not model the parameter c here, and attribute it to the model uncertainty. The nonlinear dynamics in [2] are given as

$$a = f(\kappa_e \dot{E}_{\min}, h_\ell), \tag{4.28}$$

where $\dot{E}_{\min} \in \mathbb{R}_+$ denotes the minimum extrusion rate considered in model development, $\kappa_e \in [1, \bar{\kappa}]$ denotes the extrusion multiplier which is used as a free variable to determine the extrusion rate in the process, and h_ℓ is the fixed layer height of the process. Refer to the

coauthored work in [2] for further details. As we consider a linear spatial dynamical model in this chapter, we linearize (4.28) at an operating point $h' = 250\mu m$ (i.e., a first order approximation at h'). We denote the resulting dynamics as $a = a_h + \Delta a \kappa_e$. Additionally, we adjust the input range in the model so that we have $\kappa_e \in [0, \kappa]$. To define the dynamics in (4.1) we need to also consider the spatial deposition path for the process.

4.1.5.2 Spatial Deposition Path

The spatial deposition path of the inverted pyramid shell geometry is shown in Fig. 4.2-a. We consider a spatial domain Λ aligning with the $i - j$ plane, as partially shown in Fig. 4.2-b. At each layer, a square shell geometry in the $i - j$ plane is deposited following the spatial deposition path points in Λ. The solid, thick blue lines and dotted red lines in Fig. 4.2 (b) show the corresponding partial spatial deposition paths at layers 5 and 6. As shown in Fig. 4.2 (b), the spatial deposition path changes after five layers for a total of $n_\ell = 20$ layers. The number of spatial deposition points on the spatial deposition path are $p^1 = 44$ for the layers $1-5$, $p^2 = 52$ for layers $6-15$, and $p^3 = 60$ for layers $16-20$. Each five layers corresponds to a layer dynamics group. Since the input matrix B_k captures the spatial deposition path, diagonal matrices with the linearized input terms corresponding to the points with deposition are evaluated for the simulation. We also illustrate some results on a process with layer-wise invariant dynamics, i.e., a single layer dynamics group.

4.1.5.3 State matrices

The material deposition in the FDM process results in 3D deposited beads with ellipsoidal cross-sections as shown in Fig. 4.1. To simplify model development, we only consider the L2L height evolution at the locations of the spatial deposition paths at each layer. These locations correspond to the top of the cross-section of a deposited bead shown in Fig. 4.1.

Consequently, the dimensionality of our model is the sum of the number of spatial deposition locations across all layers, $n_x = n_u = \sum_i p^i = 156$. The input matrix $\boldsymbol{B}_k = diag([\Delta a \circ \mathbf{1}_{p_1}, \mathbf{0}_{p_2+p_3}]), k \in [1, 4]$ is then a diagonal matrix with nonzero entries as Δa (see (4.28)) for the locations corresponding to the deposition path for the layers $1 - 5$. Here $\circ$ denotes the entry-wise product. The input matrices $\boldsymbol{B}_k$ for the remaining layers are defined similarly for the corresponding state dimensions. We use the input dynamics $\boldsymbol{B}_1$ with a feasible initial input to print the first layer $\boldsymbol{x}_1$ in the simulation study, and then utilize the LP-ILC to learn the L2L spatial input for the subsequent layers. The references for each layer are adjusted by the constant term (a_h) in the linearization. To simulate the process

with model mismatch, we apply a norm bounded disturbance to the spatial dynamics such that we have $\bar{\boldsymbol{B}}_k \neq \boldsymbol{B}_k$. The controller utilizes $\bar{\boldsymbol{B}}_k$ with the given control updates and the process is simulated with the true dynamics $\boldsymbol{B}_k$.

The spatial state matrix $\boldsymbol{A}_k$ is defined as an identity matrix for all layers except for layer indices 5 and 15. As noted earlier, the deposition path varies between layers $5-6$ and $15-16$. Figure 4.2 (b) illustrates a partial corner spatial deposition path between layers $5-6$ and we assume that the material deposition on layer 6 is built on top of the corresponding *neighboring* locations at layer 5. The relationship of neighboring locations at subsequent layers is illustrated with the thin black arrows in Fig. 4.2. The state matrices $\boldsymbol{A}_5$ and $\boldsymbol{A}_{15}$ encode this relationship between layers in addition to the identity property of other layers. For further details on the L2L modeling approach see [13]. We have the input constraints given as $\boldsymbol{u}_k \in \mathcal{U} = [0, \kappa]^{n_u}$. We assume that a spatial measurement system similar to the one in [2, 16] is in place and measures a spatial height map of a layer after it is formed, by scanning the deposition path at that layer.

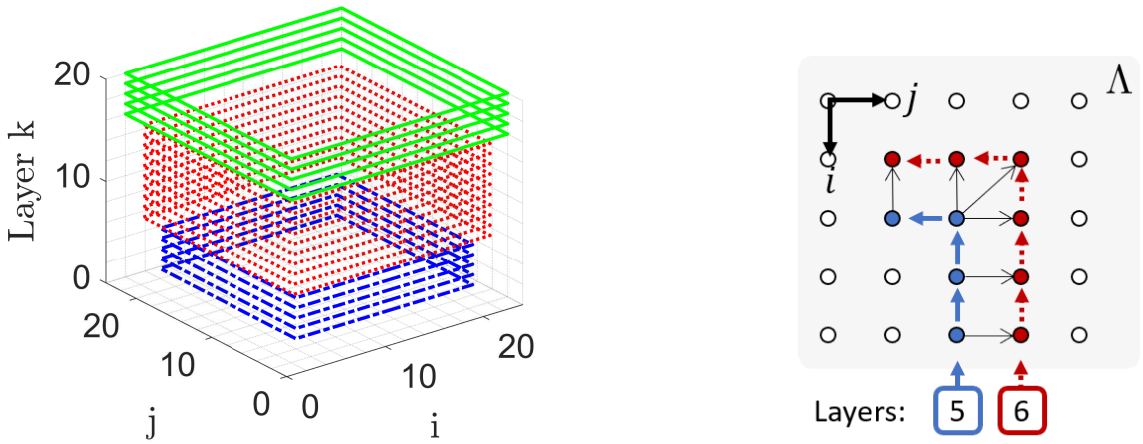


Figure 4.2: (a) Perspective view illustration of the inverted pyramid 3D reference geometry considered in the case study. (b) Top-down view illustration of partial spatial deposition paths on layers 5 and 6, together with the height information relationship between layers (thin black arrows).

4.1.6 Results and Discussion

We first present results for the simulation study with a single layer dynamics group to demonstrate the propositions on single layer dynamics groups. Then, we illustrate the simulation results for the inverted pyramid geometry in Fig. 4.2 to illustrate the effectiveness of the proposed approach.

4.1.6.1 Single Layer Dynamics Group

Figure 4.3 compares the convergence results of the normalized error and the $\boldsymbol{u}^r$ within a single layer dynamics group for varying spectral radii of $\boldsymbol{Z}$ in (4.18). As proposed, we see that whenever the spectral radius is smaller than unity, we get convergent behavior of both the error and the input signal, whereas for the case with $\rho(\boldsymbol{Z}) = 1.02$ we get divergent behavior.

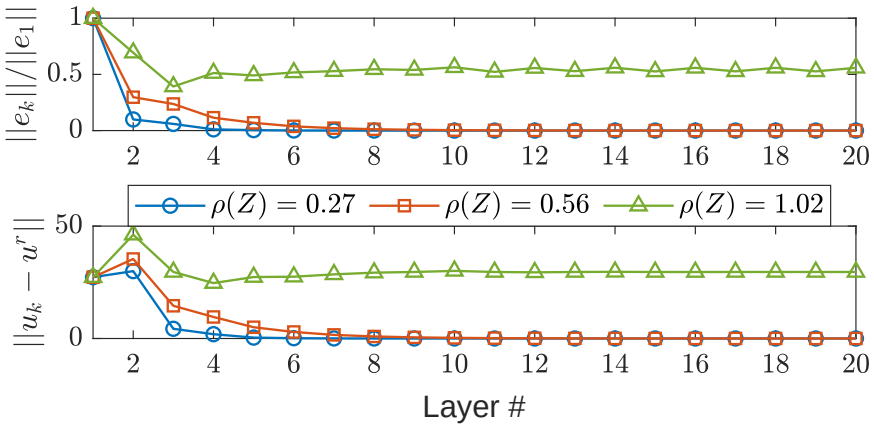


Figure 4.3: Comparison of varying model mismatch levels for the single layer dynamics group process.

Figure 4.4 illustrates the effect of different upper bounds κ on the convergence behavior. As the upper bound on the allowable control input is tightened, the convergence rate is slowed. Additionally, all cases converge to the fixed point since the reference height of the final layer is reachable under the given control inputs and the reference $\boldsymbol{u}^r \in \mathcal{U}$.

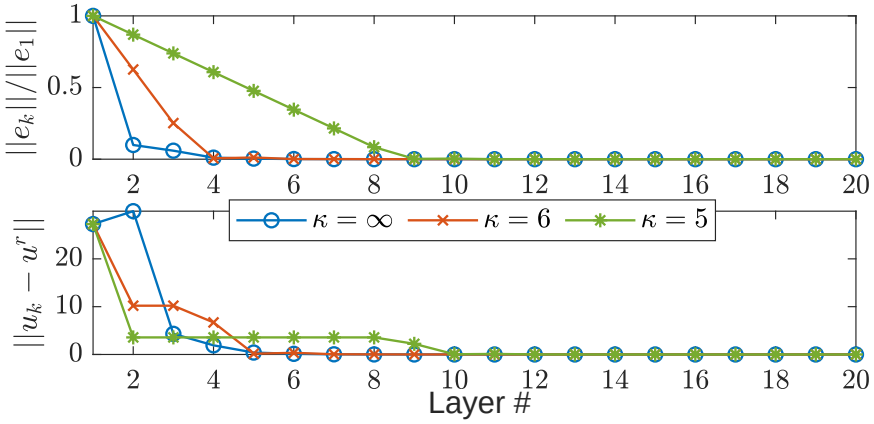


Figure 4.4: Comparison of different input constraint levels for the single layer dynamics group process.

4.1.6.2 Multiple Layer Dynamics Groups

Figure 4.5 illustrates the effect of model mismatch for the inverted pyramid geometry. As stated earlier, the bound on the inputs evaluated by the proposed controller can be evaluated by the corresponding layers in the same layer dynamics group and switches between groups. As expected, we see that under perfect model information, we get convergence of the error even when we switch between layer dynamics groups on layers 6 and 16. For the case with model mismatch, we observe the error induced by the model gradient, which is characterized by Proposition 4.4.

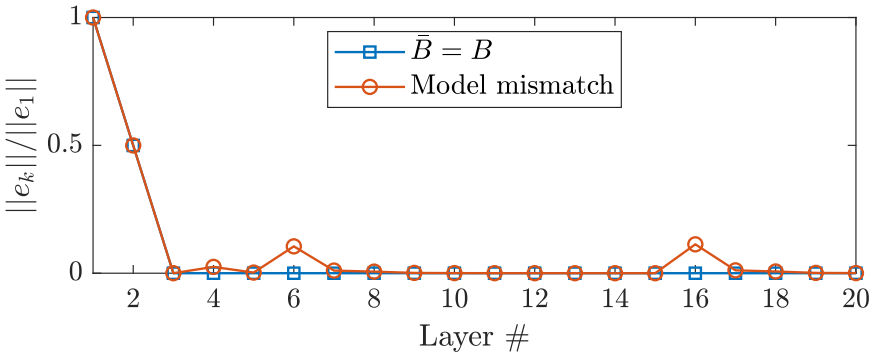


Figure 4.5: Comparison of the model mismatch versus perfect model case for the inverted pyramid geometry in Fig. 4.2 with multiple layer dynamics groups.

Figure 4.6 illustrates the effectiveness of the proposed model with the gradient rule (4.8). We compare the normalized errors of the proposed control versus two other controllers that either use data gradient or the model gradient for all the layers. We see that the model gradient controller performs worse when compared to the proposed approach, even in the switch layers. This is an expected result since the performance at the switch layers depends on the error of the previous layer through $\boldsymbol{\eta}_k$, and the proposed controller with the data updates attains a lower error and $\boldsymbol{\eta}_k$ prior to the switch layers. The data gradient controller performs comparable to the proposed approach for within layer dynamics groups. However, since switch layers are not yet observed by the data gradient, we have large errors for the switch layers. The proposed approach presents a hybrid of the other two and maintains a low error by utilizing the data gradient or the model gradient whichever more advantageous.

The results for the inverted pyramid geometry show how the IP-ILC controller may be utilized with iteration-varying spatial deposition paths under constrained inputs. This is an important result as most FDM produced parts utilize layer-wise varying spatial deposition paths. While previous research relies on layer-wise invariant initial conditions and spatial deposition paths, our proposed controller is able to address a more general and realistic

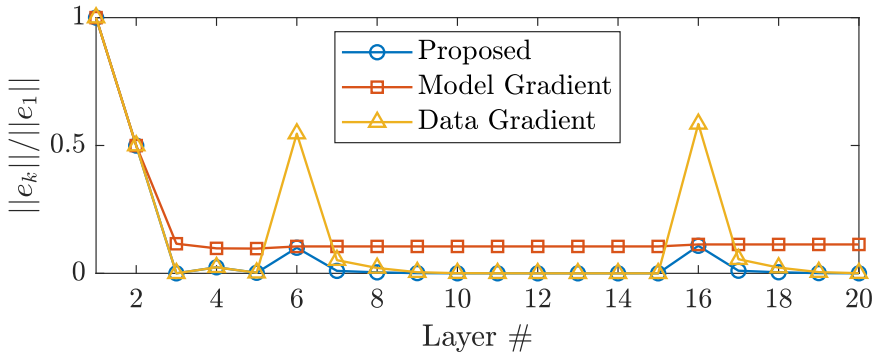


Figure 4.6: Comparison of the proposed approach with the gradient rule (4.8) versus the cases with data gradient and model gradient only for the inverted pyramid geometry in Fig. 4.2 with multiple layer dynamics groups.

modeling and assumption framework for FDM processes.

4.1.7 Section Conclusion

In this section, a novel layer preview ILC scheme is presented for AM processes with layer-wise varying references and dynamics. We provide theoretical analysis for the LP-ILC controller to ensure convergence and stability under layer-wise varying dynamics and constrained inputs. Our preliminary simulation studies show that the LP-ILC may be used with practical AM processes with layer-wise varying spatial deposition paths and non-uniform initial conditions.

While the layer preview ILC provides a practical approach for reference tracking with a closed-loop learning controller, it does not provide a framework to predict if a finished part will conform to spatial performance measures such as layer-to-layer stability. In an industrial setting, where we want to maximize the yield of a given manufacturing process, it is crucial to understand during production if a finished product will conform to predefined tolerances. Therefore, we are often interested in answering the question *does there exist an L2L stabilizing controller in the remaining layers of the process.* In the next section, we formalize this question in the context of L2L stability and L2L finite stability presented in Chapter III, and provide details on L2L stabilizing controllers.

4.2 Layer-to-Layer Stabilizing Control for Additive Manufacturing Spatial Dynamics

In this section, we develop closed-loop controllers for the broad class of *layer-wise spatially varying* LSV systems. Our proposed controllers can layer-to-layer stabilize the spa-

tial dynamics either asymptotically or in finitely many layers, whenever possible within the constrained input authority. Specifically, we propose a control framework to develop spatial controllers that achieve *layer-to-layer stability* (L2L stability) for AM processes [13, 16], presented in Chapter III. Our developments advance the past literature as our work aims to meet layer-to-layer stability requirements that are given by a designer based on the part functionality requirements. We further provide methods to evaluate *certificates of stabilizability*, which is a novel concept that can be used to check whether a given process is layer-to-layer stabilizable in the remaining layers of the process or not. This certificate is a novel effort to provide an in-situ certification for an AM printed part meeting the functional requirements, given constrained control authority over the system. A certificate of stabilizability uses the process model, constraints, and measurements to predict if the spatial process can be L2L stabilized in future layers. These certificates may be used to stop a process that is not stabilizable, i.e., one that is not expected to meet functional requirements by the end of the process, given the current conditions and control constraints on the system. Therefore, our developments in this section enable novel autonomous AM processes that are able to perform closed-loop control and certify in-situ if a printed device is expected to meet its functional requirements.

In the existing literature, the layer-to-layer spatial performance is not used as a closed-loop performance metric to ensure part functionality. However, utilizing layer-to-layer spatial performance metrics improves reliability and repeatability of the process via closed-loop control since these metrics consider the functionality of the printed part and a closed-loop controller that considers these metrics aims to ensure the printed part functionality. As AM processes manufacture physical parts that have functional requirements, e.g., dimensional tolerances, controlling the process so that the resulting printed device meets these functional requirements is of great importance. Existing controllers in the literature focus on tracking stationary references at a given layer without considering the behavior of the spatial dynamics during the layer-to-layer process. While this is a feasible approach for certain cases, the layer-to-layer spatial dynamics mainly determine the functional performance of a printed object, and thus require further attention.

Spatial dynamics of AM processes concern the physical properties of the deposited material at each layer in the LSV process. Thus, developing modeling and control methods for spatial dynamics of AM processes enables applications to control the functional performance of a printed device. However, most of the existing literature in spatial control provides controllers based on models that do not capture the layer-to-layer dynamics of the process. Most of these works consider AM processes as a repetitive system in the layer domain and provide controllers to track a given reference in finite repetitions [4, 94, 170].

As a result, these controllers often overlook the effect of a previous layer on the process.

Additionally, existing controllers do not capture the constraints associated with inputs to the system $\mathcal{U}$ and finite number of layers (i.e. $k \in [1, n_\ell], n_\ell < \infty$). Layer-to-layer stability is a measure of how well a spatial process adheres to a given reference within predefined stability bounds (Chapter III). The layer-to-layer performance of the spatial dynamics characterizes the layer-to-layer stability of the system in future layers and the layer-to-layer stabilizability of the system in the remaining layers. The latter is an especially important problem as mid-print failures in AM processes pose a great challenge for the reliability and repeatability of high performance AM processes [13, 14, 16, 155].

The main contribution of this chapter is the introduction of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints (*C2*). The specific contributions of this section focus on the layer-to-layer stabilizability problem and develop further modeling structures for efficient computation of reachability arguments. The specific contributions within this context are given by:

(*C2*-4) Development of in-situ closed-loop controllers to layer-to-layer stabilize LSV systems and evaluation of the proposed controllers for the LSV dynamics with specialized forms.

(*C2*-5) Development of novel methods for evaluating stabilizability certificates of an LSV system for practical applications with in-situ spatial measurements.

4.2.1 LSV System Setting

We investigate spatial dynamical systems with the following general form given in Chapter III, equation (3.1), repeated here:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k), \tag{4.29}$$

where $\boldsymbol{u}_k \in \mathbb{R}_+^{n_u}$, $\boldsymbol{x}_k \in \mathbb{R}_+^{n_x}$ for all $k \in \mathbb{Z}_+$. The system dynamics is given by $f : \mathbb{R}_+^{n_x} \times \mathbb{R}_+^{n_u} \times \mathbb{Z}_+ \to \mathbb{R}_+^{n_x}$. The vector $\boldsymbol{x}_k$ is the spatial state of the system at *layer* k over a 2-D discretization given by $\boldsymbol{\Lambda} \subset \mathbb{Z}^2 \subset \Pi_{\mathbb{Z}^2}(\mathcal{V})$, where $\mathcal{V} \subset \mathbb{R}^3$ is the volume of interest for the spatial process given in (4.29), and $\Pi_{(\cdot)}(\cdot)$ is a projection operator. At each layer, a constrained input $\boldsymbol{u}_k \in \mathcal{U}$ is applied to the system. We assume state observability, and for the sake of simplicity denote the output of the system as the spatial state evolution itself, i.e. $\boldsymbol{x}_k$.

We assume that (4.29) represents the steady-state response of the spatial process in response to the spatial input $\boldsymbol{u}_k$ at a given layer, where k denotes the *layer* index (see Chapter III for further details). Therefore, the state $\boldsymbol{x}_k$ denotes the height of the process up to the layer index, in response to the spatial input $\boldsymbol{u}_k$. The state of the system $\boldsymbol{x}_k$ evolves at each layer k due to the input to the system $\boldsymbol{u}_k$, thus, we name the spatial systems of the form (4.29) as layer-wise spatially varying (LSV) systems. We consider AM as an LSV system where a spatiotemporal input is applied to the system at each layer to deposit material, and as the deposition in a layer is completed, the process continues to the subsequent layers to deposit additional material to manufacture a 3-D object. Therefore, each layer in (4.29) represents the effect of material deposition at the end of a printed layer in an AM process. The spatial state $\boldsymbol{x}_k$ denotes the steady-state physical evolution of the printed part due to the material deposition at the end of a layer, i.e., as we finish printing the layer. Here we limit our discussions to the case where $\boldsymbol{x}_k$ denotes the spatial height map up to layer k (see Chapter III). Then, by controlling the state $\boldsymbol{x}_k$ at each layer, it is possible to control the dimensional accuracy. Note that by defining proper output functions on the state $\boldsymbol{x}_k$ it may be possible to represent additional physical properties such as, strength, optical, or electrical properties of a printed device either measured directly in-situ or estimated through online estimation either in-situ or ex-situ. Here we focus on defining notions of L2L stabilizability for the L2L state of the process, $\boldsymbol{x}_k$.

4.2.2 Section Notation

The state trajectory for a certain initial spatial state $\boldsymbol{x}_{k_0}$ and input trajectory $\{\boldsymbol{u}_k\}_{k_0}^{N}$ pair is denoted by $\phi(\boldsymbol{x}_{k_0}, \{\boldsymbol{u}_k\}_{k_0}^{N-1}) = \{\boldsymbol{x}_k\}_{k_0}^{N}$, where we instead compactly use $\phi(\boldsymbol{x}_{k_0}, \boldsymbol{u})$ with $\boldsymbol{u} = \{\boldsymbol{u}_k\}_{k_0}^{N-1}$. Additionally, we denote $\phi^N(\boldsymbol{u}) = vec(\phi(\boldsymbol{x}_{k_0}, \boldsymbol{u}))$ as the lifted form of the solution trajectory for the next N layers, where $vec(\cdot)$ denotes a column vector concatenation made from its arguments. Similarly, $\phi(\boldsymbol{x}_{k_0}, \boldsymbol{u}; N)$ denotes the spatial state $\boldsymbol{x}_N$ at layer N, under the input $\boldsymbol{u}$.

The infinity ball of size r at point $\boldsymbol{x}$ is defined as $\beta^{\infty}(\boldsymbol{x}, r)$. The state constraint set is denoted with $\mathcal{X}$ and the input constraint set is denoted with $\mathcal{U}$. Both sets $\mathcal{X}$ and $\mathcal{U}$ are closed and convex.

4.2.3 Definitions

First, we provide a number of definitions that will be useful in future discussions. A desired spatial state trajectory is denoted as $\boldsymbol{X} = \{\boldsymbol{x}_k^d\}_1^{n_\ell} = \{\boldsymbol{x}_1^d, \boldsymbol{x}_2^d, \ldots, \boldsymbol{x}_{n_\ell}^d\}$. In this chapter we are interested in evaluating a control input that layer-to-layer stabilizes a pro-

cess. We define a control policy that ensures if a process is L2L stable at a current layer, it remains stable for future layers. Thus the stabilizability problem concerns finding a control input to drive the system to an L2L stable layer, e.g., one that is close (in the normed sense) to a desired state, and then utilizing a control policy that ensures stability of future layers.

Definition 4.7 (Single layer reachable set)**.** The single layer reachable set from the initial conditions in S, denoted as $\mathcal{R}_1(S) \subseteq \mathcal{X}$, is defined as $\mathcal{R}_1(S) = \{\boldsymbol{x}_{k+1} \in \mathcal{X} \mid \exists \boldsymbol{u}_k \in \mathcal{U}, \boldsymbol{x}_k \in S \text{ s.t., } \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \in \mathcal{X}\}$.

Definition 4.8 (N-layer reachable set)**.** The set of all states that are reachable from an initial set S in N layers is called the N-layer reachable set of S and defined iteratively as $\mathcal{R}_N(S) = \mathcal{R}_1(\mathcal{R}_{N-1}(S))$, with $\mathcal{R}_0(S) = S$.

Definition 4.9 (Single layer backward-reachable set)**.** The single layer backward-reachable set from the target set S is the set of all spatial states at layer k for which there exists a feasible control $\boldsymbol{u}_k \in \mathcal{U}$ such that $f(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{S}$. The set is formally denoted as $\mathcal{B}_1(S) \subseteq \mathcal{X}$, where $\mathcal{B}_1(S) = \{\boldsymbol{x}_k \in \mathcal{X} \mid \exists \boldsymbol{u}_k \in \mathcal{U} \text{ s.t., } \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \in \mathcal{S}\}$.

Definition 4.10 (N-layer backward-reachable set)**.** The set of all states that are backward-reachable from the target set S in N layers is recursively defined as $\mathcal{B}_N(S) = \mathcal{B}_1(\mathcal{B}_{N-1}(S))$, with $\mathcal{B}_0(S) = S$.

Definition 4.11 (Additive property of LSV systems)**.** An LSV system given in (4.29) is said to have the additive property if (i) the state trajectories $\phi(\boldsymbol{x}_{k_0}, \boldsymbol{u})$ are nondecreasing in the layer index k (i.e., $\boldsymbol{x}_k \preceq \boldsymbol{x}_{k+1};\ \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k), \forall k$) for any $\boldsymbol{u}_k \in \mathcal{U}$ and (ii) $f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \succeq f(\boldsymbol{x}'_k, \boldsymbol{u}'_k, k)$ whenever $(\boldsymbol{x}_k, \boldsymbol{u}_k) \succeq (\boldsymbol{x}'_k, \boldsymbol{u}'_k)$.

Most physical applications of LSV systems, such as AM, have the additive property due to the underlying physical process they represent. An immediate result of the additive property is the following.

Claim 4.12. *An LSV system with the additive property is a monotone control system [6], i.e.*

$$(\boldsymbol{x}_0, \boldsymbol{u}) \preceq (\boldsymbol{x}'_0, \boldsymbol{u}') \implies \phi(\boldsymbol{x}_0, \boldsymbol{u}) \preceq \phi(\boldsymbol{x}'_0, \boldsymbol{u}'). \tag{4.30}$$

Proof. Follows directly from the definition of monotone control systems in [6] and the definition of the additive property for LSV systems. □

4.2.4 Monotone operator theory

In this section, we leverage the monotone nature of certain classes of LSV systems to propose control evaluation methods that are fast and scalable. We accomplish this by posing the control evaluation problem as a monotone inclusion problem. Here, we define the monotone inclusion problem as it is closely related to the controllers we employ in this chapter and provide some preliminary background that will be useful with later discussions. The monotone operator theory is one of the fundamental tools in optimization theory and is well studied in the literature [73, 153, 168]. For further details and an insightful survey on monotone operators, the reader should refer to [168] and the references therein.

An operator $F : \mathbb{R}^n \to \mathbb{R}^n$ is called monotone if

$$(y - y')^T(x - x') \geq 0, \quad \forall (x, y), (x', y') \in \mathbf{Gr}\, F \tag{4.31}$$

where $(x, y), (x', y')$ are auxiliary variables, and we define the *graph* of F as $\mathbf{Gr}\, F = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^n : y \in F(x)\}$. We call F as γ-*strongly monotone* if $(x - x')^T(y - y') \geq \gamma||x - y||^2, \forall (x, y), (x', y') \in \mathbf{Gr}\, F$. We call F *maximal monotone* if there exists no proper monotone extension to $\mathbf{Gr}\, F$ on $\mathbb{R}^n \times \mathbb{R}^n$ [34]. In other words, if F is monotone but not maximal monotone, then there exists $(x, y) \notin \mathbf{Gr}\, F$, such that $F \cup \{(x, y)\}$ is still monotone [168]. In this section, we assume maximality of a monotone operator for theoretical analyses, unless stated otherwise. Often we are interested in finding the set of zeros for F, given as $\mathbf{zer}\, F = \{x \in \mathbb{R}^n : 0 \in F(x)\}$. Finding the set $\mathbf{zer}\, F$ for a maximal monotone F is a fundamental problem that generalizes many applications in convex optimization with first order methods.

Now, consider system (4.29) with $n_x = n_u$, and define $F(\boldsymbol{u}) = \phi(\boldsymbol{u})$ for a given initial state $\boldsymbol{x}_{k_0}$ as a maximal monotone operator satisfying (4.31). We name such systems where the solution operator ϕ of the system (4.29) is a monotone operator as *monotone operator admitting systems*. Monotone operator admitting systems play a crucial role to provide some important results in this chapter. Note that, for a monotone operator admitting system $F(\boldsymbol{u})$, the condition (4.31) implies the monotone control system condition in (4.30). Next, we state some standing assumptions that will be utilized throughout the rest of the chapter.

Assumption 4.13. *Spatial dynamics $\boldsymbol{x}_k$ (and consequently the spatial dynamical state) of the process is measurable.*

Assumption 4.14. *The input $\boldsymbol{u}_k$ can be changed after each layer, and the desired states $\boldsymbol{x}_k^d$ for each layer are known and fixed.*

Assumption 4.15. *An input of magnitude $\mu \in \mathbb{R}_+$ at a spatial location $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ has finite support over the spatial domain including the location $\boldsymbol{\lambda}$, is bounded and continuous over its support.*

4.2.5 L2L Stabilizability Problem Formulation and Certificates

Consider the LSV system $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k)$ in (4.29) with the Assumptions 4.13-4.15. Additionally, we define the layer-to-layer stable sets for each layer as

$$\mathcal{S}_k = \{x \in \mathbb{R}^{n_x} \mid x \in (\boldsymbol{x}_k^d + D_k) \subset \mathcal{X}\}, \quad \forall k \in K, \tag{4.32}$$

where we have $K = [1, n_\ell]$. Furthermore, at each layer, the set D_k is defined as closed convex stability bound around the desired trajectory $\boldsymbol{x}_k^d$. Following the definition of L2LS, we can think of D_k as a ball around the desired spatial trajectory $\boldsymbol{x}_k^d$ such that the L2L stability condition from Chapter III holds. For example, as the state $\boldsymbol{x}_k$ denotes the spatial height map of the process up to layer k, the L2L stability condition constrains how close the spatial state $\boldsymbol{x}_k$ should be to the desired height map $\boldsymbol{x}_k^d$ in order for the L2L process to be L2L stable. Defining the L2L stability using the L2L stable sets provides us with a flexible framework for characterizing dimensional performance with respect to dimensional tolerances on the spatial state. Therefore, a process is L2L stable only if the heights of each layer are within the predefined tolerances given by $\mathcal{S}_k$.

The *layer-to-layer stable tube* is defined as a sequence $\boldsymbol{\mathcal{S}} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n_\ell}\}$, which is inspired by a similar terminology in the tube model predictive control literature [114]. Since the sets $\mathcal{S}_k$ are design variables, we assume they are designed to be forward reachable in the layer domain, k.

Assumption 4.16. *The L2L stable sets $\mathcal{S}_k$ satisfy*

$$\mathcal{R}_1(\mathcal{S}_k) \cap \mathcal{S}_{k+1} \neq \emptyset, \quad \forall k \in K^-, \tag{4.33}$$

where $K^- = [1, n_\ell)$.

This condition ensures that the L2L stable set $\mathcal{S}_{k+1}$ at layer $k+1$ is reachable from the L2L stable set at layer k (i.e., there exists an input taking any state $\boldsymbol{x}_k \in \mathcal{S}_k$ to the set $\mathcal{S}_{k+1}$). Note that although the whole set $\mathcal{S}_{k+1}$ may not be reachable from the set $\mathcal{S}_k$, as long as (4.33) holds for all layers there exists a control input to ensure L2L stability for the process. Thus, the reachability condition for layer-to-layer stability requires adequate control authority on the layer-to-layer process to track an L2L stable set at an upcoming layer (i.e., control the state to be in a L2L stable set). If a feasible control does not exist, due

to the lack of sufficient control authority, the process is not L2L stabilizable. A decision-maker may utilize the methods proposed in this section to analyze the L2L stabilizability of the process and take further actions (e.g., stop the process if necessary) to mitigate further time and material costs.

Assuming (4.33) holds, we define a *nominal L2L stabilizing control policy*

$$\boldsymbol{\pi} = \{p_1(\cdot), p_2(\cdot), \ldots, p_{n_\ell - 1}(\cdot)\}, \tag{4.34}$$

which satisfies the following conditions

$$p_k(\boldsymbol{x}_k) \in \mathcal{U}, \quad \forall (\boldsymbol{x}_k, k) \in \mathcal{S}_k \times K^- \tag{4.35a}$$

$$f(\boldsymbol{x}_k, p_k(\boldsymbol{x}_k), k) \in \mathcal{S}_{k+1}, \quad \forall (\boldsymbol{x}_k, k) \in \mathcal{S}_k \times K^-. \tag{4.35b}$$

The condition (4.35) ensures that the control policy $\boldsymbol{\pi}$ is an L2L stabilizing control sequence if the spatial state $\boldsymbol{x}_k$ is in an L2L stable set at layer k. We later provide details on deriving the nominal stabilizing controller policy $\boldsymbol{\pi}$. An immediate result is the following.

Proposition 4.17. *(Nominal forward L2L stability) For an LSV system in (4.29) with the L2L stable tube that satisfies (4.33), if $\boldsymbol{x}_{k_0} \in \mathcal{S}_{k_0}$, then the system is layer-to-layer stable for all remaining layers under the control policy $\boldsymbol{\pi}$ that satisfies (4.35).*

Proof. Satisfying the conditions in (4.35) means that $\phi(\cdot, \cdot; k_0 + 1) \in \mathcal{S}_{k_0+1}$ for all future layers ($[k_0 + 1, n_\ell]$). Since the current layer is also L2L stable, it follows from the definition of L2L stability that all the remaining layers are layer-to-layer stable. □

The nominal forward L2L stability ensures that once the system is in an L2L stable set, all the remaining layers are L2L stable under the control polity $\boldsymbol{\pi}$. This is an important result as we focus on the L2L stabilizable controller evaluation in two parts. Namely, we first seek inputs that steer an initial state $\boldsymbol{x}_{k_0} \notin \mathcal{S}_{k_0}$ to an L2L stable set $\mathcal{S}_{k'} \in \boldsymbol{\mathcal{S}}$ in future layers. Then, we use the nominal L2L stabilizing controller $\boldsymbol{\pi}$ to keep the system forward L2L stable for the remaining layers.

Remark 4.18. *The nominal forward L2L stability proposition does not ensure finite convergence to the desired state trajectory, but rather ensures the spatial states remain in L2L stable sets for future layers. While finite convergence may be feasible for certain cases, we do not assume it is always feasible unless indicated otherwise.*

Formally, in this section, we analyze the following L2L stabilizability problems.

(IV-P1) L2L stabilizability problem: If at layer $k_0 > 1$ the system (4.29) is layer-to-layer unstable (i.e. $\boldsymbol{x}_{k_0} \notin \mathcal{S}_{k_0}$), does there exist a control trajectory $\boldsymbol{u} = \{\boldsymbol{u}_{k_0}, \boldsymbol{u}_{k_0+1}, \ldots, \boldsymbol{u}_{n_\ell-1}\}$ that layer-to-layer stabilizes the system within the remaining $n_\ell - k_0$ layers?

(IV-P2) L2L finite stabilizability problem: If (IV-P1) is feasible, does there exist a controller that tracks a desired spatial trajectory (with ϵ precision) (a) asymptotically (investigated as a theoretical result) and (b) in finite layers with monotonic convergence (L2L finite stability)?

The two problems are investigated in their general form first and then specialized to certain assumptions on the form of the model in (4.29).

4.2.6 Formulation and Analysis of (IV-P1)

We start by formally defining (IV-P1) and analyzing its solutions. For the stabilizability problem, we note that if all spatial trajectories of an LSV system in (4.29) remain within $\mathcal{S}_k \in \boldsymbol{\mathcal{S}}, \forall k$, then the system is said to be L2L stable for *all* its layers. In this chapter we do not make such distinctions by noting that by Proposition 4.17, L2L stability for all layers is possible if the initial layer is within $\mathcal{S}_1$. Here we investigate the problem of designing L2L stabilizing controllers for LSV systems that are not L2L stable at a current layer. Therefore, we want to recover L2L stability in future layers for the process, as early (in the layer domain) as possible.

The problem of finding a *next layer* L2L stabilizing control input at layer k_0 for an L2L unstable state $\boldsymbol{x}_{k_0} \notin \mathcal{S}_{k_0}$ is formally given as

$$\text{Find: } \boldsymbol{u}_{k_0} \in \mathcal{U}, \quad \text{such that: } f(\boldsymbol{x}_{k_0}, \boldsymbol{u}_{k_0}, k_0) \in \mathcal{S}_{k_0+1} \tag{4.36}$$

In practice the problem in (4.36) may be infeasible if for example we have $\mathcal{R}_1(\boldsymbol{x}_{k_0}) \cap \mathcal{S}_{k_0+1} = \emptyset$ due to the constraints posed on the input of the system. If (4.36) is infeasible, we search for an L2L stabilizing control to L2L stabilize the system as soon as possible in the layer domain and before the last layer n_ℓ. The problem of finding an *N-layer L2L stabilizing control* input in the next N layers is given as

$$\text{Find: } \boldsymbol{u} \in \mathcal{U}^N, \text{ such that: } \phi(\boldsymbol{x}_{k_0}, \boldsymbol{u}; k_0+N) \in \mathcal{S}_{k_0+N}, \tag{4.37}$$

where the control input $\boldsymbol{u}$ is defined as a control trajectory for the next N layers. Note that for $N = 1$, we recover (4.36). Due to Proposition 4.19, as long as the control L2L stabilizes

the system at a layer $k < k_0 + N$, (i.e. we get $\boldsymbol{x}_k \in \mathcal{S}_k$), the problem (4.37) is feasible. Therefore, based on the choice of N, (4.37) may become infeasible. We characterize the lower bound for the feasibility of (4.37) in the following.

Proposition 4.19. *Given that at layer k_0, $\boldsymbol{x}_{k_0} \notin \mathcal{S}_{k_0}$, the process in (4.29) is L2L stabilizable in at least $\zeta(\boldsymbol{x}_{k_0}) = k_r - k_0$ layers if and only if $\exists k_r$ such that $\boldsymbol{x}_{k_0} \in \mathcal{B}_{k_r}(\mathcal{S}_{k_r})$ and $\boldsymbol{x}_{k_0} \notin \mathcal{B}_{k'}(\mathcal{S}_{k'})$ for any $k_0 \leq k' < k_r$.*

Proof. Since $\boldsymbol{x}_{k_0} \in \mathcal{B}_{k_r}(\mathcal{S}_{k_r})$, $\exists \{\boldsymbol{u}_k\}_{k_0}^{k_r}$ such that $\phi(\boldsymbol{x}_{k_0}, \{\boldsymbol{u}_k\}_{k_0}^{k_r})$, i.e., there exists a control sequence that drives the spatial trajectory $\boldsymbol{x}_{k_0}$ into the set $\mathcal{S}_{k_r}$ which is the L2L stable subspace for layer k_r, thus L2L stabilizing the spatial trajectories. Suppose there exists a control trajectory $\{\boldsymbol{u}'_k\}_{k_0}^{k'_r}$ that L2L stabilizes the system, where $k'_r < k_r$, then by definition $\boldsymbol{x}_{k_0} \in \mathcal{B}_{k_r-1}(\mathcal{S}_{k_r-1})$, which results in a contradiction and completes the proof. □

By computing the least number of layers for stabilizability as $\zeta(\boldsymbol{x}_{k_0}) \in \mathbb{N}$, it is possible to determine whether the spatial state $\boldsymbol{x}_{k_0}$ is stabilizable in the remaining $n_\ell - k_0$ layers of the process or not. Then by setting $N \geq \zeta(\boldsymbol{x}_{k_0})$, (4.37) is feasible.

Remark 4.20. *The system in (4.29) may be high dimensional in practice. Therefore it is favorable to choose N as small as possible, while ensuring feasibility, for (4.37) to reduce the computational complexity.*

Next, utilizing (4.36) we provide a formulation for the nominal L2L stabilizing control policy. Assume we have now $\boldsymbol{x}_{k'} \in \mathcal{S}_{k'}$. Then, based on the condition in (4.33), problem (4.36) is feasible. Therefore, we denote

$$p_k(\boldsymbol{x}_k) \in \{\boldsymbol{u}_k \in \mathcal{U} \mid f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) \in \mathcal{S}_{k+1}\}, \quad \forall k \geq k', \tag{4.38}$$

as the nominal L2L stabilizing controller for the system in (4.29) for subsequent layers. Therefore, (4.38) satisfies the conditions given in (4.35) and Proposition 4.17 holds. The following proposition summarizes the form of an L2L stabilizing controller for an LSV system in (4.29). A suitable control input $p_k(\boldsymbol{x}_k)$ is then evaluated by solving (4.36).

Proposition 4.21. *Suppose for a system of form (4.29), we have $\boldsymbol{x}_{k_0} \notin \mathcal{S}_{k_0}$, and suppose $\zeta(\boldsymbol{x}_{k_0})$ is known. Additionally, let $\boldsymbol{u}' = \{\boldsymbol{u}'_0, \boldsymbol{u}'_1, \ldots, \boldsymbol{u}'_{N-1}\}, N = \zeta(\boldsymbol{x}_{k_0})$ denote a solution of (4.37). Then the following control law provides an L2L stabilizing controller for the LSV system in (4.29).*

$$\boldsymbol{u}_k = \begin{cases} \boldsymbol{u}'_k & k \leq \zeta(\boldsymbol{x}_{k_0}) - 1, \\ p_k(\boldsymbol{x}_k) & \textit{otherwise}, \end{cases} \tag{4.39}$$

where $p_k(\boldsymbol{x}_k)$ is according to (4.38).

Proof. Follows immediately by (4.38) and (4.37). Therefore, (4.37) steers the spatial state to an L2L stable set, and the nominal L2L stabilizing policy in (4.38) keeps the system L2L stable for the remaining layers. □

In many practical cases, determining $\zeta(\boldsymbol{x}_{k_0})$ may be computationally infeasible, or equivalently demanding to solving (4.36) or (4.37). Therefore, we provide the minimization in the following.

$$\min_{\boldsymbol{u}\in\mathcal{U}^N} \frac{1}{2}\sum_{l=1}^{N} d^2(\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k'+l), \mathcal{S}_{k'+N}), \tag{4.40}$$

where we have the distance function $d(x, C) = ||x - \Pi_C(x)||$ for a closed convex set C. Then, (4.37) has a solution if and only if (4.40) has a solution $\boldsymbol{u}^*$ with $d^2(\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}^*; k'+N), \mathcal{S}_{k'+N}) = 0$. Notice that feasibility of reaching an L2L stable set in N layers is no longer an issue for (4.40) as the optimal solution is a constrained control that minimizes the distance of the LSV system to an L2L stable set. Consequently, (4.40) may be used when the feasibility of (4.37) is difficult to assess, or when the structure of the LSV dynamics are favorable with the minimization problem as we illustrate through examples. The solution of (4.40) is an L2L stabilizing control sequence in N layers only if the optimal input makes the distance to the desired L2L stable set zero, i.e. if (4.37) has the equivalent solution.

4.2.7 Formulation and Analysis of (IV-P2)

Here we first formally define (IV-P2) and analyze its solutions for the LSV system given in (4.29). While the L2L stabilizability problem evaluates a control input to steer the spatial trajectories into an L2L stable set in $\boldsymbol{\mathcal{S}}$, here the problem is to evaluate control actions to track the desired spatial trajectory $\boldsymbol{X}$ asymptotically or in finitely many layers if possible. An important observation is that (IV-P2) is feasible only if (IV-P1) is feasible, so here we assume the existence of an L2L stabilizing controller as given in Proposition 4.21. Then, the *next layer L2L finite stabilizing controller* problem is given as

$$\min_{\boldsymbol{u}\in\mathcal{U}}\{J(\boldsymbol{u}, \boldsymbol{x}^d_{k+1}) \mid f(\boldsymbol{x}_k, \boldsymbol{u}, k) \in \mathcal{S}_{k+1}\}, \tag{4.41}$$

where we assume J is a suitable loss function to penalize the distance between $f(\boldsymbol{x}_k, \boldsymbol{u}, k)$ and the desired spatial state $\boldsymbol{x}^d_{k+1}$. In general, we have a quadratic loss function, i.e. $J = ||f(\boldsymbol{x}_k, \boldsymbol{u}, k) - \boldsymbol{x}^d_{k+1}||_Q$, with Q as a symmetric and positive definite weight matrix. While J may include regularizers for the input $\boldsymbol{u}_k$ in its form, here we will focus on the case with

only the penalty term. If the system is L2L stable at layer k and $\boldsymbol{x}^d_{k+1} \in \mathcal{R}_1(\boldsymbol{x}_k)$, then (4.41) has a solution with the optimal cost $J^* = 0$. Next, we provide the more general *N-layer L2L finite stabilizing control* problem in the following.

$$\min_{\boldsymbol{u}\in\mathcal{U}^N} \{J(\boldsymbol{u}, \boldsymbol{x}^d_{k'+N}) \mid \phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k'+N) \in \mathcal{S}_{k'+N}\}, \tag{4.42}$$

where $J(\boldsymbol{u}, \boldsymbol{x}^d_{k'+N}) = \sum_{l=1}^{N} ||\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k'+l) - \boldsymbol{x}^d_{k'+N}||_Q$ and we apply the constraint $\phi(\boldsymbol{x}_k, \boldsymbol{u}; k+N) \in \mathcal{S}_{k+N}$ to ensure the system is within the L2L stable set $\mathcal{S}_N$ in N layers, even if $\boldsymbol{x}_{k'+N} \neq \boldsymbol{x}^d_{k'+N}$ at the optimum. If we know $\zeta(\boldsymbol{x}_{k_0})$ and set $N \geq \zeta(\boldsymbol{x}_{k_0})$, then (4.42) is feasible. Additionally, if (4.42) attains its minimum with $\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}^*; k'+N) = \boldsymbol{x}^d_{k'+N}$ then the system is L2L finite stabilizable.

Remark 4.22. *Note that (4.42) evaluates an L2L finite stabilizing controller even if we only take $J(\boldsymbol{u}, \boldsymbol{x}^d_{k'+N}) = ||\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k'+N) - \boldsymbol{x}^d_{k'+N}||_Q$ (instead of penalizing each state as given in (4.42)), and have $J^* = 0$ at the optimum. If we have $N \leq \zeta(\boldsymbol{x}_{k_0})$, this solution and the optimal solution of (4.42) coincide. However, if $N > \zeta(\boldsymbol{x}_{k_0})$ the resulting controller may be ill-posed in the sense that it does not necessarily steer the system states towards the reference spatial state. Thus, we utilize the form in (4.42) with the given cost function instead of penalizing only the final state.*

Another important case is when the system has exogenous additive noise $\nu \in \mathcal{W}$ for a convex and compact set $\mathcal{W}$ that contains the origin. So that the dynamics in (4.29) are now

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k, k) + \nu_k. \tag{4.43}$$

Due to the presence of bounded disturbances, the conditions in (4.35) no longer ensure the L2L stability of (4.43). To remedy this issue, we denote the contracted L2L stable sets

$$\bar{\boldsymbol{\mathcal{S}}} = \{\bar{\mathcal{S}}_k = \mathcal{S}_k \ominus \mathcal{W} \mid \mathcal{S}_k \in \boldsymbol{\mathcal{S}}\},$$

where we denote the Pontryagin set difference with $\ominus$ (i.e., for $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^n$, $A \ominus B = \{x \in A \mid x + y \in A, \forall y \in B\}$) and assume that the disturbance set $\mathcal{W}_k$ is nonempty and contains the origin. This choice of disturbance sets corresponds to bounded disturbances on the spatial process which are common in practice. If we are considering an analysis involving multiple layers in future, the effect of the disturbance over multiple layers must be accounted for such that the effect of set $\mathcal{W}$ is considered accordingly over the horizon, e.g., by computing a disturbance invariant set in the linear case [109]. Then, we use the L2L stable sets $\bar{\mathcal{S}}_k$ in place of $\mathcal{S}_k$ when we are dealing with the LSV of type

(4.43) instead of (4.29).

While L2L stability can be ensured by utilizing $\bar{\boldsymbol{\mathcal{S}}}$ in place of $\boldsymbol{\mathcal{S}}$ for the dynamics in (4.43), finite convergence may not be possible due to the presence of noise in the dynamics. Thus, convergence to a ball $\beta^2(\boldsymbol{x}_k^d, \epsilon),\ \epsilon > 0$ for small ϵ is desired for the dynamics in (4.43). Then, the N-layer L2L finite stabilizing control problem for the dynamics in (4.43) is solved by utilizing (4.40) with $\beta^2(\boldsymbol{x}_{k'+N}^d, \epsilon)$ in place of $\mathcal{S}_{k'+N}$, with ϵ suitably chosen based on the effect of set $\mathcal{W}$ over the horizon.

4.2.8 L2L Stabilizability Certificates

In this section, we summarize some of the results, and provide overall results on L2L stabilizability. While the results in the previous sections provide controllers that theoretically L2L stabilize the LSV system in (4.29) and (4.43), most practical LSV systems have finitely many layers, e.g., $k \in K = [1, n_\ell]$. Consequently, in practice, it is important to understand the feasibility of an L2L stabilizing controller. For (IV-P1), we characterize the L2L stability within K by checking if $k_0 + \zeta(\boldsymbol{x}_{k'}) \leq n_\ell$. We present the following results for characterizing L2L stabilizability of (4.29).

Lemma 4.23. *(N-layer L2L Stabilizability Certificate) The LSV system in (4.29) is L2L stabilizable in n' layers if and only if (4.37) has a feasible solution for $N = n'$.*

Proof. If the system is L2L stabilizable in n' layers, then (4.37) has to have a feasible solution due to Proposition 4.17. Similarly, if (4.37) has a feasible solution, then it implies that the system is L2L stabilized in at most n' layers with no feasible solution if the negation is true, which concludes the proof. □

Lemma 4.23 states a practical result that can be used for assessing L2L stabilizability of a system given its current state $\boldsymbol{x}_{k'}$, thus enabling predictive detection for L2L stability properties for an LSV system. Next, we provide a similar characterization for the L2L finite stabilizability.

Lemma 4.24. *(N-layer L2L Finite Stabilizability Certificate) The LSV system in (4.29) is L2L finite stabilizable in n' layers if and only if (4.42) has an optimal solution for $N = n'$ with the optimal input providing $\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}^*; k'+N) = \boldsymbol{x}_{k'+N}^d$*

Proof. Follows similarly to the proof of Lemma 4.23. □

As stated earlier, the results extend to the LSV dynamics with noise given in (4.43) by employing $\bar{\boldsymbol{\mathcal{S}}}$ in place of $\boldsymbol{\mathcal{S}}$. Utilizing Lemma 4.23 and Lemma 4.24, we evaluate *certificates of N-layer L2L stabilizability* for an LSV process at a given layer. After measuring $\boldsymbol{x}_{k'}$, we compute if the corresponding stabilizability problem has a feasible solution

in the remaining layers. Thus, if we have an N-layer L2L stabilizability certificate for $N = n_\ell - k'$ layers, where n_ℓ is the total number of layers in the process, we conclude that the process is L2L stabilizable before the end of the L2L process, otherwise, we conclude that the process is not L2L stabilizable before the end of the L2L process.

Remark 4.25. *Note that although a process may be L2L stabilizable in n' layers, if the process has a finite number of layers, e.g., an AM process, then L2L stabilization may not be feasible during the process. Therefore, N-layer stabilizability certificates provide insight on the expected performance of the closed-loop system in the L2L domain.*

This is an important result since it provides a certification for if the process is within the L2L stable tube $\boldsymbol{\mathcal{S}}$. To give a concrete example, if $\mathcal{S}_k$ defines the allowable dimensional tolerance for the spatial height map of up to layer k, the certificate of the process being the L2L stable tube would ensure if the resulting printed part is within the prescribed dimensional tolerances. If a process is not L2L stabilizable, a decision-maker may choose to stop the process to save time and material. Design of decision-making logics to act on the certificates of L2L stabilizability is beyond the scope of this chapter and is subject for future work.

Remark 4.26. *Note that if we use the tightened sets $\bar{\boldsymbol{\mathcal{S}}}$, then since we utilize approximations of the L2L stable sets, the results of Lemmas 4.23 and 4.24 are only sufficient conditions for L2L stability.*

4.2.9 Computation of L2L Stabilizing Controllers

In this section, we provide numerical solution methods for evaluating the L2L stabilizing controllers. The computational complexity evaluating the proposed controllers depends mainly on the complexity of the dynamics in (4.29).

We start our discussion with the most general case where (4.29) has no special structure, e.g. a nonlinear spatial dynamical system with layer-wise variations in the spatial domain and no additive property. In this case, no additional property for the optimizations for the L2L controllers can be inferred in general, although case-by-case specialized solutions may be possible. Thus, for the general case, sequential solution procedures for non-convex optimization such as sequential quadratic programming (SQP) may be employed for such LSV systems to solve the most general form in (4.40) as well as its L2L finite stable variant. Dimensionality of the domain $\boldsymbol{\Lambda}$ may become the limiting factor in this case as the problem size scales with the size and resolution of the spatial domain.

In the following subsection, we provide specialized solutions for two cases where the spatial dynamics in (4.29) have additional properties such as linearity and monotonicity.

4.2.9.1 Linear LSV (LLSV) Systems

In this section, we treat the case where the dynamics in (4.29) are linear. We start by investigating **L2L Stabilizability** (Problem (IV-P1)).

Since ϕ^N is a linear map, we can efficiently solve (4.42) by a proximal point iteration [153, 168]. First, note that (4.40) may be rewritten as

$$\min_{\boldsymbol{u}} g(\boldsymbol{u}) + \iota_{\mathcal{U}^N}(\boldsymbol{u}), \tag{4.44}$$

where $\iota_{\mathcal{U}^N}$ is a set-indicator function for the set $\mathcal{U}^N$ such that

$$\iota_{\mathcal{U}^N}(\boldsymbol{u}) = \begin{cases} 0 & \boldsymbol{u} \in \mathcal{U}^N \\ \infty & \text{otherwise,} \end{cases} \tag{4.45}$$

and the function $g(\boldsymbol{u})$ is given by (see (4.40))

$$g(\boldsymbol{u}) = \frac{1}{2}\sum_{l=1}^{N} d^2(\phi_l(\boldsymbol{u}), \mathcal{S}_{k'+N}),$$

where $\phi_l(\boldsymbol{u}) = \phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k'+l)$. Note that an input $\boldsymbol{u} \in \mathbb{R}^{n_u N}$ is a solution of (4.44) if and only if $\boldsymbol{u} \in \mathbf{zer}(\nabla g(\boldsymbol{u}) + \partial \iota_{\mathcal{U}^N}(\boldsymbol{u}))$. The gradient $\nabla g(\boldsymbol{u})$ is computed using the gradient of the distance function. For a linear mapping L and a nonempty convex set C, we have that

$$\frac{1}{2}\nabla d^2(Lx, C) = L^*(Lx - \Pi_C(Lx)),$$

where L^* is the adjoint of L, (see [54]). Furthermore, we have $\partial \iota_{\mathcal{U}^N}(u) = \mathcal{N}_{\mathcal{U}^N}(u)$. Then we use the following iteration to solve (4.44)

$$\boldsymbol{v}^{j+1} \leftarrow \Pi_{\mathcal{U}^N}\left(\boldsymbol{v}^j - \alpha \nabla g(\boldsymbol{v}^j)\right) \tag{4.46}$$

where we have $\alpha \in (0, 2/||\phi^N||^2)$ since $\nabla g(\boldsymbol{u})$ is 1-Lipschitz by [22], Corollary 12.30. The iteration (4.46) is known as a projected gradient algorithm in the literature since for each iteration of the algorithm, we take a gradient step and a projection step to ensure feasibility. Under the step size selection $\alpha \in (0, 2/||\phi^N||^2)$, the fixed point iteration (4.46) converges to its fixed point, given that one exists [40, 73, 168].

The fixed point solution $\boldsymbol{v}^*$ of (4.46) is an L2L stabilizing control input in N layers, only if $d(\phi(\boldsymbol{x}_{k'}, \boldsymbol{v}^*; k'+N), \mathcal{S}_{k'+N}) = 0$. It is important to note, however, that the fixed points of

(4.46) may result in nonzero objective value at the end of the horizon (e.g., $d(\phi(\boldsymbol{x}_{k'}, \boldsymbol{v}^*; k'+N), \mathcal{S}_{k'+N}) \neq 0$). These fixed point solutions denote the optimal control action toward an L2L stable set, since $g(\boldsymbol{u})$ is a cost for the distance of each state to the corresponding L2L stable set at the end of the horizon (e.g., $\mathcal{S}_{k'+N}$). This is an important result since we evaluate *the best control action that is not necessarily L2L stabilizing* in N layers, thus we are not required to know $\zeta(\boldsymbol{x}_0)$ a priori. In practical applications, this result is especially useful for receding horizon controllers where L2L stabilizability may not be feasible at the end of a given short horizon, but is possible given a sufficiently long horizon.

Next, we analyze the problem of **L2L Finite Stabilizability** (Problem (IV-P2)). We reformulate (4.42) for the linear spatial dynamics $\phi^N(\boldsymbol{u}) = \boldsymbol{\Phi}\boldsymbol{u} + \boldsymbol{\eta}$, where $\boldsymbol{\Phi}$ denotes the lifted linear dynamics, and $\boldsymbol{\eta}$ denotes the effect of initial conditions on the output. Note that due to the linearity of ϕ^N, the function

$$J(\boldsymbol{u}, \boldsymbol{x}^d_{k'+N}) = \sum_{l=1}^{N} ||\phi_l(\boldsymbol{u}) - \boldsymbol{x}^d_{k'+N}||^2_Q$$

is now a quadratic objective that is strictly convex. Similarly, the constraint $\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}; k' + N) \in \mathcal{S}_{k'+N}$ is now convex. In the following, we rearrange the terms in this optimization problem into a convex nonlinear program. In this setting, (4.42) becomes a quadratic program (QP) with the form

$$\min_{\boldsymbol{u}} \{\boldsymbol{u}^T \boldsymbol{P}\boldsymbol{u} + \boldsymbol{q}^T\boldsymbol{u} \mid \boldsymbol{G}\boldsymbol{u} = \boldsymbol{p}, \boldsymbol{p} \in P\}, \tag{4.47}$$

where $\boldsymbol{P}$ is symmetric and positive definite, $\boldsymbol{q} \in \mathbb{R}^{\bar{n}_u}, \boldsymbol{G} \in \mathbb{R}^{n_c \times \bar{n}_u}, \boldsymbol{p} \in \mathbb{R}^{n_c}, P \subseteq \mathbb{R}^{n_c}$, with a nonempty, closed and convex set P, and $\bar{n}_u = Nn_u$. Arranging the terms, we have $\boldsymbol{P} = \boldsymbol{\Phi}^T\boldsymbol{Q}\boldsymbol{\Phi}$ with $\boldsymbol{Q} = I_N \otimes Q$, $\boldsymbol{q} = 2\boldsymbol{\Phi}^T\boldsymbol{Q}(\boldsymbol{\eta} - \boldsymbol{r})$, with $\boldsymbol{r}$ defined in by

$$\boldsymbol{r} = \mathbf{1}_N \otimes \boldsymbol{x}^d_N. \tag{4.48}$$

Thus, we extend the desired reference at the end of N layers by a Kronecker product to penalize the distance between each state in the optimization horizon and the desired state $\boldsymbol{x}^d_N$. Additionally, we have

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{\Gamma}\boldsymbol{\Phi} \\ I_{\bar{n}_u} \end{bmatrix}, \quad P = (\mathcal{S}_{k'+N}(\boldsymbol{x}_0) \times \mathcal{U}^N),$$

where $\mathcal{S}_{k'+N}(\boldsymbol{x}_0) = \mathcal{S}_{k'+N} \ominus \boldsymbol{\eta}$ and we define

$$\boldsymbol{\Gamma} = [0_{n_x}, 0_{n_x}, \ldots, 0_{n_x}, I_{n_x}] \in \mathbb{R}^{n_x \times N n_x}, \tag{4.49}$$

as a selector matrix for the final state in the prediction horizon of the optimization. QPs with the form (4.47) can be solved efficiently by a variety of well-established methods in the literature including interior point methods [195] and proximal point methods [153]. Here, we present a solution approach using alternating direction method of multipliers (ADMM) with the form presented in [17, 179], and provide the resulting algorithm here for completeness. We rewrite (4.47) as

$$\begin{aligned} \min \quad & \tilde{\boldsymbol{u}}^T \boldsymbol{P} \tilde{\boldsymbol{u}} + \boldsymbol{q}^T \tilde{\boldsymbol{u}} + \iota_{\boldsymbol{G}\boldsymbol{u}=\boldsymbol{p}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{p}}) + \iota_P(\boldsymbol{p}) && \text{(4.50a)} \\ \text{s.t.:} \quad & (\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{p}}) = (\boldsymbol{u}, \boldsymbol{p}), && \text{(4.50b)} \end{aligned}$$

where $\tilde{\boldsymbol{u}} \in \mathbb{R}^{\bar{n}_u}, \tilde{\boldsymbol{p}} \in \mathbb{R}^{n_c}$ are auxiliary variables. The resulting ADMM iterates are

$$\begin{aligned} (\boldsymbol{u}^{j+1}, \tilde{\boldsymbol{p}}^{j+1}) &\leftarrow \underset{(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{p}}), \boldsymbol{G}\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{p}}}{\operatorname{argmin}} \tilde{\boldsymbol{u}}^T \boldsymbol{P} \tilde{\boldsymbol{u}} + \boldsymbol{q}^T \tilde{\boldsymbol{u}} + \frac{\sigma}{2} ||\tilde{\boldsymbol{u}} - \boldsymbol{u}^j||^2 + \frac{\rho}{2} ||\tilde{\boldsymbol{p}} - \boldsymbol{p}^j + \rho^{-1} \boldsymbol{y}^j||^2 \\ \boldsymbol{p}^{j+1} &\leftarrow \Pi_P(\tilde{\boldsymbol{p}}^{j+1} + \rho^{-1} \boldsymbol{y}^j) \\ \boldsymbol{y}^{j+1} &\leftarrow \boldsymbol{y}^j + \rho(\tilde{\boldsymbol{p}}^{j+1} - \boldsymbol{p}^{j+1}), \end{aligned} \tag{4.51}$$

which converges to a solution, given that one exists, for $\sigma > 0, \rho > 0$ [17, 179]. Here, we assume L2L stabilizability of the system in N layers, which makes (4.50a) a feasible problem. To detect when this assumption is valid, infeasibility detection methods such as the ones given in [17] may be employed to detect infeasibility of (4.51), which in turn provides a certificate of L2L stabilizability by Lemma 4.23. Additionally, we check if $\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}^*; k'+N) = \boldsymbol{x}^d_{k'+N}$ to provide a certificate of L2L finite stabilizability by using Lemma 4.24. Therefore, the resulting $\boldsymbol{u}^*$ of (4.51) is a solution of (4.47), and consequently (4.42).

We may relax the L2L stability constraint of (4.42) to evaluate a simpler reference tracking formulation, which is always feasible by assuming a reference spatial state $\boldsymbol{x}^d_k$ is admissible for the LSV system in (4.29). The relaxed problem of (4.42) is given by

$$\min_{\boldsymbol{u}} \{ J(\boldsymbol{u}, \boldsymbol{x}^d_{k'+N}) \mid \boldsymbol{u} \in \mathcal{U}^N \}, \tag{4.52}$$

which may be rewritten in the form of (4.44) with $g(\boldsymbol{u}) = \boldsymbol{u}^T \boldsymbol{P} \boldsymbol{u} + \boldsymbol{q}^T \boldsymbol{u}$, and then solved by (4.46). Thus, the iterates $\boldsymbol{u}^j$ converge to the minimizer $\boldsymbol{u}^*$ of (4.52). If in addition, we

have $\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}^*; k'+N) = \boldsymbol{x}^d_{k'+N}$, the control input $\boldsymbol{u}^*$ is an L2L finite stabilizing control in N layers, which is also an L2L stabilizing control input and thus, the solutions of (4.42) and (4.52) coincide. As the problem (4.52) denotes a general reference tracking problem through solving a QP, a variety of additional optimization methods from the literature may be used to evaluate a control $\boldsymbol{u}^*$.

Remark 4.27. *While (4.52) evaluates a control input to track the desired spatial state $\boldsymbol{x}^d_{k'+N}$, it is possible to optimize a control input $\boldsymbol{u}$ to minimize the tracking error $J(\phi, \boldsymbol{X}) = ||\phi(\boldsymbol{x}_{k'}, \boldsymbol{u}) - \{\boldsymbol{x}^d_{k'+i}\}_1^N||_Q$ for the desired spatial state trajectory $\boldsymbol{X}$, i.e., minimizing the distance to each desired state rather than the final one. For the case when ϕ is linear, the new tracking problem is again a QP and can be solved efficiently using the techniques introduced in this section and in the literature.*

4.2.9.2 Monotone Operator Admitting LSV Systems

In this section, we investigate LSV systems that are monotone operator admitting. This provides a specialized structure for the general nonlinear spatial dynamics, which may be exploited for improved computational efficiency.

An important note is that since LSV systems are spatial processes, we may measure the process outputs at each layer at the spatial locations $\boldsymbol{\lambda}$ that we apply the input on. Thus, we consider $\bar{n} = n_u = n_x, \bar{n} \in \mathbb{N}$ as the case where the state dimensions coincide with the input dimension.

We may utilize the monotone dynamics in (4.29) in their exact form to find a control that tracks a given reference, e.g. a desired spatial state in (IV-P2). Additionally, we assume the LSV dynamics in (4.29) are γ-strongly monotone. We call an operator F as *γ-strongly monotone* if $(x-x')^T(y-y') \geq \gamma||x-y||^2, \forall (x,y), (x',y') \in \mathbf{Gr}\, F$. This is not a restrictive assumption as it loosely corresponds to assuming no deadband zone exists in the dynamics, which is a practical assumption for many LSV systems, e.g. most AM spatial dynamics.

Namely, assuming a desired state $\boldsymbol{x}^d \in \mathbf{range}\, \phi(\boldsymbol{u})$ and the unconstrained minimizer of (4.41) is attained, the minimizer satisfies the following condition.

$$\boldsymbol{u}^* \in \mathbf{zer}((\phi - \boldsymbol{x}^d) + \mathcal{N}_{\mathcal{U}}), \tag{4.53}$$

where we assume the current state index is $k_0 = 0$ without loss of generality for later discussions. Additionally, since $\boldsymbol{x}^d_N \in \mathcal{S}_N$, we do not explicitly state the constraint $\phi^N \in \mathcal{S}_N$ in (4.41) due to the assumptions on feasibility. The solution of (4.53) is evaluated by

the fixed point of the following iteration

$$\boldsymbol{v}^{j+1} \leftarrow \Pi_{\mathcal{U}} \left(\boldsymbol{v}^j - \alpha(\phi(\boldsymbol{v}^j) - \boldsymbol{x}^d) \right), \tag{4.54}$$

where $\alpha \in (0, 2/\mathcal{L}_\phi)$, with $\mathcal{L}_\phi$ being the Lipschitz constant of ϕ, is the step size. The iterates of (4.54) have monotonic convergence under the given conditions, and the fixed point $\boldsymbol{v}^*$ is a solution of (4.53) [40, 73, 168]. Since $\phi(\boldsymbol{u})$ is γ-strongly monotone, (4.53) has a unique solution, i.e., $\mathbf{zer}((\phi - \boldsymbol{x}^d) + \mathcal{N}_{\mathcal{U}^N})$ is a singleton.

Remark 4.28. *The problem (4.54) evaluates a control input that steers the spatial state at a current layers to the desired spatial state in the next layer. Therefore the resulting control input can be interpreted as one that reaches the desired state as fast as possible (within the input constraints) due to the monotonicity and positivity of the dynamics.*

Furthermore, if the L2L spatial dynamics over multiple layers, i.e., ϕ^N is itself a monotone map, the equation (4.53) can be extended by defining an appropriate desired states trajectory over the future layers. The algorithm (4.54) for the resulting problem then evaluates a layer-to-layer finite stabilizing control input for the LSV system.

Remark 4.29. *The solution scheme here utilizing (4.54) evaluates the control input given in (4.41) (with the stable set constraints relaxed) for the exact form of the nonlinear LSV dynamics in (4.29) with the additive property. No approximations or linearization is required for the solution and no function inverses are required thanks to the algorithm employed in the solution and the monotonicity of the LSV dynamics.*

4.2.10 Simulation Studies in Additive Manufacturing

Here, we present two simulation examples to illustrate the utility of the L2L stabilizing control policy and the stabilizability certificates. We start by presenting results for L2L stabilizability using the single layer dynamics group linear model from Section 4.1.5. Then we illustrate a case where we utilize the nonlinear dynamics of the process to develop a next layer finite stabilizing controller for the process utilizing the methods presented in Section 4.2.9.2.

4.2.10.1 Layer-to-Layer Stabilizability

Here we have the linear dynamics of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + B\boldsymbol{u}_k + \nu_k, \tag{4.55}$$

where $\nu_k \in \mathcal{W}$ represents process noise with $\mathcal{W}$ as a compact noise set with $0 \in \mathcal{W}$. The linear input dynamics B corresponds to the single layer dynamics group model from Section 4.1.5. Since each dimension of the spatial state vector $\boldsymbol{x}_k$ represents the height of a spatial location along the deposition path, each corresponding row in B represents the effect of the inputs on the corresponding spatial location. If B is a diagonal matrix, we have that each input along the deposition path affects only the corresponding spatial location at the input. In the linear model considered in this case study, we additionally model the effect of the input at a spatial location to the previously deposited neighboring location by an additional term for each corresponding row excluding the first row. Therefore, we augment the model from Section 4.1.5 by adding off diagonal terms to represent the input at a spatial location affecting the spatial height of the neighboring point along the spatial deposition path. Therefore the matrix B has terms on its main diagonal and lower diagonal. Following a similar example to the one in Chapter III Section 3.5, we utilize the L2L stability to represent dimensional tolerances on the desired height of each layer in the process.

By denoting the desired state trajectory $\boldsymbol{x}_k^d$ at predefined incremental layer heights of $h_\ell = 350\mu m$ for each layer, we define the L2L stable sets as

$$\mathcal{S}_k = \{x \in \mathbb{R}^{n_x} | x \in [h_\ell k \pm 20]\}, \tag{4.56}$$

where we allow a 20 micron tolerance band for each dimension of the desired state $[\boldsymbol{x}_k^d]_i = h_\ell k$, where each dimension of the desired state is identically defined at a given layer by the desired layer height. Note that the size of the L2L stability region and tolerance margins depend on the specific process and design specifications of the printed part. Here, we utilize a conceptual example to illustrate the concepts.

In the simulation study we have the noise set $\mathcal{W}$ defined as a box around the origin with size of 10 micron in each direction,

$$\mathcal{W} = \{\nu \in \mathbb{R}^{n_x} | [\nu]_i \in [-5, 5], i = 1, \ldots, n_x\}. \tag{4.57}$$

Due to the presence of process noise we tighten the L2L stable sets by utilizing a set difference to get

$$\bar{\mathcal{S}}_k = \mathcal{S}_k \ominus \mathcal{W}, \tag{4.58}$$

which results in the tightened bounds of ± 15 microns around the desired state for the next layer. Note that if we evaluate an L2L stabilizing controller over a horizon of layers, the set difference must be appropriately adjusted to account for the effect of the disturbance over

the horizon of the process. For example if we look at the worst case effect of the bounded noise over a horizon of two layers, the tightened L2L stable set would be tightened by twice the effect of the noise, resulting in tightened bounds of ± 10 microns. For the linear dynamics, it is also possible to evaluate the disturbance invariant set for L2L stable set constraint tightening, utilizing well established methods from the literature [109].

Our goal is to design a controller such that the spatial states $\boldsymbol{x}_k \in \bar{\mathcal{S}}_k$. For the physical process this corresponds to the spatial heightmap of each layer conforming to the design tolerance specifications in the presence of process noise. Therefore, L2L stabilizing control provides us with a practical control strategy to ensure part quality and functionality.

We first present a nominal L2L stabilizing control policy using (4.44) with (4.46). Given that the process is L2L stable at the current layer, we utilize (4.46) to evaluate an L2L stabilizing control action for the next layer. Therefore, at each layer, we get the spatial state measurement, e.g. utilizing a setup similar to the one in Chapter III. Then we use the next layer tightened stable set from (4.58) to compute (4.44).

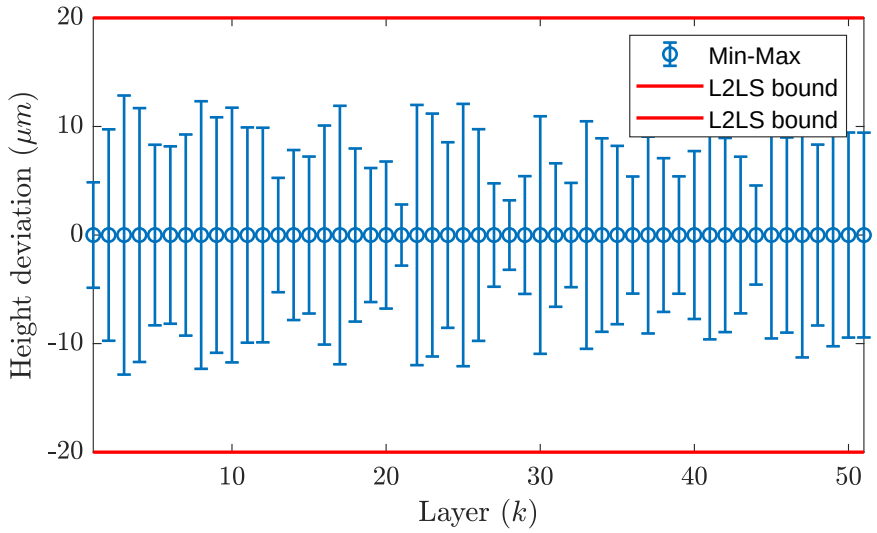


Figure 4.7: Results of the simulation for the next layer L2L stabilizing controller with the tightened L2L stability bounds for process noise. The layer-to-layer controller successfully retains the process within the L2L stability limits defined by the ± 20 microns.

Figure 4.7 demonstrates the results of the simulation of the nominal L2L stabilizing control policy controlling the process (4.55) over 50 layers. The controller uses the state information and the noise bounds to evaluate a control action that L2L stabilizes the process in the context of process noise. The plot shows the maximum deviation of the spatial height at each layer from the desired spatial state $\boldsymbol{x}_k^d$. Therefore the min and max in Fig. 4.7 denote the minimum and maximum deviation of the spatial height at that layer from the desired

state $\boldsymbol{x}_k^d$ over the spatial domain. The L2L stability bounds for each layer are defined as a maximum of 20 micron deviation given in (4.56). The control bounds are defined as box constraints, i.e., $\boldsymbol{u}_k \in \mathcal{U} = [0, u^{\max}]^{n_u}$, with $u^{\max}$ denoting the maximum allowable control input for each channel. The input constraint set for the control is suitably chosen such that the reachability condition for the L2L process is satisfied with the tightened L2L stable sets. The results show that the controller successfully retains the process within the L2L sets at each layer in the presence of process noise.

Remark 4.30. *By performing the L2L stable set tightening procedure explained above, we obtain L2L stabilizability certificates for a given current spatial state $\boldsymbol{x}_k$ and a horizon of N layers for the simulation study. An important note is that due to the set tightening procedure, the resulting certificates are related to the worst case results and may be overly conservative in practical situations. In low process noise scenarios where ν_k is negligible, the L2L stabilizability certificates using this procedure become less conservative.*

4.2.10.2 Layer-to-Layer Finite Stabilizability

Next, we illustrate an example where the layer-wise spatial dynamics are nonlinear and monotone. Here we adopt a simplified form of the bead cross section model developed in the coauthored work presented in [2] and denote the height dynamics as a function

$$[\boldsymbol{x}_{k+1}]_i = [\boldsymbol{x}_k]_i + c_1 \left(h_\ell + \sqrt{c_2 \max\{[\boldsymbol{u}_k]_i, 0\}^2 + h_\ell^2} \right), \tag{4.59}$$

where we take $h_\ell = 250\mu m$, and c_1, c_2 are model constants. The equation (4.59) is monotone in the input $[\boldsymbol{u}_k]_i$ which may be easily verified by checking (4.31). Additionally the subgradient of (4.59) is bounded by $c_1\sqrt{c_2}$. Since the spatial dynamics of each index i corresponding to a spatial location on the deposition path is defined by (4.59), the layer-wise spatial dynamics is the concatenation of monotone operators, which results in a monotone operator itself. Additionally, the dynamics is strongly monotone in the positive range of the inputs, which is where the control constraints set $\mathcal{U}$ lies. By assuming the parameters c_1, c_2 are identical for all i, we have the Lipschitz constant $\mathcal{L}_\phi = c_1\sqrt{c_2}$. Therefore, we can utilize (4.54) to evaluate next layer finite stabilizing control action for the LSV process.

Figure 4.8 shows the maximum deviation of the spatial trajectories of the LSV process from the desired spatial state for multiple runs starting from random initial conditions within one step finite backward reachability of the desired spatial state of layer 2. By utilizing (4.54) at each layer we compute the next layer finite stabilizing controller for the process, given the current state of the process and the control constraints. In Fig. 4.8 we see

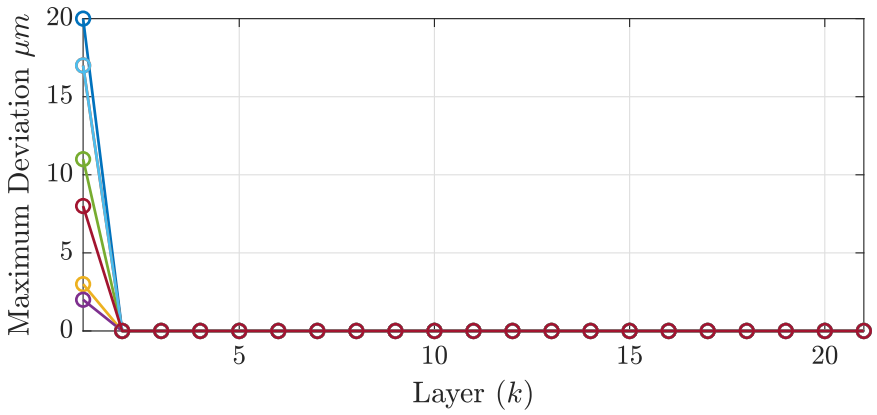


Figure 4.8: Simulation results for the next layer finite stabilizing control with the monotone LSV dynamics. The plot shows the simulation results for multiple runs, where the height of the initial layer is randomly changed subject to next layer reachability constraints.

that the controller successfully tracks the desired spatial state in the next layer (i.e., layer 2) and tracks the desired spatial state for the remaining layers. It is important to note that we evaluate the finite stabilizing control action for the next layer by utilizing the monotonicity of the dynamics and the solution of the monotone inclusion problem (4.53).

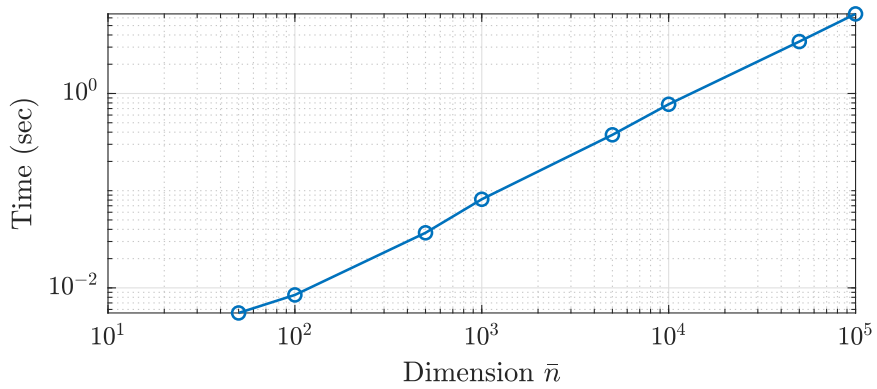


Figure 4.9: Time complexity of the algorithm (4.54) in the case study as a function of the increased spatial dimensionality.

Since the algorithm (4.54) utilizes the monotone dynamics and simple box constraint projections, it is computationally efficient when compared to the methods involving inversion of the dynamics. We illustrate the computational efficiency of (4.54) in Fig. 4.9, where we time the computation of the control action per layer in a 20 layer simulation while increasing the size of the spatial domain $\bar{n}$. For each layer, the computation of the control action is limited to 1000 iterations, which is experimentally verified to be adequate for convergence of the algorithm (4.54). Figure 4.9 shows the time complexity as a function of

spatial dimension, which is promising for utilizing the proposed method on large scale AM systems. We see that for larger scales such as 10^5, our preliminary implementation takes around 7 seconds to evaluate a control action utilizing the nonlinear monotone LSV dynamics. Further scale and verification studies must be conducted to extend the preliminary results and provide computational guarantees of the proposed method.

4.3 Chapter Conclusions

This chapter presents two main sections on the control of spatial AM processes. The main contribution of this chapter is the introduction of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints (*C2*). The ILC method presented in Section 4.1 is a novel approach to iteratively learn a control input that tracks a given layer-to-layer reference. The approach considers spatial processes that are spatially varying, subject to control inputs over the spatial domain. The use of process data in conjunction with the model information for the layers where sufficient data is not available provides an efficient hybrid approach where data and model information are utilized together. Due to its simplicity, the proposed method is suitable for implementation on real processes that have the required sensor setup and control authority available.

An immediate research question extending from the results of Section 4.1 is if we can utilize data from previous runs or similar machines to improve the performance of the model gradient. We provide further remarks and applications about this application in Chapter V, where we develop a centralized control architecture for an AM Fleet as an example of a spatially distributed system.

Section 4.2 presents a new framework to ensure layer-to-layer stabilizability of an LSV process. By utilizing layer-to-layer stability metrics from Chapter III, we build layer-to-layer stable sets to characterize the desired system performance and develop controllers to ensure desired performance with respect to layer-to-layer stability of the spatial dynamics. Additionally, we utilize structures such as monotonicity of the spatial height evolution of an AM process to propose an efficient algorithm that can efficiently compute next layer finite stabilizing controllers. The results of Section 4.2 are especially important in the context of higher-level decision makers. Depending on the L2L stabilizability of a process, it may be desirable to stop the print and rerun the process from the beginning on a different machine, since L2L stability can be attributed to the design tolerances and specifications. These analysis methods provide novel performance analysis and anomaly detection methods for industrial AM processes and have further extensions to other spatial processes.

While the chapters so far have dealt with spatial processes at the process level, as mentioned in the introduction, AM processes are often utilized in the form of a fleet in industry. Therefore, we are interested in investigating the modeling and control applications in the context of system-level spatially distributed systems as an extension of the work presented so far. In the next chapter, we investigate AM Fleets as an example of spatially distributed systems. We present a framework for system-level modeling, analysis, and control.

CHAPTER V

System-Level Modeling and Control of Spatially Distributed Additive Manufacturing Fleets

This chapter focuses on spatially distributed systems and presents modeling and control methods for additive manufacturing fleets (AM Fleets). The main contribution of this chapter is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets (*C3*). We present a system-level centralized control framework for AM Fleets and provide details of its components. The framework in Section 5.1 has multiple components for run-time data collection, modeling, analysis, and control of the resources in an AM Fleet. Then, the following two sections (Sections 5.2-5.3) focus on two important practical applications that can be implemented utilizing the proposed framework. Section 5.2 presents a closed-loop model predictive control strategy for run-time scheduling in AM Fleets. The presented controller is capable of working with timed automata models to ensure constraint satisfaction and disturbance rejection due to the incorporation of specialized constraint structures and its receding horizon nature. Such run-time control problems at a system level are crucial for AM Fleets as customized production orders require bespoke production schedules in contrast to serial mass production systems. Lastly, in Section 5.3, we demonstrate a knowledge transferring control application enabled by a centralized controller. We present a formulation and an extension of the layer-to-layer control methods presented in Chapter IV to demonstrate the utility and advantage of building a knowledge base to transfer knowledge between resources in an AM Fleet. Accordingly, the first section presents a system-level centralized framework architecture, the second section utilizes this framework to present a controller utilized for closed-loop production scheduling, and the third section utilizes the centralized framework to illustrate knowledge transferring control examples in simulation. The contents of this chapter are also partially presented in [10, 14].

5.1 A Centralized Control Framework for AM Fleets

We start by introducing a system-level centralized framework for modeling and control of a set of AM processes as a spatially distributed system. This section introduces the framework and its components, and illustrates conceptual examples of how the system-level decision-making takes place in practice. The remaining two sections in this chapter (Sections 5.1-5.2) utilize the centralized framework to present specific applications on run-time closed-loop production scheduling and knowledge transferring control.

AM is an important enabler of smart manufacturing due to its flexibility and agility; however, the adoption of AM in industry has been limited due to the in-process, run-to-run (R2R) and machine-to-machine variabilities that affect reliability and cost of production. Currently, the variabilities in AM systems are not well understood and best practices for modeling and sensing are not yet standardized for many AM processes, representing an important research challenge [74, 97, 155]. Another important drawback is the long processing time of the build process. To achieve higher volume production, manufacturers utilize a set of AM machines (AM-Fleet) in parallel. This system of AM machines often consists of machines from various vendors and may include different AM processes. Despite the increasing use of multiple AM machines in practice, system-level control of this set of AM machines is heuristically regulated based on operator expertise and process variabilities, which are especially nuanced in fleet settings with multiple machines. There are numerous factors affecting the build process ranging from the cyber aspects such as the 3D model, slicing parameters and in-layer parameters, to the physical factors such as material type, temperature and deposition rate [171]. Additionally, due to the lack of system-level anomaly detection (AD) and quality assurance modules, the AM machines in the set are often prone to faults that may go several hours before detection, resulting in significant cost and time losses [155]. To realize high volume production using AM technologies, formal solutions to system-level modeling and control of AM-Fleets must be developed. To address this problem, we propose a framework that can be used to model normative behavior of AM machines and analyze momentary machine behavior for intelligent decision making in AM-Fleets. The objective of the system-level control is to schedule the requested production jobs, adjust the AM machine parameters for the given jobs, and monitor the machines in the fleet to optimize the system-level metrics of the AM-Fleet. Two important system-level metrics for the AM-Fleet are throughput (high reliability) and quality (high repeatability) of the fleet.

The *main contribution of this chapter* is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling con-

trol and knowledge transfer/reuse in AM Fleets (*C3*). Within the contributions of this chapter, the specific contributions of this section include [14]:

(*C3*-1) Identification of key problems for a system-level AM-Fleet control framework.

(*C3*-2) A proposed framework for system-level control of multiple AM machines.

(*C3*-3) A high-level discrete event model to be used in monitoring and control of AM machines.

As a result of the increased interest in industrial use, literature references for modeling and process optimization for FDM have seen a significant increase in numbers [134]. Recent work in [169] introduced the concept of functional states for monitoring the working conditions and events of manufacturing machinery. Similarly, [75] proposed the use of semantics-based AM process specifications and described a framework for process control with in-situ measurements. Their proposed methodology provides a high-level view of the semantics-based ideas without elaborating on the practical use of the concepts or requirements for implementation on a system. [128] proposes an analytical digital framework for collaborative additive manufacturing with information models and ontologies for AM. Proposed models and ontologies can be leveraged for reusable data and model management in AM-Fleets.

Scaling from a single machine to an AM-Fleet involves many challenges such as monitoring the states of the individual machines to make system-wide decisions and detecting faults and anomalies that occur in the fleet. Making system-level decisions rather than optimizing each machine in isolation is shown to be effective in the literature [105, 126]. While these are well studied challenges in the general manufacturing systems literature, they remain as open challenges in the AM literature. Centralized approaches are adopted in manufacturing systems to increase the visibility into the system and enable decision making for the entire system.

The semiconductor manufacturing industry has adopted centralized system-level controllers for improving system productivity by reducing R2R variabilities in the machines [137]. AM and semiconductor manufacturing are both layer-wise manufacturing processes; however, there is a rich literature in system-level process control for semiconductor manufacturing, while system-level control for AM fleets is a relatively young field. Thus, the framework proposed in this section leverages the existing literature in semiconductor manufacturing to address some of the challenges in AM. For example, virtual metrology (VM) is the prediction of product parameters using in-situ measurements of related variables. VM is well established and adopted in semiconductor manufacturing for R2R feedback control [105]. An important challenge in the adoption of VM methods for AM is the lack of

in-situ measurement and sensing methods for AM [155]. To achieve system-level control, the integration of in-situ measurements in multiple AM machines and an understanding of AM system behavior must be explored.

The use of multiple AM machines in a given environment is not a new concept. In [136], 22 selective laser-melting machines are simulated and the effect of build time and operator cost on the throughput of the system is investigated. For the simulation environment used in [136], each machine was modeled as an isolated tool. As such, the interactions between machines as well as the development of system-level scheduling and control policies were not addressed. Knapp et al. proposed In [107], building blocks for the digital twin of laser-based AM processes are proposed and some empirical results are presented. However, the proposed models are process specific and usually limited to the thermal characteristics of the build material. The modeling of system-level functional interactions or the development of system-level decision-making methods are not provided. The integration of multiple AM machines and the analysis of their interfacing and behavior as a system remains as a significant technical challenge.

AM process identification is labor intensive and the correlations between different process parameters that affect system performance are not easily established [134, 155]. Modeling efforts usually capture very specific design-process-material combinations and there is no ability to transfer these models and controllers between different machines. Rao et al. proposed a multi-sensor framework for real-time process measurements in an FDM process [160]. Although their classification algorithm performs well in real-time, it needs labor-intensive setup experiments to be applied on other AM machines. Additionally, faults occurring in the processes are not logged in a knowledge base to be used in future anomaly detection and decision making efforts. Tracking the encountered anomalies in the AM-Fleet will help a decision maker to identify a re-occurrence of the same anomaly. Normal and anomalous states of the FDM process are modeled in [197]. State identification techniques using acoustic emission data with principal component analysis are shown to improve the monitoring system performance.

Prognostics and health management methods to detect faults in an FDM process is presented in [204]. Despite demonstrating promising results, the methodology is focused on a specific machine-fault setup and does not build a knowledge base, or scale to multiple machines. It is also important to note that these approaches are reactive to the errors detected in the process and do not consider predictive approaches to identify the early onset of faults. To truly address the control of an AM-Fleet, a robust cyber-physical AM framework that considers system-level models, knowledge-based models, and in-situ measurements to enable predictive actions must be developed.

The rest of the section is structured as follows. Subsection 5.1.1 identifies key problems to be solved by a centralized system-level controller for AM-Fleets. A discussion on the opportunities and challenges for a centralized approach is also given. Subsection 5.1.2 presents the proposed framework architecture based on a software defined control approach introduced in [126]. Subsection 5.1.3 introduces a functional state-based model of AM machines to support system-level decision making in the proposed framework. Subsection 5.1.4 presents an illustrative anomaly detection example for the framework.

5.1.1 Centralized Control Approach for AM-Fleets

This section proposes a centralized approach for system-level control of AM-Fleets. The proposed framework is scalable through the use of machine data protocols and digital twins, and able to model functional aspects of AM and perform on-line measurements for anomaly detection. The framework also supports predictive decision making on the system-level parameters of the AM-Fleet.

System-level control involves analysis of the behavioral system models using an abstracted global view of the system to detect trends and anomalies. Based on this analysis, the system-level controller applies control actions through the original equipment manufacturer (OEM) designed proprietary low-level controllers of the AM machines. To accomplish system-level control of an AM-Fleet, four key problems must be solved:

1. Integration of sensors (both in-situ and post-process) that transfer data from the AM machines to the centralized controller

2. Development of decision-making strategies that do not require access to proprietary controllers

3. Development of hybrid models (continuous/discrete, physics-based/data-driven) of AM processes for use in data analytics and predictive performance analysis

4. Development of a knowledge base (library) for system-level modeling, intelligent decision making, and knowledge transfer between machines and processes

Advantages of a hierarchical and centralized framework for the AM-Fleet and the challenges associated with the realization of the framework are discussed in this section.

5.1.1.1 Advantages of a Centralized Approach

Centralized approaches are adopted in manufacturing systems to make system-level decisions and optimize system metrics such as throughput, yield, and cycle time. Hierar-

chical structures are utilized in centralized approaches for the efficient communication and management of the data.

Optimization of Process Parameters and Schedule: The proposed centralized framework collects in-situ and post-process measurement data from the machines in the fleet, which can be used for optimizing process parameters of individual machines. AM research has been focused on analyzing AM processes in isolation, whereas in the centralized setting, data from multiple machines can be used to analyze the AM process and design optimization algorithms for improved process performance. Optimization may occur at the machine level for feed-forward parameter tuning based on the process knowledge, or it may be focused at the system-level to optimize the AM-Fleet metrics for process time and printing quality across the fleet.

Scheduling for the AM-Fleet is another important aspect of a centralized approach. Scheduling tasks may include off-line scheduling for optimized throughput, as well as dynamic re-scheduling to address anomalies within the system or the introduction of new high-priority build jobs.

Data Analysis and Predictive Capabilities: Process monitoring has always presented a great challenge in AM processes. Despite several *proof-of-concept* methods presented in literature, standardized measurement protocols for AM systems have yet to be developed [155]. The centralized framework can accommodate heterogeneous sensor networks, as proposed in [160], for in-situ measurements in the AM-Fleet. The measurement can be used for predicting anomalies in the AM machines to enable predictive maintenance events and re-schedule the affected parts to other machines. Using the past-job data in the knowledge base, anomalies that progress over many machine runs can be identified and the machine health can be monitored.

Knowledge Transfer in the AM-Fleet: Using existing machine learning techniques, correlations between the process parameters for each machine can be estimated by the central controller and stored in the knowledge base. Using *transfer learning* techniques, the framework can study the similarities and dissimilarities between the machines in the fleet. A statistical transfer learning problem in AM is presented in [42]. While their approach focuses on the estimation of errors in a new geometry, the transfer task in the proposed framework can also transfer the process knowledge between the machines in the AM-Fleet. Statistical models could be developed by many control applications such as [94, 162]. The knowledge transfer capabilities can improve the performance of the optimization and prediction applications.

5.1.1.2 Challenges and Issues in a Centralized Approach

Some of the potential challenges that must be addressed by a centralized approach are discussed below.

Security and Network Issues: There is recent work focusing on the cyber-security of the AM processes [145, 198]. The central controller must have a secure infrastructure to prevent the data leakage through the side channels of AM machines [44]. Security must be assured in both physical and cyber domains. We present a detailed treatment of cyber-security for cyber-physical manufacturing systems later in Chapter VI. Additionally, since there is no standardized machine communication protocol or network type for AM machines, an existing communication protocol must be adopted for the use of AM systems. Since the monitoring of the central controller may include real-time data communications at high sampling rates, the network and the communication protocols must be able to accommodate a high bandwidth.

Data Processing and Management: As the fleet scales up, data management becomes an important issue. The complexity of the optimization problem to be solved for the AM-Fleet increases with the number of machines. While the hierarchy in the centralized approach alleviates some scaling effects, efficient algorithms must be developed for the computations of optimization solutions in this complex environment. The abstractions of the AM machines should be sufficient to represent the physical system and enable optimization.

AM processes have a long timespan and the amount of streaming data that must be processed during a single printing job may be too large to be stored on a local database. The knowledge base is tasked with storing the machine and measurement data for further analysis. As the size of the data increases, cloud based storage solutions can be considered. A comprehensive data pipeline that stores recent machine data in a local database and pushes historical data to the cloud for later analysis must be developed for an efficient data management schema in practical large-scale industrial implementations.

Limited Access to Machine Parameters: Due to the limitations posed by the proprietary OEM controllers, not all the machine parameters are accessible to the central controller. The centralized approach should be able to identify the controllable parameters for the AM machines and communicate the control actions to be executed through the OEM controllers.

The control algorithms should be developed according to the accessibility constraint. Designing control algorithms for limited control authorization is an important challenge in the AM literature due to the lack of standardized proprietary controller architectures for AM machines [97]. Since different OEMs have different control architectures, the centralized

controller should be communicating through a data format that is compatible with all the OEM controllers in the fleet.

5.1.2 SDC-AM Framework for AM-Fleets

Software-defined approaches are widely used in networks where the centralized controller utilizes abstract representations of the network nodes [112]. Software defined control (SDC) is an approach for the hierarchical modeling and control of smart manufacturing systems [126]. SDC uses both control and enterprise data to provide a global view of the system and allows the central controller to efficiently evaluate reconfiguration recommendations. The architecture of the proposed framework for the system-level control of AM-Fleets is based on the SDC architecture [126].

Figure 5.1 presents the architecture of the proposed framework. The framework consists of a central controller that communicates with the AM-Fleet through the *Data Input/Output Interface* (DIOI). The central controller houses a knowledge base to store the process parameter correlations for each machine, estimated by machine learning techniques, digital twins that include models of AM machines and the AM-Fleet, a decision maker to process the measurement data and make system-level decisions, and a library of services to be utilized by the decision maker or external applications. The services and the data in the central controller can be used by external applications through the *Application Programming Interface* (API). The AM-Fleet control framework proposed in this chapter is referred to as SDC-AM.

SDC central controller in [126] has the southbound and the northbound interfaces that will be utilized in SDC-AM. DIOI can be implemented using the southbound interface to collect data from the plant floor. In SDC-AM, DIOI is used for both input and output of data streams. SDC-AM has authority over the AM-Fleet and can implement control actions. The API of the SDC-AM and the external applications can be implemented on the northbound interface as shown in Fig. 5.1.

Using the integration of the information technology (IT) and operational technology (OT) data in the enterprise, performance-aware optimizations mentioned in Section 5.1.1 can be realized. This gives the framework a global view of the manufacturing enterprise, which has been shown to improve system-level decision making [126].

5.1.2.1 Data Input/Output Interface

The DIOI collects data from the machines in the AM-Fleet and pre-processes the data before it is used in the central controller. Pre-processing is a step that is often necessary for

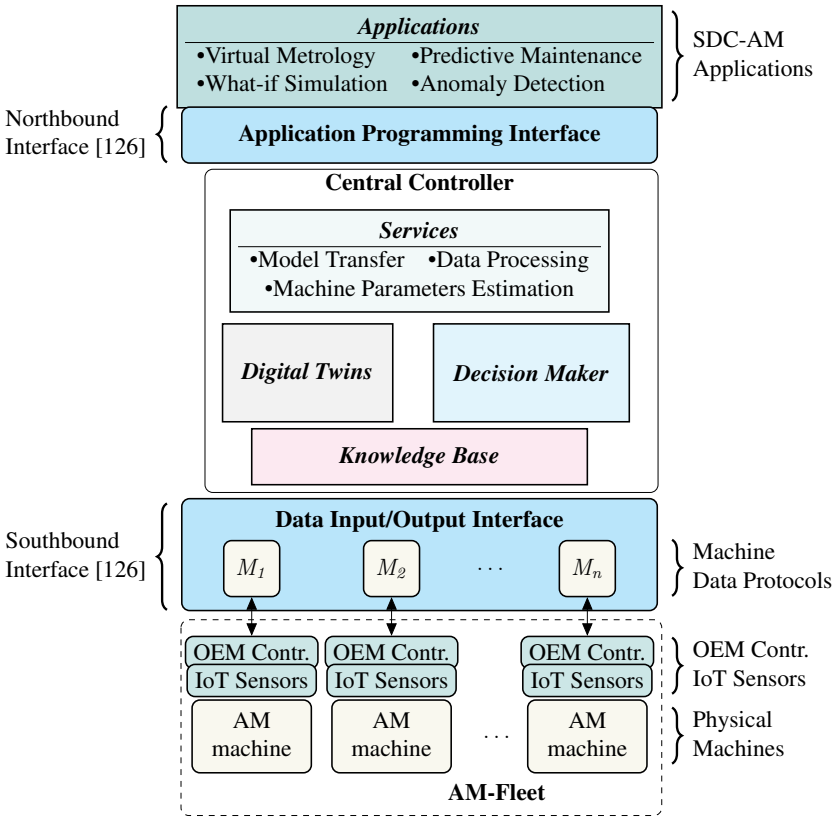


Figure 5.1: The SDC-AM controller for the AM-Fleet.

raw data such as analog sensor readings, or classification of time series data. AM systems use flexible controller architectures such as personal computers and programmable microcontrollers. This creates a rich environment to access the data in the machine controllers using Internet of things (IoT) applications.

Additionally, the DIOI communicates the control actions and decisions of the central controller to the AM-Fleet. Control actions may include adjustments to the process parameters or modifications to the cyber data, such as a modified design. The DIOI also distributes the Computer Aided Design (CAD) files that are stored in the knowledge base for the individual machines and acts as a gateway between the central controller and the AM-Fleet. Abstractions of the individual AM machines in the fleet are shown with $M_1, M_2, \ldots, M_n$ in Fig. 5.1. The DIOI uses these abstractions to monitor individual machines in the AM-Fleet, and manage the communication between the AM-Fleet and the central controller.

5.1.2.2 AM-Fleet

The AM-Fleet may consist of machines using various AM processes from different vendors. Additionally, an AM Fleet includes additional mobile manipulators, robots, automated guided vehicles (AGVs), post processing stations, and other necessary resources for specific application domains. Data collection and communication is through the machine data protocols for monitoring the machines in the fleet and available control actions for the machines in compliance with the proprietary controllers. Through standardization, the machine data protocols layer in the data interface provides a scalable and structured method for fleets with large number of resources. The machine data protocols and their level of detail is chosen according to the specific application that will use the abstraction. The SDC-AM framework uses run-time data from the plant floor supplied through the machine data protocols in the communication interface for analysis and control. The data should capture sufficient information to be used by predictive applications for anomaly detection, maintenance, and control. The run-time data is further used within the digital twins to provide a run-time model of the AM Fleet for various purposes.

5.1.2.3 Central Controller

Using the physics-based and data-driven models of the AM-machines, the central controller makes decisions to improve the AM-Fleet metrics ($e.g.$ throughput, quality, cycle time). Elements within the central controller provide the enabling infrastructure to address the problems identified in Section 5.1.1.

Decision Maker: The decision-maker is defined as a software module that houses the system-level control and intelligent decision making algorithms for the AM-Fleet. Using the data from the physical fleet, the decision maker determines how to change the system-level and machine parameters to improve the AM-Fleet metrics. When a new production request is submitted to the AM-Fleet, the decision maker uses system knowledge from the knowledge base to derive scheduling routes that achieve the desired quality and throughput in the specified time. The decision maker uses multiple algorithms to determine the scheduling and dispatch and maintenance events for the AM-Fleet. Machine and system models in the knowledge base are used for machine-level and system-level parameter selection. Additionally, the decision maker uses abstractions of the AM-Fleet to observe the process in each machine. A detailed architecture of the decision maker and the details of the fundamental algorithms to be implemented for prediction, optimization, and control is a subject for future work.

Knowledge Base: As the DIOI forwards the machine data to the central controller, the

machine data is first stored in the knowledge base. The knowledge base includes a database to store the data coming from DIOI. In addition to the measurement data, knowledge base stores abstractions, parameter correlation estimations, and physics based system dynamics models for the AM machines.

Other components of the central controller can access the data in the knowledge base database. The type of data stored in the knowledge base is classified in terms of machine-specific data and fleet data. Each machine has a profile in the knowledge base where the physics-based models of the machine, past jobs, in-situ and post-process measurements, controller models, abstractions and parameter correlation estimations specific to the machine are stored. Fleet data include the monitored system-level metrics, filters and estimators for system behavior prediction, production plans, schedules, CAD models, and production requests.

Digital Twins: The central controller houses digital twins of the AM Fleet. A digital twin is a software replica of a physical thing (i.e., the physical twin) and has the purpose of impacting an aspect of the physical twin and its environment in a positive way through utilizing models, data analytics, and subject matter expertise (SME) [138]. Within this context, digital twins of the physical processes and components as well as cyber processes (e.g., digital twins of anomaly detection systems) are utilized by the decision maker in the central controller for run-time decision-making. As opposed to static models of processes and components in the fleet, digital twins provide a run-time up-to-date representation of the AM fleet to improve decision-making. Simulation models are used by the decision maker to make predictions about the AM-Fleet and both the machine and the AM-Fleet models are updated with streaming data from the AM-Fleet. Increasing data visibility in production systems enables the use of digital twins with manufacturing systems.

Digital twins use multiple data-driven and physics based models to simulate the physical system with the real-time measurement data. Digital twins are updated using live streaming data in order to enable high-fidelity simulations of the physical system to gain insight about the process dynamics of the system. Digital twins in SDC-AM provide a simulation infrastructure for predictive decision making by using the streaming data from the DIOI and the machine knowledge from the knowledge base. Physics-based models of the AM processes may be complex and computationally expensive. The central controller uses the abstraction models to abstract the in-situ data and uses the digital twins to monitor the machine state in order to make decisions about machine or fleet parameters. We provide a detailed analysis of DT-based monitoring and detection methods in Chapter VI, and provide further implementation examples for AM processes in Appendix C. In this chapter, we utilize DTs as run-time monitoring and analysis tools to improve the visibility

of the AM Fleet for central controller and the decision maker. We illustrate how digital twins can be used to enable run-time closed-loop scheduling applications in an AM Fleet in Section 5.2.

Services: Services are a list of functions that make up the integral capabilities of the central controller. These capabilities may be requested by the digital twins, decision maker or external applications through the API. An example service is data processing that houses the machine learning and classification tools. Machine parameter estimation service uses the data processing service to estimate the process parameter correlations for the specific AM machines. Machine parameter estimation is used in control applications to determine optimal machine parameters for a desired output. Data processing service is utilized in identifying statistical process similarities between the AM machines in the fleet. The model transfer service is another service that uses the statistical process similarities to solve transfer learning problems. Transfer learning enables the use of the statistical models from one machine on the other machines in the AM-Fleet. Therefore, the model transfer service enables the re-use of the knowledge between the machines to improve the modeling efficiency.

The central controller (1) uses DIOI to collect data from the integrated sensors and utilizes the machine data protocols to establish a global view of the AM-Fleet, (2) has a decision maker to evaluate decisions which can be communicated through the DIOI and executed without requiring access to the proprietary controllers, (3) simulates the AM-Fleet using the digital twins to make predictive analysis about the system-level performance, and (4) utilizes the process knowledge across the AM-Fleet using the services (data processing and model transfer) and the machine data.

5.1.2.4 API and Applications

API provides an interface to externally access the data in the knowledge base, digital twins, and the services. Access through the API requires authentication for security purposes and the access may be restricted for some applications. Applications enable the integration of externally developed system analysis and control tools with the central controller for the control and management of the AM-Fleet. VM applications can be integrated in the proposed framework for R2R process control, what-if scenarios could be simulated using the digital twins to predict the response of the system under certain decisions or process parameters, and predictive maintenance scheduling may be performed by analyzing the machine data using the data processing service to detect degradations in the machines.

Anomaly detection is a fundamental application for the system-level controllers of smart manufacturing systems [125]. Different anomaly detection algorithms can be de-

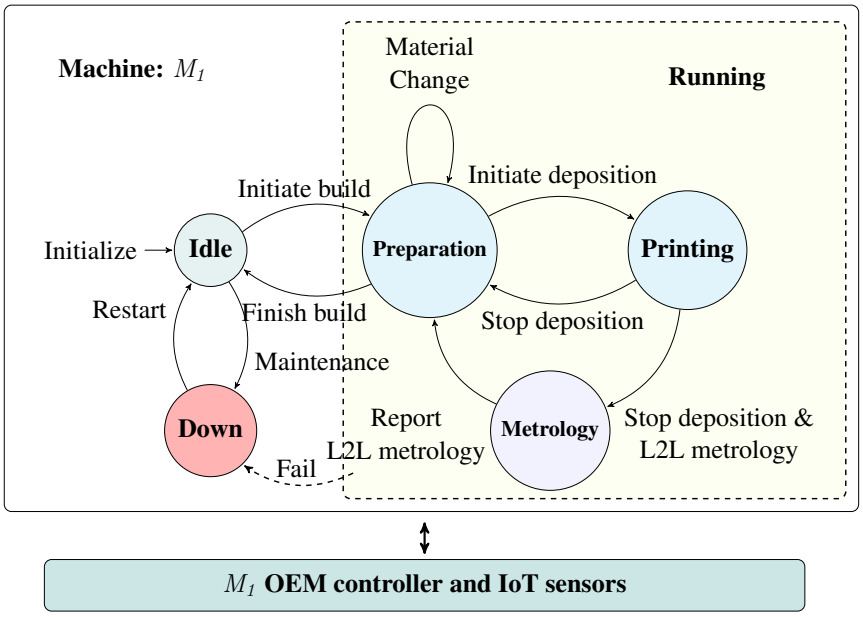


Figure 5.2: Model for the functional states of additive manufacturing and the AM machine with the OEM controller and IoT sensors.

veloped and deployed through the API. The role of the anomaly detection applications is to inform the decision maker about the anomalies in the system. Analyzing the historical data of the machines, R2R anomalies and trends in the system can be identified. The anomaly detection application utilizes the machine data protocols and the digital twin simulations to perform residual based analysis.

Through the API, the central controller can connect to cloud applications. Integration of the services with the cloud applications would enable the integration of the AM-Fleet with other production systems or AM-Fleets. Cloud based integration of the services enables automated AM-Fleets that can update their availability and take production requests through cloud based manufacturing frameworks such as Production as a Service [8, 11, 157].

5.1.3 Discrete Event Models of AM Machines for System-level Monitoring

The AM workflow that begins with the CAD model of a conceptual product is described in [77]. The CAD model is sliced into layers according to user-defined printing parameters and a G-Code instruction file is generated for the AM machine to execute the specified commands to build the part. Discrete event models are commonly used in modeling and supervisory control of systems with deterministic and stochastic events [38]. Here, a functional state model of an AM machine to model the AM process workflow is proposed. Functional states capture the discrete process states of the AM machine for monitoring

purposes. State model is the abstraction of the AM machine (Section 5.1.2.2), used by the central controller to monitor the machines in the AM-Fleet. Using state models, the central controller has a global view of the states of individual machines in the fleet.

Figure 5.2 presents a process-independent functional state model for a single AM machine using a finite state automaton (FSA). The events in the model are triggered by the G-Code instructions passed from the proprietary controller of the machine. The events can be observed by tracking the execution of the individual G-Codes in the software by the proprietary controller. The discrete states of the system are as follows:

Idle: This state captures the pre-build dynamics. The idle state is initiated by the central controller through the DIOI using the CAD and process descriptors.

Preparation: Represents the between-layer actions for the system. After the *initiate deposition* event, this state is characterized by changing material and moving up in Z axis for the new layer.

Printing: Starting with the first layer until the last layer, the dynamics in the $X-Y$ plane and the deposition axis E are modeled in this state. In-situ VM and in-layer outputs are represented in this state.

Metrology: Represents the layer-to-layer true metrology measurements. This state is especially important for *anomaly detection* and *quality control*. Post-build measurement protocols are also represented in this state. As an example, at the finish of the process, the machine would execute the sequence of events $L2Lmetrology, Report, FinishBuild$ to reach the *idle* state and wait for the next job.

Down: This state describes the downtime actions of the printer. The transition might be due to the planned maintenance events or anomalies that occurred in the system.

The FSA can be represented with the tuple $M = (Q, \Sigma, \gamma, q_0)$, where Q is the set of states, Σ are the events, γ is the transition function and the q_0 is the initial state. The tuple for the AM machine M_1 is defined as:

$$
\begin{aligned}
Q &= \{\text{Idle}, \text{Preparation}, \text{Printing}, \text{Metrology}, \text{Down}, \text{Running}\} \\
\Sigma &= \{\text{Initialize}, \text{Initiate/Finish build}, \text{Initiate/Stop deposition}, \\
&\qquad \text{Stop deposition\&L2L metrology}, \\
&\qquad \text{Report L2L metrology}, \text{Maintenance}, \text{Restart}, \text{Fail}\} \\
\gamma &: Q \times \Sigma \rightarrow Q \\
q_0 &: \{\text{Idle}\}
\end{aligned}
$$

The proposed FSA model enables supervision of the AM system in the central controller. Monitoring the events of an AM machine can reveal any unexpected events that occurred during the printing process. The in-situ, layer-to-layer (L2L) metrology, and post-process data are extracted from the machine using the IoT Sensors (Fig. 5.1). The FSA model of each machine is monitored by the central controller through the DIOI. This abstraction enables the central controller to quickly assess the state of the AM-Fleet. In the FSA model, it is assumed that all the events and the states are observable using IoT sensors. In practice, some of the events may become unobservable due to the hardware and software structure of the specific AM machine.

The FSA model is utilized within the digital twins of individual machines in the AM Fleet to provide a global view of the availability and status for the fleet. In Appendix C, an extension of this FSA model is utilized to implement performance monitoring and anomaly detection digital twins. Here, the purpose of the FSA model is to provide information about machine availability to the decision maker and the central controller.

5.1.4 Conceptual Case Study: Anomaly Detection in an AM-Fleet

Figure 5.3 illustrates how the system level anomaly detection application leverages the historical measurement data to perform predictive anomaly detection for the AM-Fleet.

1&2: As the in-situ, L2L metrology, and post-process measurements from each machine are collected through DIOI, they are stored in the machine profiles in the database of the knowledge base.

3&4: The anomaly detection (AD) application runs all the time and uses the data processing service to access current and past machine data in the knowledge base.

5: The AD application runs anomaly detection algorithms on the post-process dimensional measurements and detects that the measurements from the machine M_1 are trending outside the admissible quality limits.

Detection of the anomalous trend is an example of how the R2R measurements of the machines can be used to predict and detect anomalies. The anomaly detection application can predict the anomalous processes before they are observed in the machine. The prediction enables the predictive maintenance scheduling for the machine, and re-scheduling of affected parts to other machines. The decision maker can use the digital twins to decide which machines to use for the re-scheduled jobs based on the machine health, throughput, and expected quality.

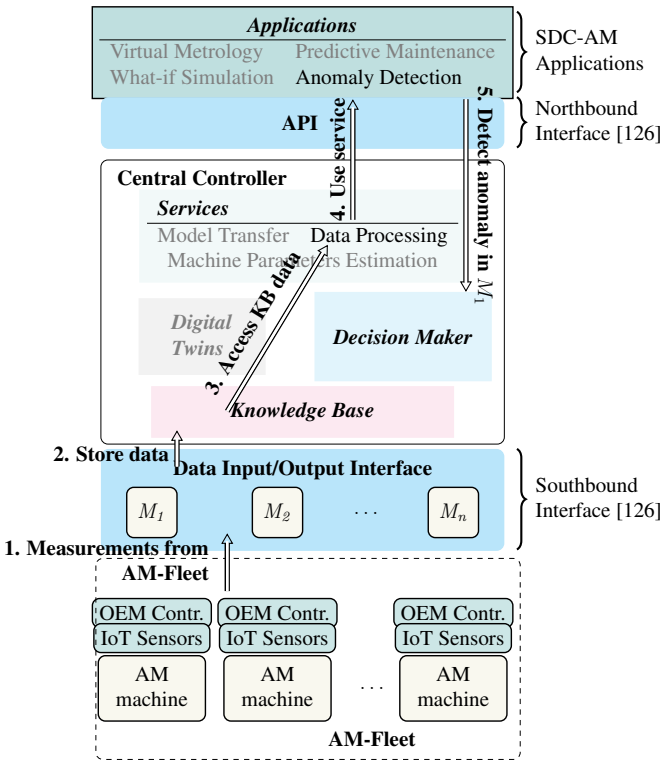


Figure 5.3: An example anomaly detection loop for SDC-AM. Inactive parts in the figure are grayed out.

As an additional example, a VM application for SDC-AM can estimate the dimensional accuracy of the build based on the in-situ measurements. Based on the advice from the VM application, the central controller may request additional metrology events from the AM machine through the DIOI. As additional metrology is requested by the decision maker, the request is sent to the proprietary controller of the machine, through the DIOI and a $Stop\ deposition\ \&\ L2Lmetrology$ event in Fig. 5.2 is triggered at the end of the layer in progress. The metrology event is performed using the IoT sensors and the data is pushed to the central controller through the DIOI. Implementing these examples using the SDC framework will enable a systematic decision making and anomaly detection system for AM-Fleets in industry.

5.2 Run-Time Scheduling for Spatially Distributed Systems

Section 5.1 presents a scalable centralized control framework approach for AM Fleets as an example of spatially distributed systems. Although the architecture and the components of the presented system-level control framework are specialized for AM Fleets, ex-

tensions to other application domains with spatially distributed systems are possible. The conceptual case study in Section 5.1.4 presents the high-level ideas for anomaly detection and rescheduling in an AM Fleet. This section focuses on closed-loop control for run-time scheduling in AM Fleets. Run-time scheduling is an essential challenge for AM Fleets as many production orders may be customized, requiring varying levels of cost, quality, and time. Therefore, a static schedule for an AM Fleet may not be optimal or feasible since orders are customized and may be difficult to forecast due to the customization. Additional events such as anomalies, unexpected downtimes, and maintenance events require rescheduling in run-time. Scheduling is often studied on an abstracted model of the underlying system representing the resources, connectivity, and cost metrics. This section presents a general discrete event model class for scheduling applications. Furthermore, we present how the digital twins of the AM Fleet may be utilized to provide a run-time model of the AM Fleet for closed-loop scheduling control. The presented approach illustrates a case where production constraints and efficiency of the AM Fleet are ensured by the proposed controller considering the run-time information and models provided by the digital twins. The *main contribution of this chapter* is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets (*C3*). The specific contributions of this section in the context of the main contribution *C3* are provided at the end of the next subsection, following an introductory discussion about the research question. The outline of the section is as follows. Subsection 5.2.1 introduces the research question and provides the specific contributions. Subsection 5.2.2 introduces a timed-automata modeling formalism. Subsection 5.2.3 formulates the control problem, which is then implemented utilizing the format in Subsection 5.2.4. Subsection 5.2.5 illustrates an AM Fleet scheduling case study to demonstrate the use of the proposed controller within the SDC-AM framework. The contents of the rest of this section are from the co-authored work published in [10].

5.2.1 Model Predictive Control of Priced Timed Automata Encoded with First-Order Logic

Discrete event system modeling is often used to create abstract models and behavior specifications for complex real-world systems, enabling rigorous verification of specification satisfaction [132, 196]. Timed automata are a common example of such a framework in which the system states include not only a discrete location, but also the continuous values of a set of clocks [5]. Linear priced timed automata (PTA) are an extension of timed automata, featuring a cost function wherein each discrete location is associated with a linear cost rate, and each edge is associated with a cost increment [115]. The cost function

enables cost-optimal reachability analysis (CORA), i.e. the optimization of path planning with respect to criteria other than the quantity of discrete transitions or elapsed time. These automata have become widely used because they are intuitive to design and interpret [5], highly descriptive and flexible [161], and are supported by particularly mature and well-maintained formal verification tools [24, 161].

Most implementations adopt an open-loop approach to controlling systems modeled with PTA, meaning CORA is performed offline and the resultant sequence of planned actions is executed over time. While this approach has proven effective in many situations, it is limited in that it is not robust to disturbances, i.e. uncontrolled changes in the system that occur during execution of the planned path. Disturbances are an important practical concern in scheduling problems, and failure to address them may increase the cost of a plan or render it infeasible. For example, in a manufacturing system, damage to a tool may increase its energy consumption or make it unavailable at the time of planned use. Similarly a problem in an AM machine may require the product to be rescheduled to other machines in the fleet, or a problem on a downstream machine may render the previously calculated schedules infeasible. To address this gap, this work introduces the concept of model predictive control (MPC) using PTA.

MPC is a closed-loop control strategy that performs repeated optimal control over time, enabling it to account for disturbances. Because MPC intrinsically makes use of optimization-based planning, it is a natural closed-loop extension of open-loop optimization-based PTA control. Most of the existing works on MPC for discrete event systems focus on systems represented using max-plus algebra [56, 57]. These systems handle the synchronization of predefined events, and are incapable of modeling the ability to choose between multiple events that accomplish the same goal. The controllers operate over multiple iterations of a repetitive process, seeking to improve the performance of subsequent trials rather than mitigating the effect of unforeseen time-domain disturbances within the current trial. There has also been work on time-domain MPC for deterministic hybrid systems [28, 187], which have more extensive continuous dynamics than PTA. While the theory of MPC for hybrid systems is mature, the structures of the hybrid automata used for MPC are significantly different from PTA, making direct application of hybrid system MPC tools infeasible. In short, the current literature provides no application of MPC to discrete event systems that (1) can be used with PTA to improve the robustness of CORA-based solutions to unforeseen time-domain disturbances, and (2) is able to compute optimal control actions based on constraint feasibility, without the need to iterate on repetitive tasks.

The solvers used in the above prior works share one important capability that is absent from contemporary CORA, namely, constraint softening. A soft constraint is one that

may be violated in exchange for an additive penalty in the cost function of an optimization problem. This is advantageous for planning and scheduling because the constraints in these problems often represent objectives or nominal conditions rather than inviolable physical laws. Violations of these constraints can thus represent compromise or emergency solutions. By representing violations as cost penalties, constraint softening enables the consideration of contingency solutions without complicating the PTA's structure.

To allow the softening of constraints in PTA MPC, this section utilizes a method modified from the existing literature to translate the automaton structure and CORA problem to a set of first-order logic (FOL) constraints, enabling the adoption of an optimization modulo theories (OMT) solver that easily admits constraint softening. While this is not the first translation of PTA/CORA into FOL [7, 30], to the best of our knowledge it is the first work to harness the FOL framework for constraint softening. Additionally, the FOL translation scheme provided in this section is efficient in the sense that it enables the proposed MPC controller to optimize over only the discrete transitions of a PTA rather than encoding all possible delay and discrete transitions as proposed in [7, 30].

The *main contribution of this chapter* is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets (*C3*). Within this main contribution, this section focuses on the development of novel closed-loop controllers for run-time scheduling control. The specific contributions of this section are [10]

(*C3*-4) A model predictive control framework for PTA to increase robustness to disturbances.

(*C3*-5) The application of constraint softening to CORA to increase the feasible solution set to path planning problems without increasing model complexity.

(*C3*-6) A modified FOL representation of PTA based on previous work in [7, 30] for facilitating the execution of MPC and constraint softening via an OMT solver.

The solver is integrated into a simulation implementation of MPC on a PTA model of an AM Fleet scheduling scenario, where we simulate a case with the decision-maker of SDC-AM utilizing the proposed controller for closed-loop production scheduling in the fleet.

5.2.2 Priced Timed Automata

5.2.2.1 Definition of Priced Timed Automata

This section gives both an intuitive explanation and formal definition of PTA, based on [26]. A PTA is composed of a set of locations, Q, connected by a set of edges, E. Each

edge is labeled with an event, $\sigma \in \Sigma$. A PTA also has clocks represented by the elements of $\mathbf{c} \in C = \mathbb{R}^{n_c}_{\geq 0}$, where $\mathbb{R}_{\geq 0}$ is the set of real numbers greater than or equal to zero and n_c is the number of clocks in the system. All clocks have the same constant, positive growth rate and an initial value of zero.

A finite set of Boolean indicator functions of clock values, $\mathcal{B}(C)$, represents the constraints. For example, the Boolean indicator function $\mathcal{I}_{\mathbf{c}^i \leq a}(\mathbf{c}) \in \mathcal{B}(C)$ evaluates to *true* if and only if the value of the i^{th} element of $\mathbf{c}$ is less than or equal to a. Each element of $\mathcal{B}(C)$ is either an invariant for a location, which must evaluate to true for the system to occupy the location, or a guard on an edge, which must evaluate to true to traverse an edge. Reset maps on edges set clocks to predetermined values when the edge is traversed.

Additionally, a PTA has costs on locations and edges. Costs on edges are discrete increments added to the total running cost when the edge is traversed, while the locations have cost rates, and steadily accumulate cost over time.

Definition 5.1 (Priced Timed Automata)**.** A PTA $\mathcal{A}$ is defined as an 8-tuple $\mathcal{A} = (Q, C, \Sigma, E, I, R, P, q_0)$, where

- $Q = \{q^0, q^1, ..., q^{n_q}\}$ is a finite set of locations
- $C = c^0 \times c^1 \times ... \times c^{n_c} = \mathbb{R}^{n_c}_{\geq 0}$ is the clock state space
- $\Sigma = \{\sigma^0, \sigma^1, ..., \sigma^{n_\sigma}\}$ is a finite set of events
- $E \subseteq Q \times \mathcal{B}(C) \times \Sigma \times Q$ is a finite set of edges
- $I : Q \to \mathcal{B}(C)$ is the invariant operator
- $R : E \times C \to C$ is the reset operator
- $P : Q \cup E \to [0, \infty)$ maps locations and edges to costs
- $q_0 \in Q$ is the initial location

Note that guards are embedded in the edge definition, with the second element of the edge 4-tuple being the guard.

5.2.2.2 Transitions and Paths on PTA

A transition is a formal description of a change in the system state, the transition type, and the price of the change. Transitions are notated as

$$(q_i, \mathbf{c}_i) \xrightarrow{v}_p (q_{i+1}, \mathbf{c}_{i+1}) \tag{5.1}$$

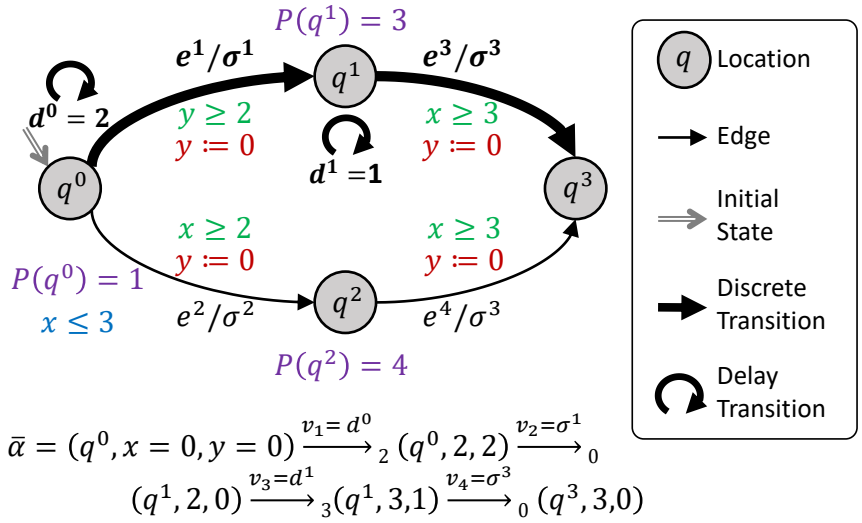


Figure 5.4: A PTA with a path of total cost 5, $\bar{\alpha}$, indicated in bold. x is the global clock and y is the local clock. Example costs (e.g. $P(q^0) = 1$), invariants (e.g. $x \leq 3$), guards (e.g. $y \geq 2$, $x \geq 2$), and resets (e.g. $y := 0$) are labeled.

where i and $i + 1$ indicate the pre- and post-transition system state, p is the price of the transition, and v is the method of the transition, which depends on the transition type. There are two types of transitions, discrete transitions (changes in location) and delay transitions (changes in clock values).

Definition 5.2 (Discrete Transitions)**.** A transition (5.1) is a discrete transition if: the PTA contains $e = (q_i, g, \sigma, q_{i+1}) \in E$, the guard is satisfied $\mathbf{c}_i \models g$, $v = \sigma$, $\mathbf{c}_{i+1} = R(e, \mathbf{c}_i)$, and $p = P(e)$. We say a discrete transition is *taken* when an edge in the PTA is traversed.

Note that $\models$ is used to indicate that the condition on the left side satisfies the condition on the right side.

Definition 5.3 (Delay Transitions)**.** A transition (5.1) is a delay transition if: $q_i = q_{i+1}$, v is a duration of time d, $\mathbf{c}_{i+1} = \mathbf{c}_i + d$, both $\mathbf{c}_i$ and $\mathbf{c}_{i+1}$ satisfy the invariant $I(q_i)$, and $p = P(q_i)d$.

Definition 5.4 (Path)**.** A path α on a PTA is a sequence of discrete and delay transitions in which the post-transition state of each transition is equal to the pre-transition state of the subsequent transition. Thus a path can be notated as

$$\alpha = \left\langle (q_0, \mathbf{c}_0) \xrightarrow{v_1}_{p_1} (q_1, \mathbf{c}_1),\ (q_1, \mathbf{c}_1) \xrightarrow{v_2}_{p_2} (q_2, \mathbf{c}_2), \cdots \right\rangle \tag{5.2}$$

and the path's total cost is given by $\text{cost}(\alpha) = \sum_{i=1} p_i$.

Remark 5.5. *The work in [25] has proven that determining a path of cost infinitesimally close (quantized by computer hardware in practice) to the infimum cost of reaching a particular location on a PTA is decidable.*

Fig. 5.4 shows an example of a path. In this figure, the global clock, x, keeps track of the time from the initial transition. The local clock, y, keeps track of the time spent at each location by resetting to 0 upon each discrete transition. The path illustrated on the figure, $\bar{\alpha}$, consists of four transitions: a delay, a discrete transition, a second delay, and a second discrete transition.

5.2.2.3 Priced Timed Automata Assumptions

The following assumptions on PTA models are used for developing the controller proposed In this chapter:

Assumption 5.6. *The PTA is deterministic, meaning*

1. *all transitions are controllable, i.e. the controller decides when to take a discrete or delay transition, and*

2. *for each $q \in Q$, there can be at most one edge leaving q that has a particular event σ*

Assumption 5.7. *The graph of the PTA is simple. A simple graph does not have any self-edges or multi-edges. A self-edge connects a location to itself. Multi-edges are edges that have the same starting and ending locations.*

Assumption 5.8. *The PTA has at least two clocks: a local and a global clock. The local clock represents the time spent at the current location, i.e. every edge in the PTA has a reset map that sets the local clock to 0. The global clock represents the time elapsed since the system's initialization, i.e. no edges reset the global clock.*

A similar clock structure is found in [7] and provides a formal structure for the control problem in later sections. Furthermore, although additional clocks can be defined, the two clock structure will be analyzed In this chapter for simplicity.

Assumption 5.9. *The graph of the PTA is acyclic. There is no possible way to start from one location and, by following the edges in the graph, end at the same location.*

This assumption is adopted to ensure unique clock valuations. The loop back index is proposed in [7] for cyclic graphs. We employ acyclic graphs for simplifying our MPC formulation, noting that cyclic graphs can be efficiently "flattened" for a given horizon length [140]. As a result of Assumption 5.9 and Assumption 5.7, underlying graph structures of the PTA In this chapter are directed acyclic graphs (DAG).

Assumption 5.10. *A disturbance in the system is defined as a change to a clock constraint (i.e. altering, removing, or adding a Boolean indicator function in $\mathcal{B}(C)$) or a change to the cost function P of the PTA.*

PTA models with these assumptions can be used for control of a number of real-world systems, such as the manufacturing system example described in [111].

5.2.3 Model Predictive Control Using PTA

5.2.3.1 Optimal Control Problem for PTA

The control objective for a system modeled by a PTA may be determined in part by a string (i.e., a sequence of events) called a spec. Let the spec be represented as $\Sigma^d = \langle \sigma_1^d,\, \sigma_2^d,\, \cdots,\, \sigma_{N_d}^d \rangle$. In this case, the optimal control problem is to find a sequence of transitions $V = \langle v_1,\, v_2,\, \cdots, v_{i_N} \rangle$, such that the resultant path has minimum cost and contains (but is not limited to) the events in Σ^d in order. Hence, the optimal control problem on a PTA, $\mathcal{A}$, can be posed as

$$V^* = \underset{V}{\operatorname{argmin}}\ \operatorname{cost}(\alpha) \tag{5.3a}$$

$$\text{subject to:} \quad \alpha\ \in \mathcal{L}(\mathcal{A}) \tag{5.3b}$$

$$v_{i_1} = \sigma_1^d, ..., v_{i_N} = \sigma_{N_d}^d \tag{5.3c}$$

$$i_1 < i_2, ..., i_{N-1} < i_N \tag{5.3d}$$

where $\mathcal{L}(\mathcal{A})$ is the set of feasible paths on $\mathcal{A}$. Constraint (5.3b) is equivalent to enforcing all of the constraints dictated by definitions 5.2-5.4. Constraint (5.3c) ensures that V^* contains all of the events in Σ^d and constraint (5.3d) ensures that these events occur in the order specified by Σ^d.

Note the path in Fig. 5.4, $\bar{\alpha}$, results from the solution to the optimal control problem (5.3) with the desired event $\Sigma^d = \{\sigma^3\}$.

5.2.3.2 Model Predictive Control with PTA

The central principle of MPC is the repeated solution of an optimal control problem over a receding horizon. Because MPC performs optimizations repeatedly, requiring every optimization to find a path satisfying all future events may not be efficient. The PTA-MPC strategy finds a path of minimum cost on a PTA such that only a sub-spec of Σ^d is executed. The sub-spec of desired events is defined as $D = \langle \sigma_j^d,\, \sigma_{j+1}^d,\, \cdots,\, \sigma_{j+N_{spec}-1}^d \rangle$, where j is the index of the first remaining desired event yet to be executed and N_{spec} is the number

of events in D. Hence, the PTA-MPC strategy is to solve (5.3), replacing constraint (5.3c) with:

$$v_{i_1} = \sigma^d_j, ..., v_{i_N} = \sigma^d_{j+N_{spec}-1} \tag{5.4}$$

The value of N_{spec} is defined as the spec horizon, which is a tuning parameter that affects feasibility of the solution of the optimal control problem for PTA-MPC. A discussion on computing lower-bounds for N_{spec} is provided in Section 5.2.4.3.

When a sequence of transitions is obtained from the solution to the PTA-MPC optimal control problem, only the first transition is executed. The PTA is then updated based on measured information. Specifically, $\mathcal{A}$ is modified to reflect any disturbances (see Assumption 5.10) that may have occurred since the preceding optimization. Additionally, if a desired event was executed, it is removed from D and, if Σ^d has more than $j + N_{spec} - 1$ events, the $j + N_{spec}$ event is appended to D. With the updated model and sub-spec, (5.3) is solved again and another transition is executed. Thus, the total path ultimately executed by the controller accounts for disturbances.

5.2.3.3 Soft Constraints for PTA

During system modeling, the inclusion of soft constraints allows for penalized constraint violations. For a PTA $\mathcal{A}$, relaxing guards and invariants by constraint softening allows for more flexibility in the timing occurrence of events. Note that this work only addresses the softening of clock constraints.

This set of softened clock constraints is a subset of the Boolean clock constraints, $S \subseteq \mathcal{B}(C)$. To soften the guard and invariant constraints in the MPC-PTA problem, (5.3) must be modified. Specifically, (5.3b) is adapted by adding slack variables to constraints in S. A softened constraint is formulated as $\mathcal{I}_{\mathbf{c}^s \leq a+\gamma_s}(\mathbf{c})$, where $\mathbf{c}^s$ is the clock value associated with the soft constraint and γ_s is the corresponding slack variable. (5.3a) is modified to include penalties on slack variables as:

$$V^* = \underset{V, \gamma_s \in S}{\operatorname{argmin}} \ \operatorname{cost}(\alpha) + \sum_{s=1}^{|S|} f_p(\mathbf{\Gamma}_s, \gamma_s), \tag{5.5}$$

where f_p is a penalty function, $\mathbf{\Gamma}_s$ is a constant that represents the cost penalty on the s^{th} slack variable, and $|S|$ is the number of soft constraints. By solving this new optimal control problem, a control action might be obtained even if there are no paths in $\mathcal{L}(\mathcal{A})$ that satisfy (5.3c) and (5.3d).

5.2.3.4 Solving the MPC Problem for PTA

The optimization problem (5.3) can be solved via CORA [25], which traditionally uses a branch-and-bound algorithm detailed in [26] and implemented in the UPPAAL CORA software [23]. However, there has been no development of tools for closed-loop extensions of CORA needed to implement the proposed MPC framework. Additionally, existing CORA algorithms do not admit constraint violations. To solve the MPC problem with soft constraints, we leverage an optimization modulo theories (OMT) solver. OMT is a branch of satisfiability modulo theories (SMT) with cost functions that readily admit constraint softening [31]. However, OMT operates exclusively over first-order logic (FOL) constraints.

A number of works have encoded dynamical systems with propositional logic and have used model checking software to obtain feasible and optimal solutions to their problems [7, 28, 30, 89, 92, 165]. Several have developed frameworks that use SMT solvers to solve an optimization problem over a PTA [7, 30]. To solve the MPC problem posed in (5.3), we leverage these works to reformulate the optimal control problem in FOL and analyze the recursive feasibility of the MPC problem.

5.2.4 First-order Logic Problem Representation

This section expresses the optimal control problem in FOL and analyzes the recursive feasibility of the MPC problem.

5.2.4.1 Representation of the PTA in First-Order Logic

To encode the underlying timed-automata of the PTA, we employ an encoding similar to the one in [7, 30], with several changes. We employ an event-driven encoding scheme that evaluates the clock valuations $val(\mathbf{c})$ only when a discrete transition is taken. The clock valuations are expressed as $val(\mathbf{c}) = [c_l(\mathbf{q}^0), c_g(\mathbf{q}^0), \ldots, c_l(\mathbf{q}^{n_q}), c_g(\mathbf{q}^{n_q})]^T$, where $c_l(\mathbf{q}^j)$ represents the local clock valuation and $c_g(\mathbf{q}^j)$ represents the global clock valuation at the location $\mathbf{q}^j$. The local clock is reset after each discrete transition (similar to the "offset" variable in [7]) and the global clock is never reset to denote the time since the beginning of a path (similar to the "absolute time" variable in [7]). Therefore, we can pose a consistency constraint on the valuations of the global clock at a location, $\mathbf{q}^i$, such that $(c_g(\mathbf{q}^i) = c_l(\mathbf{q}^i) + c_g(\mathbf{q}^m)) \Leftrightarrow e^j$, where e^j is the edge from $\mathbf{q}^m$ to $\mathbf{q}^i$. Further information on evaluating all such constraints for a PTA can be found in [9].

For notational convenience, we denote the input transition matrix $B_{in} \triangleq \max(A, 0)$, where $A \in \{-1, 0, 1\}^{n_q \times n_e}$ is the incidence matrix of $\mathcal{A}$, $\max(\cdot, 0)$ is a function com-

puted element-wise for each element of the first argument, $\triangleq$ denotes a definition for the left-hand side, and $n_e = |E|$ where $|\cdot|$ is cardinality. Similarly, we define $\tilde{A} \triangleq \max(-A^T B_{in}, 0)$. Additionally, we compactly notate a "multiple exclusive or" Boolean constraint as $\texttt{mXor}(\mathbf{a}, \mathbf{b}) \triangleq \mathbf{a}^T\mathbf{b} == 1$, where $\mathbf{a}$ and $\mathbf{b}$ are binary column vectors of equal length. Finally, a first-order logic MPC horizon, N_{mpc}, is defined as the length of the horizon for the MPC formulation. The decision variables of the first-order logic MPC are the discrete transitions of a path on a PTA. Thus, the sequence of discrete transitions $U = \langle u_1, \ldots, u_{N_{mpc}} \rangle$ denotes the decision variables for the first-order logic MPC problem where $u_i \in \mathbb{B}^{n_e}$.

5.2.4.2 Optimal Control Problem in First-Order Logic

The PTA-MPC problem, represented by (5.3) with constraint (5.4) in place of (5.3c), can be formulated in first-order logic utilizing the first-order constraints developed in the previous section. The optimal solution should satisfy all of the desired events in the subspec $\sigma^d \in D$, in sequential order. A sequential satisfaction constraint is given by constraints (5.3d) and (5.4) for the optimal control problem. To translate constraints (5.3d) and (5.4), the following predicate proposition on U is introduced:

$$\varphi(D, n) = \forall j < |D| \text{ and } \forall i \text{ s.t. } u_i \models \sigma_j^d, \exists \ell \text{ s.t. } i < \ell \leq n \text{ and } u_\ell \models \sigma_{j+1}^d \quad (5.6)$$

$\varphi(D, n)$ denotes that for the next n discrete transitions in which $u_i \in U$ satisfies one of the desired events $\sigma_j^d \in D$, there exists a future discrete transition u_ℓ that satisfies the next desired event $\sigma_{j+1}^d \in D$, provided that σ_j^d is not the last desired property in D. Let $\mathbf{p} \in \mathbb{R}^{2n_q}$ be a vector of costs for the clock valuations. $\mathbf{r} \in \mathbb{R}^{2n_q}$ is defined as a vector with the clock valuations and their associated costs; formally as $\mathbf{r} = \mathbf{p} \circ val(\mathbf{c})$, where $\circ$ is the Hadamard product (entry-wise product). Then, the first-order logic PTA-MPC problem is formulated

as:

$$\underset{u_k \in U, val(\mathbf{c})}{\operatorname{argmin}} \quad P(\mathbf{q}_0)c_l(\mathbf{q}_0) + \mathbf{\Gamma}^T\boldsymbol{\gamma} + \sum_{k=1}^{N_{mpc}} \mathbf{r}^T B_{in} u_k \tag{5.7a}$$

$$\text{subject to:} \quad \mathbf{q}_k \leftarrow B_{in}u_k, \qquad k = 1, \ldots, N_{mpc} \tag{5.7b}$$

$$u_k \models \texttt{mXor}(\omega_k, u_k), \quad k = 1, \ldots, N_{mpc} \tag{5.7c}$$

$$\omega_{k+1} = \tilde{A}u_k, \qquad k = 1, \ldots, N_{mpc} \tag{5.7d}$$

$$q_0 = \bar{q}_0, \quad \omega_0 = \bar{\omega}_0 \tag{5.7e}$$

$$U \models \varphi(D, N_{mpc}) \tag{5.7f}$$

$$\mathbf{\Psi}\, val(\mathbf{c}) \leq \mathbf{t} + \boldsymbol{\gamma}, \tag{5.7g}$$

$$val(\mathbf{c}) \models \mathcal{G} \wedge \mathcal{C}, \quad val(\mathbf{c}) \geq \mathbf{0} \tag{5.7h}$$

where, $\bar{q}_0$ denotes the initial state (location), $\Psi \in \mathbb{R}^{\eta \times 2n_q}$, $\mathbf{t} \in \mathbb{R}^{\eta}$ with η is the number of invariant constraints, $\bar{\omega}_0$ denotes a vector of the initial available transitions, $\boldsymbol{\gamma} \in \mathbb{R}^{\eta}_{\geq 0}$ is a vector of slack variables, and $\mathbf{\Gamma} \in \mathbb{R}^{\eta}_{\geq 0}$ is a vector with constraint violation penalties. Constraints (5.7b)-(5.7e) encode the graph structure of the PTA and constraints (5.7f)-(5.7h) encode path-related constraints. Additionally, $\mathcal{G}$ and $\mathcal{C}$ denote the guard conditions and clock (local and global) constraints, respectively. Further details on the computation of $\mathcal{G}$ and $\mathcal{C}$ can be found in Appendix B and [9].

The MPC problem in (5.7) outlines the first-order logic problem to be solved to find the optimal control sequence $U^* = \langle u_1^*, \ldots, u_{N_{mpc}}^* \rangle$ and $val(\mathbf{c})^*$.

The slack variables and constraint violation penalty in (5.7) allow the softening of constraints. Let $S \subseteq \mathcal{B}(C)$ denote a set of constraints to be softened. Then, $\boldsymbol{\gamma}$ is formed as a vector with individual slack variables corresponding to the constraints in S and zeros elsewhere. Additionally, $\mathbf{\Gamma}$ is formed as a vector with a positive integer cost associated with each slack variable in $\boldsymbol{\gamma}$. Note that by softening some of the constraints in $\mathcal{B}(C)$, it may be necessary to add new constraints to limit the behavior of soft constraints. An in depth analysis of such treatments for efficient relaxation schemes is left for future work.

5.2.4.3 Recursive feasibility

To ensure recursive feasibility of the MPC problem in (5.7), lower-bounds on the spec horizon, N_{spec}, and MPC horizon, N_{mpc}, must be established. The spec horizon is the cardinality of $D \subseteq \Sigma^d$ in (5.6) and indicates the number of sequential desired events to be satisfied by the controller in (5.7). If the spec horizon N_{spec} is too small, the MPC optimal path may lead to locations where the satisfaction of the entire set of desired specs, Σ^d, is

not possible. Therefore the choice of N_{spec} must ensure recursive feasibility of the MPC given in (5.7).

Definition 5.11 (Recursively Feasible Spec Horizon)**.** Consider a PTA with the assumptions given in Section 5.2.2.3 and the spec $\Sigma^d = \langle \sigma_1^d, \ldots, \sigma_{N_d}^d \rangle$. A spec horizon N_{spec} is said to be recursively feasible if for all sub-specs $D \subseteq \Sigma^d$ with $D = \langle \sigma_j^d, \ldots, \sigma_{j+N_{spec}-1}^d \rangle$:

1. There exists a control input $U = \langle u_0, \ldots, u_{n_0} \rangle$ such that $U \models \varphi(D, n_0)$, where $\varphi(\cdot, \cdot)$ is given by (5.6) and

2. As $\sigma_j^d \in D$ is satisfied, there exists a new control input $U' = \langle u'_0, \ldots, u'_{n'_0} \rangle$ such that $U' \models \varphi'(D', n'_0)$, where φ' is evaluated for all subsequent updated $D' \leftarrow \langle \sigma_{j+1}^d, \ldots, \sigma_{j+N_{spec}}^d \rangle$ until all events in Σ^d are satisfied.

The lower bound on the recursively feasible spec horizon, denoted by $\hat{N}_{spec}$, is defined as the least number of events that should be considered in D, such that feasibility of the first-order logic (FOL) PTA-MPC problem in (5.7) implies feasibility of satisfying all the events in Σ^d. From this definition, it is clear that $\hat{N}_{spec} \leq |\Sigma^d|$.

Lemma 5.12. *If a spec horizon h is chosen as $\hat{N}_{spec} \leq h \leq |\Sigma^d|$, then the FOL PTA-MPC problem given in (5.7) with sufficiently large N_{mpc} is guaranteed to satisfy all events in Σ^d.*

Proof. Suppose the statement is false. Then, as some desired event $\sigma_i^d \in D$ is satisfied (by the execution of the corresponding discrete transition of the PTA), there exists a desired event $\sigma_j^d \in \Sigma^d, j > i$, such that there does not exist a control input U that could satisfy the sub-spec $D' \ni \sigma_j^d$. In other words, there exists a desired event σ_j^d that cannot be reached from the current location and clock valuations of the PTA. This results in a contradiction with the definition of $\hat{N}_{spec}$. Therefore the statement must be true. □

Remark 5.13. *Since $\hat{N}_{spec} \leq |\Sigma^d|$ holds trivially, the spec horizon N_{spec} can always be chosen as $|\Sigma^d|$. In the cases where $\hat{N}_{spec} < |\Sigma^d|$, smaller spec horizons may be chosen to reduce the computational cost of the FOL PTA-MPC problem in (5.7).*

Similarly, a lower bound on N_{mpc} to ensure recursive feasibility of the FOL PTA-MPC problem in (5.7) is defined as the smallest MPC horizon length for a given spec horizon length N_{spec} such that the constraints Eqs. (5.7b)-(5.7h) can be satisfied. Note that the lower-bound on N_{spec} and N_{mpc} may be different since multiple discrete transitions may need to take place to satisfy a single event $\sigma_i^d \in D$.

Algorithm 2 summarizes the hierarchical MPC structure proposed In this chapter to ensure recursive feasibility along the spec and the MPC horizons. First, a satisfiability solver is used to check if the original set of constraints with all of the events in Σ^d are

Algorithm 2 Proposed MPC structure

1: *Given*: $\mathcal{A} = (Q, C, \Sigma, E, I, R, P, q_0)$ and Σ^d
2: *Initialize*: $N_{spec} \leftarrow |\Sigma^d|$, $D \leftarrow \Sigma^d$, $N'_{mpc} \leftarrow |E|$, $q_k \leftarrow q_0$
3: (i) Check SAT for Eqs. (5.7b-5.7h)
4: **if** (i) is SAT **then**
5: $\hat{N}_{spec} \leftarrow$ lower-bound of N_{spec}, given N'_{mpc}, such that (5.7) is recursively feasible, found by search
6: $N_{spec} \leftarrow \hat{N}_{spec}$, $D \leftarrow \langle \sigma^d_1, \ldots, \sigma^d_{\hat{N}_{spec}} \rangle$
7: $\hat{N}_{mpc} \leftarrow$ lower-bound on N_{mpc}, given $N_{spec} = \hat{N}_{spec}$, such that (5.7) is recursively feasible, found by search
8: *Choose* horizons s.t. $N_{mpc} \geq \hat{N}_{mpc}$ and $N_{spec} \geq \hat{N}_{spec}$
9: $D \leftarrow \langle \sigma^d_1, \ldots, \sigma^d_{N_{spec}} \rangle$
10: **else**
11: **return** UNSAT
12: **end if**
13: **while** not all specs in Σ^d are SAT **do**
14: $(U^*, val(\mathbf{c})^*) \leftarrow$ The solution of (Eq. (5.7))
15: Execute delay transition of duration $c_l(\mathbf{q}_k)^* \in val(\mathbf{c})^*$
16: Execute the discrete transition $u^*_1 \in U^*$
17: $\mathbf{q}_k \leftarrow B_{in} u^*_1$
18: **if** a spec $\sigma^d_j \in D$ is SAT **then**
19: Remove σ^d_j from D and append $\sigma^d_{j+N_{spec}}$ to D
20: **end if**
21: **end while**

satisfiable (SAT). $N_{mpc} = |E|$ is taken as a large enough horizon for the satisfiability problem in step (i). If (i) is unsatisfiable, the algorithm terminates and returns UNSAT. Since $|E|$ may be very large, a smaller feasible horizon length for N_{mpc} may be evaluated for line 5 in Algorithm 2, by using longest path algorithms such as the ones given in [53] (Ch. 4,6) to find the longest path in the underlying DAG starting from the initial location q_0. Therefore $\exists N'_{mpc} \leq |E|$ to satisfy all events in Σ^d, given that (i) is SAT. The next step is finding the lower-bound $\hat{N}_{spec}$ that ensures recursive feasibility with the MPC horizon taken as N'_{mpc}, using a search algorithm, such as linear search or binary search.

Proposition 5.14. *Consider a PTA with the assumptions given in Section 5.2.2.3. If a sequence of discrete transitions $\tilde{U} = \langle \tilde{u}_1, \ldots, \tilde{u}_{n_1} \rangle$ is the minimal sequence of discrete transitions that satisfies all the events in a spec Σ^d, (i.e. $\tilde{U} \models \Sigma^d$), there exists another sequence of discrete transitions $\bar{U} = \langle \bar{u}_1, \ldots, \bar{u}_{n_2} \rangle$ with $n_2 \leq n_1$ that satisfies the sub-spec $\bar{D} \subset \Sigma^d$.*

Proof. The following constructive proof is given for Proposition 5.14. For $n_2 = n_1$, $\bar{U} \models \bar{D}$ follows immediately by choosing $\bar{U} = \tilde{U}$ and noting that $\bar{D} \subset \Sigma^d$. For the situation with

strict inequality $n_2 < n_1$, again let $\bar{D} \subset \Sigma^d$. Then $\exists \sigma_m^d \in \Sigma^d$ such that $\sigma_m^d \notin \bar{D}$. Since $\tilde{U} \models \Sigma^d$, $\exists \tilde{u}_i \in \tilde{U}, \tilde{u}_i \models \sigma_m^d$. Combining the last two conditions and noting that a discrete transition $\tilde{u}_i$ satisfies a unique event, results in $(\tilde{U} \setminus \tilde{u}_i) \models \bar{D}$. Thus, choosing $\bar{U} = \tilde{U} \setminus \tilde{u}_i$, the result $\bar{U} \models \bar{D}$ is obtained with $|\bar{U}| < |\tilde{U}|$, where $|\cdot|$ is the length of a sequence. □

It is straightforward to show that $N'_{mpc} \geq |\tilde{U}|$, where $\tilde{U}$ is the minimal sequence of discrete transitions that give $\tilde{U} \models \Sigma^d$. Thus, by Proposition 5.14 the search for $\hat{N}_{spec}$ given that $N_{mpc} = N'_{mpc}$ is always feasible for all $D \subset \Sigma^d$. Next, a lower-bound of N_{mpc} for the recursive feasibility of (5.7) given that $N_{spec} = \hat{N}_{spec}$ (and $D \leftarrow \langle \sigma_1^d, \ldots, \sigma_{\hat{N}_{spec}}^d \rangle$) is found by utilizing a search algorithm. The spec and MPC horizons are chosen as greater than or equal to the respective lower-bounds $\hat{N}_{spec}$ and $\hat{N}_{mpc}$ and the corresponding D is evaluated. The proposed MPC is run with the choices of N_{spec}, N_{mpc}, and D until all the events in Σ^d are satisfied. Each MPC loop consists of solving the PTA-MPC problem in (5.7). The optimal solution $(U^*, val(\mathbf{c})^*)$ of (5.7) is used for evaluating the optimal delay transition at the current location as $c_l(\mathbf{q}_k)^* \in val(\mathbf{c})^*$ and the optimal discrete transition as $u_1^* \in U^*$. As an event in D is satisfied, the satisfied event is excluded from D and the next event from Σ^d is appended to D.

Theorem 5.15. *The MPC structure in Algorithm 2 for the PTA-MPC problem in (5.7), with no disturbance and N_{spec} and N_{mpc} chosen according to Algorithm 2, will either satisfy all the events in Σ^d in order or return* `UNSAT` *if the PTA-MPC problem is unsatisfiable.*

Proof. Algorithm 2 evaluates the appropriate horizons N_{spec} and N_{mpc} to ensure recursive feasibility on both horizons. Recursive feasibility ensures that the optimization will satisfy the next N_{spec} desired events until all the desired events in Σ^d are satisfied. This is shown by Lemma 5.12. Checking the satisfiability of (5.7) ensures that the desired events within the horizon D are satisfied in order (order is encoded by constraint (5.7f)). Therefore, the controller satisfies all the desired events in the given spec Σ^d. In the case where the PTA-MPC problem is unsatisfiable, Algorithm 2 returns `UNSAT`.

□

Remark 5.16. *The lower-bounds $\hat{N}_{spec}$ and $\hat{N}_{mpc}$ are computed offline for a given PTA in Algorithm 2. The computational cost is related to the complexity of the employed search algorithm. Note that SMT solving is utilized in the search rather than OMT, which has a significantly reduced computational cost since OMT employs solutions to a series of SMT problems to evaluate an optimal solution [31].*

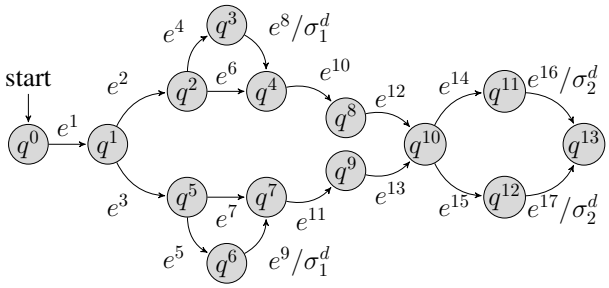
5.2.4.4 First-Order Logic to Original Optimal Control Problem

The solution to (5.7) consists of U^*, the sequence of discrete transitions, and $val(\mathbf{c})^*$, the clock valuations for all of the states. An optimal path, α^*, is evaluated for the PTA using U^*, $val(\mathbf{c})^*$, and the initial state $\bar{q}_0$. The initial delay transition is evaluated by noting that the path starts at time 0 and stays at the initial state for the time in the local clock valuation, $val(c_l(\bar{q}_0))^*$. The following discrete transition is evaluated by obtaining the first transition in U^* and noting that the time of the transition is at $val(c_l(\bar{q}_0))^*$. Subsequent delay and discrete transitions of the path are evaluated until all of the transitions in U^* have been taken.

5.2.5 AM Fleet Example

In this subsection, the proposed MPC controller is demonstrated using a PTA model for customized part scheduling within an AM Fleet. In this setting, we assume that a certain production job needs to be scheduled within the AM Fleet to be printed, post processed, and delivered to an output buffer for delivery. The central controller from Section 5.1 is assumed to be in place as the central controller for the AM Fleet. Then, using the digital twins such as the ones presented in Section 5.1, the central controller efficiently analyzes the availability of resources in the AM Fleet. Note that since many of the machines may be unavailable for the rescheduling taking place, the PTA model created by the SDC-AM central controller will span a subset of the resources in the AM Fleet. We have the proposed MPC controller implemented in the decision maker in the central controller of SDC-AM as a core capability of the decision maker. Due to the availability of the digital twins and run-time data available to the central controller and the decision maker, the PTA model is updated in run-time. The availability of run-time observations through the digital-twins combined with the receding horizon nature of the proposed controller enables robustness to disturbances when compared to static models as we demonstrate in this case study example.

An alternate method in the absence of the SDC-AM and digital twins of the resources in the AM Fleet is to synthesize a representation of the AM-Fleet as a PTA and generate an optimal schedule based on the given model to be executed as a production plan. However, once a path is planned, this method does not have any means to counteract disturbances that may arise during the plan's execution. We demonstrate how this approach may lead to infeasibility and unexpected disruptions in the production and illustrate the utility of the proposed controller in conjunction with the system-level control architecture (SDC-AM) given in Section 5.1.



Clock Valuation Constraints:

$c_l(\mathbf{q^0}) \geq 1$ $c_l(\mathbf{q^1}) \geq 1$ $c_l(\mathbf{q^3}) \geq 8$
$c_l(\mathbf{q^6}) \geq 3$ $c_g(\mathbf{q^8}) \geq 12$ $c_g(\mathbf{q^9}) \geq 24$
$c_l(\mathbf{q^8}) \geq 1$ $c_l(\mathbf{q^9}) \geq 1$ $c_l(\mathbf{q^{10}}) \geq 2$

$$c_l(\mathbf{q^{11}}) = \begin{cases} \geq 1, \textbf{if } c_g(\mathbf{q^{11}}) < 8 \\ \geq 5, \textbf{if } c_g(\mathbf{q^{11}}) \geq 8 \end{cases} \quad c_l(\mathbf{q^{12}}) \geq 3$$

Prices:

$P(q^0) = 1$ $P(q^1) = 1$ $P(q^2) = 1$ $P(q^3) = 3$
$P(q^4) = 1$ $P(q^5) = 1$ $P(q^6) = 3$ $P(q^7) = 1$
$P(q^8) = 1$ $P(q^9) = 1$ $P(q^{10}) = 1$ $P(q^{11}) = 3$
$P(q^{12}) = 2$ $P(q^{13}) = 1$

Figure 5.5: PTA graph used for the case study. Locations are shown with circles (q^i) and discrete transitions are shown with edges (e^i). Details about clock valuation constraints and delay transitions prices are shown below the graph.

The PTA model for the rescheduling problem consisting of two AM machines, a post processing station, a target location ("buffer") for processed products, and autonomous ground vehicles (AGVs) for transporting products around the AM Fleet (e.g., between the machine and the post process stations). The production goal is for a product to be printed, post-processed, and delivered to the target buffer. The system topology is given as a PTA in Fig. 5.5, and location descriptions are given in Table 5.1. The product to be rescheduled enters the starting queue where the decision maker runs the proposed controller to determine the next action for the product. Note that an AM product is initially a design file, which is converted into machine instructions that processed based on product specifications and the specific machine setup, modeled by the design preprocessing location in Table 5.1. The preprocessed design enters the queue of the respective AM machine AM1 or AM2, and waits at the exit buffer after the print. AVG1 takes the printed parts to the post process station, which is then taken by either AGV2 or AGV3 to the target buffer. Guards on local clocks define the duration a process requires, and guards on global clocks represent the time at which a resource becomes available. Costs are used to represent the energy consumption of a manufacturing resource (with arbitrary unit for the conceptual example). The unit for time is considered as simulation ticks so that the resulting clock valuation times may be

Table 5.1: List of the locations in the PTA model given in Fig. 5.5

q^0	Starting queue	q^2, q^5	Queue for AM1, AM2
q^1	Design preprocessing	q^4, q^7	Exit buffer for AM1, AM2
q^3, q^6	AM1 and AM2	q^8, q^9	Transit path 1 and 2 for AGV 1
q^{10}	Post process	q^{11}	In transit (AGV 3)
q^{12}	In transit (AGV 2)	q^{13}	Target buffer

translated into real-life examples by defining the time-length of a single simulation tick. Note that this is not a restricting consideration since the time-length of a simulation tick may be defined arbitrarily small.

This illustrative case study has been modified from the one presented in our work [10], where instead of an AM Fleet, an agent-based manufacturing system control example is presented. Here we modify the physical system that is represented by the model in Fig. 5.5 and situate it in the context of the SDC-AM controller given in Section 5.1. Accordingly, we modify the resources presented in the manufacturing system to represent an AM Fleet, and focus on the availability of the model information to the SDC-AM controller through the digital twins of the resources in the fleet. The simulation results presented in the rest of this section are thus the same as those presented in [10], with the discussions modified to represent the AM Fleet and its resources wherever necessary.

5.2.5.1 Simulation Setup

Three simulation trials are to be performed on the manufacturing system model:

(S1) Open-loop optimal controller proposed in [110]

(S2) Proposed MPC with all hard constraints

(S3) Proposed MPC with allowable constraint violations

All simulations consist of finding solutions to the optimization problem in (5.3) for the PTA in Fig. 5.5. Solutions to (5.7) for (S2) and (S3) are computed with a customized PTA MPC solver. The customized solver uses the Z3 OMT solver [31] implemented in python (with Z3Py library) to compute the optimization in (5.7). All simulations are run on a desktop PC with Intel® Core™ i7-6700 CPU.

The open-loop optimal control problem in (S1) consists of a single computation of the optimization solution based on the system model known at time zero (i.e. not including disturbance information). The MPC simulations (S2) and (S3) are conducted according to Algorithm 2. At each execution, the central controller utilizes the digital twins to update the

Table 5.2: The parameters (horizons, slack variable values, deadline enforcement) and results (statistics for computation time to solve the MPC problem and clock value for accomplished spec) for the three different simulation trials (S1, S2, and S3).

Simulation	Horizon		Slacks at End		Deadline Enforced	Computation Time		Label SAT $c_g(\cdot)$	
#	N_{spec}	N_{mpc}	γ_1	γ_2	$c_g(\mathbf{q}^{13}) \leq 23$	μ_t [ms]	σ_t [ms]	σ_1^d	σ_2^d
(S1)	—	—	—	—	N/A	22.9	2.6	$c_g(\mathbf{q}^4) = 17$	$c_g(\mathbf{q}^{13}) = 26$
(S2)	1	4	—	—	N/A	24.5	8.4	$c_g(\mathbf{q}^7) = 12$	$c_g(\mathbf{q}^{13}) = 31$
	1	6	—	—	N/A	43.4	12.8	$c_g(\mathbf{q}^4) = 17$	$c_g(\mathbf{q}^{13}) = 24$
	2	8	—	—	N/A	51.7	14.3	$c_g(\mathbf{q}^4) = 17$	$c_g(\mathbf{q}^{13}) = 24$
(S3)	1	6	0	1	✓	87.8	16.2	$c_g(\mathbf{q}^4) = 17$	$c_g(\mathbf{q}^{13}) = 23$
	2	8	0	1	✓	112.5	39.6	$c_g(\mathbf{q}^4) = 17$	$c_g(\mathbf{q}^{13}) = 23$

PTA with an up-to-date representation of the system, accounting for disturbances that may have occurred, which in turn enables the controller to react to the changes and disturbances occurring in the AM Fleet as presented below.

All simulations are subject to a disturbance, which is given by the guard in Fig. 5.5, and represents a non-catastrophic failure in AGV3 (q^{11}). This failure allows the AGV to continue, but at a slower travel speed. The desired string of events Σ^d is defined as $\langle \sigma_1^d, \sigma_2^d \rangle$. σ_1^d represents machining completion and is associated with edges e^8 and e^9. σ_2^d represents arrival at the target buffer and is associated with edges e^{16} and e^{17}. Quality checking after the machining process is completed at q^{10}.

A new hard constraint is introduced for the third simulation (S3), $c_g(\mathbf{q}^{13}) \leq 23$. To restore satisfiability, the constraints on AGV3 ($c_l(\mathbf{q}^{11})$) and AGV2 ($c_l(\mathbf{q}^{12})$) are softened with new costs $\gamma_1 = 60$ and $\gamma_2 = 40$. These constraint violation costs are set proportional to nominal state costs to preserve modeling of AGV3 as more costly to operate than AGV2. Additionally, the hard constraints $c_l(\mathbf{q}^{11}) > d(q^{10}, q^{13})/max(spd_{AGV_3})$ and $c_l(\mathbf{q}^{12}) > d(q^{10}, q^{13})/max(spd_{AGV_2})$ are added, where $d(q^{10}, q^{13})$ is the distance between the quality check station and the target buffer and $max(spd_{AGV})$ represents the maximum speed of an AGV. This softening represents a scenario in which the constraints indicate the maximum sustainable AGV speed, and constraint violation represents operating AGVs at a higher speed that may be feasible only for short durations and risks increased battery degradation. Note that the constraints, costs, and constraint violations can be mapped to various manufacturing metrics, such as energy or material cost [111].

The proposed PTA-MPC is tested with multiple recursively feasible optimization horizons (N_{spec} and N_{mpc}). The lower bounds on horizons are evaluated offline by following the steps given in Algorithm 2. Three different optimization horizons are considered for (S2) (see Table 5.2). The lower bound on the spec horizon for (S2) is evaluated as $\hat{N}_{spec} = 1$ and the corresponding lower bound for the N_{mpc} is evaluated as 4. The lower bound for

the spec horizon in (S3) is $\hat{N}_{spec} = 1$ as well. However, due to the new deadline constraint introduced in (S3), the lower bound on the N_{mpc} is now 6. Currently, the calculation of the lower bounds on the horizons are performed offline and kept constant throughout the simulation. However, future work will look to identify when re-computation of the lower bounds for the horizon is needed.

A summary of the horizon lengths and the corresponding simulation results are presented in Table 5.2. All simulations use the same cost function to evaluate the minimum cost optimal path in (5.7), starting from the starting buffer and ending at the target buffer. Note that the mean and standard deviation of computation time in Table 5.2 refer to solving a single MPC problem for (S2) and (S3). The mean and standard deviation are averaged over the 8 MPC loops needed to reach the target buffer (q^{13}) from start (q^0).

5.2.5.2 Results and Discussion

A task graph plot illustrating the results from three simulations for comparison is given in Fig. 5.6. The open loop controller from (S1) is compared to the MPC from simulations (S2) and (S3) both with $N_{spec} = 2$ and $N_{mpc} = 8$. As shown in the figure, the utilization of the closed-loop, model predictive controller improves the robustness of the system when disturbances are captured. The primary difference between the discrete transitions taken by the simulations is that the optimal open loop controller in (S1) chooses to take AGV3 to the target buffer, while the MPC simulations in (S2) and (S3) take AGV2. This is because in the nominal system, while AGV3 is more costly to operate, it is faster, yielding an overall lower cost in the open loop controller's plan. However, the open loop strategy does not account for the disturbance, which occurs during the product's machining, and causes AGV3 to slow without reducing operating cost, therefore (S1) uses more energy and finishes later than the MPC controllers as shown in Fig. 5.6. The MPC controller in (S2) begins with this path planned as well. However, it is updated with the information of AGV3's damage after it takes the discrete transition out of the AM machine, and in its subsequent optimization is able to modify its plan to take AGV2. This update on the modeling information is provided by the digital twins in the central controller of the SDC-AM framework presented in Section 5.1. By utilizing functional state information of the resources on the plant floor, the central controller detects the failure that causes the AGV3 to operate suboptimally. Subsequently, this information is reflected in the PTA model of the MPC controller, and the controller is then able to react to this disturbance. This shows a clear benefit of utilizing a closed-loop controller to improve the robustness to time-domain disturbances in the system. The open-loop scenario in (S1) is also implemented using UPPAAL CORA to verify that the optimal solution of (5.7) is identical to the CORA solution.

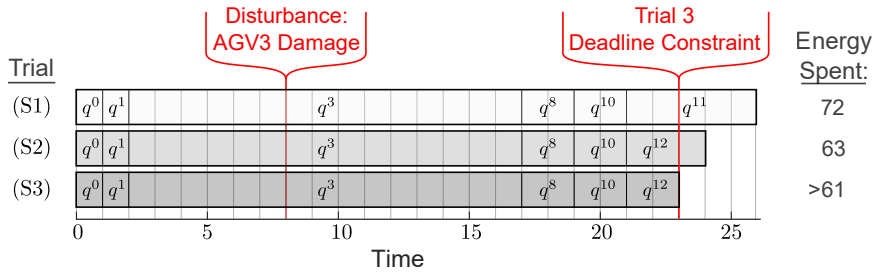


Figure 5.6: Task graph plot for each simulation, along with the energy spent in each simulation, according to the nominal location cost rates. The horizon lengths N_{spec} and N_{mpc} are 2 and 8, respectively for both (S2) and (S3).

5.2.5.3 The benefit of using soft constraints

In (S3), the MPC controller successfully meets the hard constraint of its production deadline by violating the softened constraint $c_l(q^{12}) \geq 3$, instead traversing the edge e^{17} while $c_l(q^{12}) = 2$. While the increased AGV speeds are available for short times, it is not desirable to use the AGVs at those speeds for longer periods, thus the nominal constraints are set for the desired working conditions. A controller without soft constraints, such as (S1) or (S2), would fail to identify an appropriate control action for the system (the problem given in (5.7) would be `UNSAT`).

5.2.5.4 The effect of different horizon lengths

Table 5.2 presents the simulation results and the computational times for all of the simulations in the case study. For each of simulation trial, horizon lengths, values of the two slack variables at the last MPC iteration, mean (μ_t) and one standard deviation (σ_t) of the computational times in milliseconds, global clock valuations of the corresponding states when the desired events in Σ^d are satisfied, and the inclusion of the deadline constraint are presented in Table 5.2. The simulations (S2) and (S3) are run according to Algorithm 2 given in Section 5.2.4.3. In both simulations (S2) and (S3), the MPC loop is run 8 times to reach the target buffer (q^{13}) from start (q^0). The open loop controller in (S1) is implemented in UPPAAL CORA. Computational times of eight repetitions of the same open-loop optimization problem in (S1) are recorded using the command line interpreter of UPPAAL CORA [23].

The MPC simulations in (S2) and (S3) have the same location trajectory with the exception of (S2) with $N_{spec} = 1$ and $N_{mpc} = 4$. Due to the short MPC horizon length in this simulation, the MPC chooses to go to AM2 at location q^6 and therefore satisfies the desired event σ_1^d at $c_g(\boldsymbol{q}^7) = 12$. However, using AM2 results in a longer wait at AGV1 with the

transit path 2, denoted by location q^9 with the constraint $c_g(\boldsymbol{q}^9) \geq 24$, when compared to the transit path 1 of AGV1, denoted by location q^8. Consequently, the simulation (S2) with $N_{spec} = 1$ and $N_{mpc} = 4$ satisfies the second desired event σ_2^d at $c_g(\boldsymbol{q}^{13}) = 31$, which results in a sub-optimal overall trajectory. All the rest of the MPC simulations with longer horizons are able to consider this trade-off and choose to go through AM1 at location q^3 which results in a smaller incurred path cost and an optimal overall trajectory. The MPC in (S3) utilizes the soft constraint γ_2 on q^{12} to violate the constraint $c_l(\mathbf{q}^{12}) \geq 3$ by 1 and meet the deadline constraint so that the part is at the target buffer (q^{13}) at $c_g(\boldsymbol{q}^{13}) = 23$.

The computational times in Table 5.2 show that the use of the proposed MPC scheme without soft constraints (S2) compares well, in terms of computation time, to the use of UPPAAL CORA without constraint violations (S1). The computational times increase with the introduction of slack constraints for the MPC and remain approximately around 0.1 seconds for (S3). The increase in computational time due to the slack variables is an expected outcome since the complexity of the optimization is increased.

The contributions presented so far in this chapter are summarized in the following. Section 5.2 presents a closed-loop run-time production scheduling controller for spatially distributed systems. We demonstrate the utility of the closed-loop controller from Section 5.2 in the context of the SDC-AM framework presented in Section 5.1. Utilizing the centralized view of the AM Fleet provided by the digital twins in the central controller of SDC-AM (see Section 5.1), the decision-maker is able to generate an up-to-date PTA model of the available resources in the AM-Fleet to then schedule production. By utilizing the digital twins, the closed-loop controller is able to react to disturbances occurring during the production at a system-level, while the open-loop controller without the up-to-date information from the digital twins in run-time (during the scheduled production) fails to meet deadline constraints and performs suboptimally.

5.3 Knowledge Transfer Application for Layer-to-Layer Spatial Control

In this section, we present another use case for the centralized SDC-AM framework. We focus on utilizing the knowledge base in the central controller (see Fig. 5.1) to improve the performance of the layer-to-layer controllers presented in Section 4.1. More specifically, we present how the data stored in the knowledge base from the past prints on the same AM resources, as well as data from other AM resources printing similar geometries, may be utilized to improve the tracking performance of the layer-to-layer controller. This application is enabled by utilizing proper transformation maps between the dynamics of two

different AM processes, which are assumed to be available to the central controller within the knowledge base. The rest of this section formally introduces the problem formulation and provides simulation studies to illustrate the benefit of the proposed approach.

The *main contribution of this chapter* is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets. Within this main contribution, this section focuses on the development of novel knowledge transferring controllers for improved layer-to-layer spatial tracking performance. The specific contributions of this section are

(*C3*-7) A knowledge reuse formulation to utilize previous process data of the same AM resource to improve tracking performance.

(*C3*-8) A preliminary knowledge transfer approach that utilizes the models of two different AM processes to transform the measured data of one process to be used by a second process.

5.3.1 Problem Formulation

Consider the LLSV process given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k, \tag{5.8}$$

which is defined previously in Chapter III. We consider the LP-ILC controller presented in Section 4.1. As previously discussed in Section 4.1, we approximate the gradient information either through the data gradient if we have previously encountered the spatial dynamics in the current layer, e.g., within a layer dynamics group. The controller update (see (4.5) and (4.8)) to track a desired spatial height reference $\boldsymbol{r}_k$ utilizing the data gradient within a single layer dynamics group is given as

$$\boldsymbol{u}_{k+1} = \Pi_{\mathcal{U}}^{\boldsymbol{P}_{k+1}}(\boldsymbol{u}_k - \alpha \boldsymbol{P}_{k+1}^{-1}(\bar{\boldsymbol{B}}_{k+1}^T \boldsymbol{Q}(\boldsymbol{\xi}_{k+1} - \boldsymbol{\eta}_{k+1}) + \boldsymbol{S}\boldsymbol{u}_k)), \tag{5.9}$$

where $\bar{\boldsymbol{B}}_{k+1}$ is the model of the spatial input dynamics, $\boldsymbol{\xi}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{A}_k \boldsymbol{x}_k = \boldsymbol{B}_k \boldsymbol{u}_k$ is the measurement information that we utilize in place of $\boldsymbol{B}_{k+1}\boldsymbol{u}_k$ since $\boldsymbol{B}_k = \boldsymbol{B}_{k+1}$ and $\boldsymbol{\eta}_k = \boldsymbol{r}_{k+1} - \boldsymbol{A}_k \boldsymbol{x}_k$ (see Section 4.1).

Furthermore, we utilize the model of the spatial dynamics for the switch layers between the layer dynamics groups, where we do not have previous measurements of the spatial dynamics corresponding to the current layer, i.e., we do not have measurement information from previous layers since $\boldsymbol{B}_k \neq \boldsymbol{B}_{k+1}$. Due to the model mismatch, switch layers have an

increased error when compared to the layers within a layer group, as illustrated in the case study of Section 4.1. In this section, we propose methods to improve the tracking error of the controller for the switch layers, by utilizing measurements from either previous prints of the same part geometry on the same AM resource, or other similar AM resources in the AM Fleet.

It is important to note that by utilizing the models and measurement data from the past runs of the same AM process or various AM processes in an AM fleet, we may develop methods to improve the tracking performance at not only the switch layers but for the other layers in the process as well. Run-to-run controller methods utilizing the data from past runs may be developed for improved tracking performance overall. In this section, we narrow our focus to only the switch layers as a compelling example of how past data may improve tracking performance in an AM Fleet. Additionally, the developments focusing on the switch layers provide us with a complementary approach to the controllers developed in Section 4.1. The controller heuristics developed in this section are indicative of how similar methods may be developed to improve spatial reference tracking performance and part quality in an AM Fleet by utilizing prior measurements.

We start by considering the case where we utilize measurements from a previous print of the same geometry on the same AM resource. Note that utilizing the knowledge base in the central controller of the SDC-AM (see Fig. 5.1), measurement and monitoring data from previous print jobs are available to the central controller. We name the input and measurement data available from the previous print job on the same AM process as *recorded self-data*. In this case, the spatial dynamics of the switch layers in the layer-to-layer process are identical between the current print and a previous print of the same geometry. Therefore, suppose we have measurements from a previous print job of the same geometry on the same AM process and let k' denote a dynamics switch layer. That is, for a previous print job, at the dynamics switch layer k', we used the input $\bar{\boldsymbol{u}}_{k'}$ at the state $\bar{\boldsymbol{x}}_{k'}$ to print the next layer $\bar{\boldsymbol{x}}_{k'+1}$. Then we utilize the control update

$$\boldsymbol{u}_{k'} = \Pi_{\mathcal{U}}^{\boldsymbol{P}_{k'}}(\bar{\boldsymbol{u}}_{k'} - \alpha \boldsymbol{P}_{k'}^{-1}(\bar{\boldsymbol{B}}_{k'}^T \boldsymbol{Q}(\bar{\boldsymbol{\xi}}_{k'} - \boldsymbol{\eta}_{k'}) + \boldsymbol{S}\bar{\boldsymbol{u}}_{k'})), \tag{5.10}$$

where we denote $\bar{\boldsymbol{u}}_{k'}$ as the input and $\bar{\boldsymbol{\xi}}_{k'} = \bar{\boldsymbol{x}}_{k'+1} - \boldsymbol{A}_{k'}\bar{\boldsymbol{x}}_{k'}$ as the measurement of the recorded self-data of the same layer k' from the past print job. Note that in this case, the control update (5.10) is similar to an iterative learning control update for the spatial dynamics of the same layer (layer k') between two different print runs. Similarly, the proposed controller update (5.10) can be viewed as a run-to-run controller for the AM resource, repetitively printing the same geometry in each run. As a result, the central

controller is able to utilize the data from the knowledge base to identify the previous prints of the same product on an AM resource to improve controller performance.

While the control update (5.10) provides a convenient way to utilize past measurements for the same AM resource, the method requires the availability of recorded self-data, i.e., data collected on the same machine. In many cases, it may be desirable to utilize measurements between similar AM processes to enable knowledge transfer and improve fleet performance. As an example, if geometry is previously printed by a resource in the AM Fleet, we want to utilize the process data from that print to improve the controller performance of a different AM resource so that we have the first print with minimal defects. To do this, consider the spatial dynamics of two AM processes printing the same geometry at each layer, given by

$$\boldsymbol{x}_{k+1}^1 = \boldsymbol{A}_k^1 \boldsymbol{x}_k^1 + \boldsymbol{B}_k^1 \boldsymbol{u}_k^1, \tag{5.11}$$

$$\boldsymbol{x}_{k+1}^2 = \boldsymbol{A}_k^2 \boldsymbol{x}_k^2 + \boldsymbol{B}_k^2 \boldsymbol{u}_k^2, \tag{5.12}$$

where the superscripts 1 and 2 denote the two AM processes, respectively. Let k' again denote a dynamics switch layer for the geometry. Furthermore, since the matrix $\boldsymbol{A}$ relates the spatial height information between layers (see Chapter III and Chapter IV), we adopt the same assumption from Section 4.1 and assume that the $\boldsymbol{A}$ matrices are known and identical for both processes, i.e., $\boldsymbol{A}_k^1 = \boldsymbol{A}_k^2$, for all k. Here, we assume that the switch layer has been previously printed by process 2 (5.12), therefore we have the data $\boldsymbol{u}_{k'}^2, \boldsymbol{\xi}_{k'}^2$ available in the knowledge base of the SDC-AM central controller. In contrast to the recorded self-data and the controller update in (5.10), here we have non-identical spatial input dynamics between the two processes, therefore we cannot use $\boldsymbol{u}_{k'}^2, \boldsymbol{\xi}_{k'}^2$ with the controller update (5.10). However, note that since the state matrices encode the deposition path of the print and we consider the same geometry printed by both processes, we know that the spatial dynamics of layer k' correspond to the same spatial reference for both processes.

We utilize two linear maps to transform the data $\boldsymbol{u}_{k'}^2, \boldsymbol{\xi}_{k'}^2$ appropriately to be used for the control evaluation to print the the switch layer $k'+1$ of the process 1 (5.11). Note that due to the model mismatch in the spatial dynamics in both processes and the utilization of measurements in the controller evaluation, e.g., (5.9), the transformation of the data from process 2 to process 1 is an approximation of the data in the context of the transformed process.

The control update (5.9) evaluates a control input to minimize tracking error by utilizing the spatial input dynamics model of the process, i.e., $\bar{\boldsymbol{B}}_k$. Therefore, we utilize the models of two processes to evaluate an approximate input signal that corresponds to $\boldsymbol{u}_{k'}^2$ for the

first process (5.11). To transform the input data, we utilize the least squares solution given by

$$\boldsymbol{\mu}^1_{k'} \leftarrow \underset{\boldsymbol{\mu}}{\operatorname{argmin}} ||\bar{\boldsymbol{B}}^1_{k'}\boldsymbol{\mu} - \bar{\boldsymbol{B}}^2_{k'}\boldsymbol{u}^2_{k'}||, \tag{5.13}$$

where $\bar{\boldsymbol{B}}^1_{k'}$ and $\bar{\boldsymbol{B}}^2_{k'}$ denote the models of the dynamics switch layers for the two processes, respectively. To transform the measurement data $\boldsymbol{\xi}^2_{k'}$, we utilize a linear map given by

$$\boldsymbol{T}_{1,2} = \bar{\boldsymbol{B}}^2_{k'}(\bar{\boldsymbol{B}}^1_{k'})^+, \tag{5.14}$$

where $\boldsymbol{M}^+$ denotes the left pseudoinverse of a matrix $\boldsymbol{M}$, such that $\boldsymbol{M}^+\boldsymbol{M} = \boldsymbol{I}$. Then, for an input vector $\boldsymbol{v} \in \mathbb{R}^{n_u}$ we have that $\boldsymbol{T}_{1,2}\bar{\boldsymbol{B}}^1_{k'}\boldsymbol{v} = \bar{\boldsymbol{B}}^2_{k'}\boldsymbol{v}$. Thus, we estimate the approximate measurement data for the first process as

$$\boldsymbol{\zeta}^1_{k'} = (\boldsymbol{T}_{1,2})^+\boldsymbol{\xi}^2_{k'}. \tag{5.15}$$

We utilize $\boldsymbol{\mu}^1_{k'}$ and $\boldsymbol{\zeta}^1_{k'}$ to arrive at the update equation for the input to print the switch layer given by

$$\boldsymbol{u}^1_{k'} = \Pi^{\boldsymbol{P}_{k'}}_{\mathcal{U}}(\boldsymbol{\mu}^1_{k'} - \alpha\boldsymbol{P}^{-1}_{k'}(\bar{\boldsymbol{B}}^T_{k'}\boldsymbol{Q}(\boldsymbol{\zeta}^1_{k'} - \boldsymbol{\eta}_{k'}) + \boldsymbol{S}\boldsymbol{\mu}^1_{k'})), \tag{5.16}$$

which we name *the knowledge transfer update*. We utilize (5.16) for the switch layers of process (5.11) to improve controller performance. Notice that if we have $(\boldsymbol{B}^1_k, \bar{\boldsymbol{B}}^1_k) = (\boldsymbol{B}^2_k, \bar{\boldsymbol{B}}^2_k)$, the update (5.16) becomes the update in (5.10). Therefore, the knowledge transfer update (5.16) generalizes (5.10). Next, we illustrate the two controller updates in a simulation study.

5.3.2 Simulation Setup

We utilize the simulation setup from the case study of Section 4.1 for the case study here. The same inverted pyramid geometry given in Fig 4.2 is utilized, where the switch layers are on layers 6 and 16. We study two cases. The first case presents knowledge reuse through recorded self-data of the same process and the second case presents knowledge transfer between two processes.

Each spatial process has a controller model which is an approximation of the true process model, similar to the setup given in Section 4.1. We utilize the LP-ILC controller for layer-to-layer process control and utilize the knowledge transfer update only on the switch layers. In this scenario, we assume the central controller can evaluate the appropriate lin-

ear maps through the models and measurements available in the central controller and the knowledge base of the SDC-AM framework. We present the simulation results in the next subsection where we compare the performance of the recorded self-data update and transferred data update between two processes. We present two cases, one where we have a low model mismatch and another one with relatively high model mismatch.

5.3.3 Results and Discussion

The performance of the knowledge transfer is affected by the mismatch between the process model and true process dynamics (i.e., between $\bar{\boldsymbol{B}}$ and $\boldsymbol{B}$), since the mapping between different processes utilize the model information to derive necessary transformation maps. We present performance comparisons for two cases for two different levels of model mismatch between the models and the corresponding processes. Figure 5.7 presents the comparison of the knowledge transfer cases for the low model mismatch between $\bar{\boldsymbol{B}}$ and $\boldsymbol{B}$, whereas Fig. 5.8 presents the case with relatively high model mismatch. In both figures, the controller indicated with *Model* is the LP-ILC update with the model gradient for the switch layers (see Section 4.1). The controller indicated with *Self-Data* utilizes the recorded self-data from a previous print of the *same geometry on the same AM resource*, and the controller update (5.10) for the switch layers 6 and 16. The controller indicated with *Transfer-Data* utilizes data from a previous print of the *same geometry on a different AM resource in the AM Fleet*, and the controller update (5.16) for the switch layers 6 and 16.

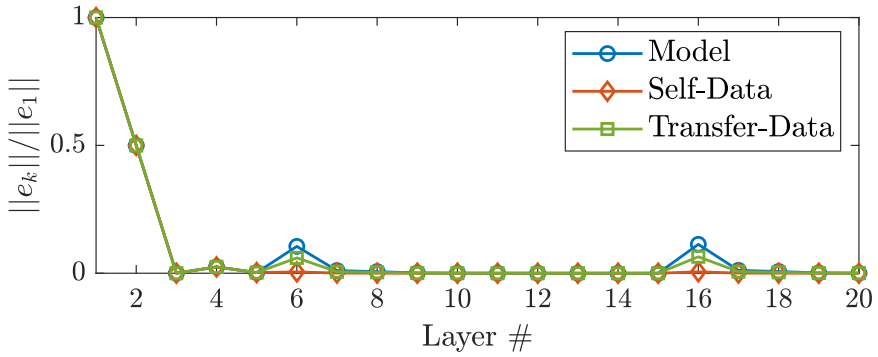


Figure 5.7: Comparison of the LP-ILC controller (denoted as Model) with the model update on the switch layers (6 and 16) for the cases of self-data and transfer-data with *low model mismatch*.

The results in Fig. 5.7 demonstrate the performance of both knowledge transfer approaches presented in this section. The self-data controller provides the most improvement and reduces the tracking error by approximately 95% when compared to the model gradi-

ent of the LP-ILC controller. The transfer data is taken from the simulation of a secondary process running the LP-ILC controller with model gradient update for the switch layers. The model mismatch of the LP-ILC controller with the simulated real process is the same for both the primary and the secondary processes. In this case, the transfer-data controller provides approximately 42% improvement of the tracking error for the switch layers.

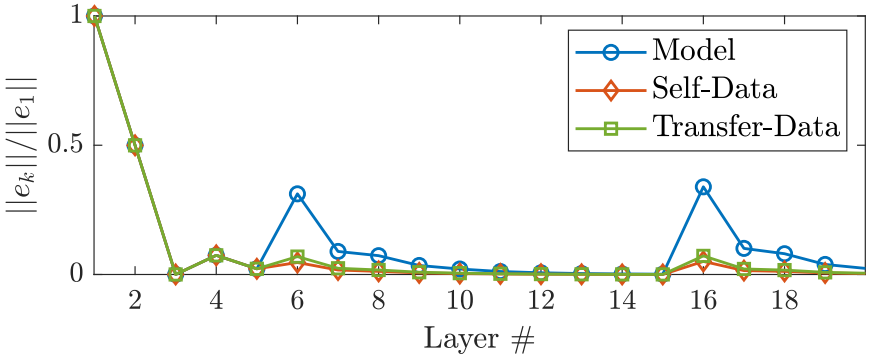


Figure 5.8: Comparison of the LP-ILC controller (denoted as Model) with the model update on the switch layers (6 and 16) for the cases of self-data and transfer-data with *high model mismatch*.

Figure 5.8 presents results for the high model mismatch case. Due to the high model mismatch, we see that the nominal LP-ILC controller with the model gradient has higher errors for the switch layers 6 and 16 in Fig. 5.8. In this case, we see that the transfer-data provides performance much closer to the self-data when compared to the low model mismatch case. The self-data controller provides approximately 85% improvement over the model gradient LP-ILC controller at the switch layers. The transfer-data controller provides approximately 78% improvement over the model gradient LP-ILC controller at the switch layers. The performance improvement with the knowledge transfer is a function of the similarity between the true process dynamics of the primary process and the secondary process where the transferred data is taken from. Due to the model mismatch and uncertainties, there may be cases where the transferred data taken from a different process outperforms the self-data if the process model of the secondary process provides a better approximation of the true process dynamics of the primary process.

The results suggest that knowledge transfer provides a performance improvement for the switch layers by utilizing the data available in the central controller. Note that we compare the utilization of the knowledge transfer for a single print on an AM resource. In practice, it is desirable to utilize transfer data, if available, for a new geometry on a primary AM resource to improve the tracking performance, which in turn improves the part functionality. After a new geometry is printed on the resource, self-data may be utilized

to further improve the tracking error. Additionally, run-to-run control methods with the self-data may be employed to systematically improve the tracking error of the process in subsequent prints of the same geometry. Similarly, the knowledge transfer and data reuse may be extended for improving the spatial reference tracking performance for all the layers for the AM processes in the AM Fleet. A combination of both methods presented here may provide improved performance and robustness. The knowledge transfer methods presented in this section are applications of the LP-ILC controller given in Section 4.1. Thus, they are developed as heuristics tools to improve the model gradient performance of the controller. However, future research may formalize the use of knowledge transfer methods presented here to develop controllers that utilize the data available from previous runs of the same resource and different resources in the AM Fleet to present novel controllers that improve process performance within the fleet. Further remarks on this topic are provided in Chapter VII.

5.4 Chapter Conclusions

This chapter presents a system-level modeling and control framework for spatially distributed AM Fleets. The main contribution of this chapter is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets (*C3*). A centralized framework for run-time monitoring and analysis of the resources in an AM Fleet is presented and a conceptual example on system-level anomaly detection is given. In Section 5.1, the centralized controller provides a foundation for scalable system-level monitoring, analysis, and control for AM Fleets. The use of digital twins in the central controller enables practical run-time control applications like the one presented in Section 5.2. We present a novel model predictive control approach for closed-loop control of spatially distributed systems modeled by priced timed automata. The case study at the end is motivated by a rescheduling problem in an AM Fleet. The examples in the case study show how a closed-loop scheduling approach enabled by the centralized controller and the digital twins outperform a static production schedule and are able to provide further robustness to disturbances and constraints. In Section 5.3 we present a knowledge transfer application for the layer-to-layer closed-loop controller presented in Section 4.1. We develop a simple methodology to utilize the models and data available in the central controller presented in Section 5.1 for improved controller performance. The results suggest that the presented heuristics may lead to improved layer-to-layer tracking performance in an AM Fleet scenario and provide a foundation for knowledge transferring control applications for AM Fleets.

The use of digital twins for run-time monitoring and analysis of physical processes is a key technological enabler for the results presented in this chapter. However, for large-scale systems with multiple digital twins, the architecture and implementation details for the digital twins become another challenge. Therefore, an important question is on how to efficiently design and implement digital twins for a manufacturing system to enable monitoring, analysis, and control applications presented in this chapter and the previous chapters. To address this, in the next chapter we present a digital twin framework for run-time monitoring and analysis for general cyber-physical manufacturing systems, which includes AM processes as specific examples.

CHAPTER VI

Online Monitoring and Analysis

In this chapter, we provide an online monitoring and analysis framework that utilizes purpose-driven digital twins of a cyber-physical Additive Manufacturing (AM) process. The main contribution of this chapter is a digital twin framework that is flexible and extensible to incorporate various models and data structures for run-time analysis of cyber-physical manufacturing systems (*C4*). We start by providing a general and extensible digital twin framework for cyber-physical manufacturing systems and demonstrate the framework for anomaly detection and cyber-attack detection applications for a spatiotemporal AM process. Then, we proceed by specializing the framework in the context of a spatiotemporal AM process and present an application of spatial monitoring for AM processes to enable performance monitoring, and in-situ/ex-situ analysis of expected part performance. Both applications are implemented on an off-the-shelf 3D printer to illustrate the applicability of the presented solutions to real-world applications. The general-purpose DT framework and the case studies presented in this chapter provide a foundation for monitoring AM processes. By presenting applications both on online analysis for anomaly detection and performance analysis, we provide a comprehesive solution for monitoring spatiotemporal processes utilizing online sensor data and digital twins. The contents of this chapter are presented in [12], currently under review.

6.1 Cyber-Attack Detection Digital Twins for Cyber-Physical Manufacturing Systems

We start by motivating online measurement applications in the context of modern manufacturing systems. Smart manufacturing (SM) is an increasingly important paradigm that promotes the use of run-time and historical data collected via onboard and additional Internet of Things (IoT) sensing in the manufacturing system to inform decisions for the

plant floor [55, 102, 138, 182]. Plant floor decisions include production scheduling and dispatch, predictive maintenance, anomaly detection, and process control. The decisions are implemented, often at run-time, on the resources in the manufacturing system to minimize disruptions, by integrating cyber and physical systems in modern manufacturing resources, allowing them to be reconfigurable and robust in response to disturbances. This framework of data-enabled decision-making coupled with cyber-physical manufacturing resources is commonly referred to in the industry as Cyber-Physical Manufacturing Systems (CPMS) [121, 205].

As CPMS become more complex, the decision-making process becomes increasingly challenging. Additionally, decision-making logics that are designed for the nominal conditions of a CPMS may underperform or fail to detect certain abnormalities in the system due to complex interdependencies between multiple resources in a manufacturing process [125, 206]. As described in Chapter V, to address the issue of monitoring complex systems, digital twin (DT) technology has emerged as a fundamental tool for *twinning* physical resources within a CPMS, providing additional analysis capabilities and delivering insights on the run-time system in addition to the as-designed conditions. The potential and flexibility of DT technology have generated significant research interest from academia and industry on applying DTs for supporting SM in practice, hailing DTs as the cornerstone technology for realizing SM [80, 138, 142, 166, 183]. While DTs are referred to as tools to monitor and control manufacturing resources in an AM Fleet in Chapter V, in this chapter we develop further details on DT architectures, purposes, and implementations for a general cyber-physical manufacturing system and demonstrate our developments with experimental results.

An important implication of the cyber-physical nature of CPMS is its vulnerability to cyber-attacks. As the cyber components are now linked to their physical counterparts, attacks that are initiated in the cyber domain may cause harm and damage to the physical manufacturing resource, product, or even the human workers that are interacting with the manufacturing resource [27]. Therefore, we need effective methods to detect cyber-attacks in a CPMS, which is not a trivial task for several reasons. The system undergoes expected abnormalities, namely, physical degradation, anomalies, and faults by nature. Expected anomalies may be hard to distinguish from a carefully targeted cyber-attack (e.g., one with malicious intent) as these attacks often mimic the expected anomalous behavior to deceive the decision-making logic. Furthermore, cyber-attacks may originate from non-malicious intent (e.g., mis-calibration, version mismatch, etc.), which also causes difficulties in distinguishing them from anomalies. Additionally, run-time process controllers change the setpoints and control inputs of the resources, making the cyber-attack detection task even

more challenging. The main contribution of this chapter is to propose a DT-based method to address the crucial challenge of *cyber-attack detection for CPMS*.

Specifically, a DT is a software replica of a physical thing (i.e., the physical twin) and has the purpose of impacting an aspect of the physical twin and its environment in a positive way through utilizing models, data analytics, and subject matter expertise (SME) [138]. Therefore, DTs are fundamental information technology (IT) components, which bring enhanced capabilities to CPMS. As DTs themselves are software entities, they may also bring along the additional burden of vulnerabilities that could compromise the physical components through cyber-attacks. In this work, we focus on utilizing DTs as a solution for cyber-attack detection rather than considering the cyber-security of the proposed DTs themselves.

Traditional corporate cybersecurity control implementations are not always possible or feasible within Industrial Control Systems (ICS) network environments, and improper implementations can have unintended and disastrous consequences [180]. Typically, passive monitoring capabilities for supporting threat detection within ICS network environments are implemented as risk management strategies within these networks. However, countering the growing threats facing ICS environments requires both passive and active monitoring [37]. These capabilities applied at the lowest levels could be utilized to detect signs of anomalous behavior resulting from cybersecurity threat activity.

An effective cybersecurity approach requires a cross-functional cybersecurity team consisting of IT staff, control engineer, control system operator, network and system security expert, a member of the management staff, and a member of the physical security department who work together and share domain knowledge and experience to evaluate and mitigate risk to the ICS [180]. The existing literature has focused on cyber-attack detection and mitigation for CPS by leveraging cross-domain knowledge [50, 154, 175, 193]. While highly effective methodologies for cyber-attack detection are presented, they are often customized for a specific system or operation and thus do not provide ways to extend the proposed frameworks. Additionally, the majority of the existing literature considers the cyber-attack detection problem for a CPS with no anomalies, which is often unrealistic in practical scenarios. Recent work provides a methodology to detect and differentiate specific types of cyber-attacks from equipment failure [116]. The method in [116] utilizes specific models and assumptions, which may be difficult to extend and scale for a general CPMS with various types of attacks. Therefore, there exists an opportunity to address the aforementioned shortcomings and support both manufacturing and cybersecurity automation enhancements by leveraging common technological enablers such as DT and the Industrial Internet of Things (IIoT).

Previous research, such as [146], demonstrates techniques to utilize Industry 4.0 technologies and methodologies such as IIoT, Industrial Internet of Services (IIoS), and DTs to create smart factories and establish Knowledge as a Service manufacturing processes to monitor product or service quality. We build on the previous literature and investigates utilizing cybersecurity DT technology to monitor devices and processes for abnormal conditions that could be indicators of cybersecurity events in the context of run-time controller inputs and anomalies. These cybersecurity DTs could be implemented to support a passive/active hybrid approach to protect the ICS environment from advanced device-level risks. Our method is capable of working with existing architectures for anomaly detection in industrial systems and enables scalability to multiple resources in a CPMS thanks to its DT-centric design. *The main contribution of this chapter* is a digital twin framework that is flexible and extensible to incorporate various models and data structures for run-time analysis of cyber-physical manufacturing systems (*C4*). Within this main contribution, specific contributions of this chapter are [12]:

(*C4*-1) An extensible digital twin-based solution for cyber-attack detection in CPMS, capable of integrating with existing solutions in practice.

(*C4*-2) A methodology to distinguish cyber-attacks from expected anomalies for a controlled cyber-physical system.

(*C4*-3) A novel experimental demonstration of the proposed method on an off-the-shelf 3D printer to demonstrate the effectiveness, flexibility, and scalability of the proposed approach.

The rest of the chapter is organized as follows. Section 6.2 provides preliminary definitions of concepts used in this work and provides the formal problem definition. Section 6.3 provides the framework architectures for the CPMS with all of the proposed DTs for cyber-attack detection. Section 6.4 presents the proposed DT-based cybersecurity approach and Section 6.5 demonstrates the experimental implementation and results. Furthermore, we specialize the presented DT framework in the context of spatiotemporal AM processes and present additional experimental results in Section 6.6.

6.2 Preliminaries and Problem Statement

In this section, we first present definitions and background knowledge. Then, we formally state our problem in the context of the introduced formal concepts.

6.2.1 Classification of Abnormality Types

To address the challenge of cyber-attack detection for a CPMS, we first present a classification of anomalies, attacks, and faults in the context of this chapter. Figure 6.1 presents various types of attacks and anomalies for a CPMS resource. Each item in Fig. 6.1 that is inside the box (output measurable effect on the system) is an event that results in an effect on the physical process that is categorized as the corresponding set (e.g., a failed sensor event results in a fault that is a subset of anomalies and output measurable abnormalities). The representation in Fig. 6.1 is inspired by [125], where types of anomalies and faults for smart manufacturing systems and their detection mechanisms are discussed in detail.

Definition 6.1 (Anomaly [125])**.** An occurrence that is different from what is standard, normal, or expected

Definition 6.2 (Fault [125])**.** An anomaly that is related to an unwanted situation and may be associated with failure, malfunction, or quality degradation.

Thus, an *anomaly (fault) detection* (A(F)D) mechanism detects the result or onset of an anomaly (failure) event. Some events such as network delays and controller errors may result either in failures that would be classified as faults, or only anomalous behavior that does not necessarily result in failure.

Definition 6.3 (Cyber-attack [100])**.** The realization of some specific threat that impacts the confidentiality, integrity, accountability, or availability of a computational resource.

We simply use the term *attack* instead of cyber-attack unless specified otherwise in further discussions. The "normal" behavior of the system is defined by the nominal operation of the system without any anomalies or attacks, and we use the term *abnormal* for all other system behavior. Thus, we say that a system is abnormal if its output measurements are not consistent (evaluated by a classifier) with the measurements from the nominal operation without any anomalies or attacks. Additionally, we say an input has an *output measurable effect on the system* if given sufficient measurements of the output, it is possible to determine the effect of a (possibly exogenous) input on the output, which could be immediate or at a later time.

6.2.2 Problem Statement

The set of attacks depicted in Fig. 6.1 has three distinct sub-spaces; attacks that are not output measurable (e.g., side-channel attacks), attacks that are output measurable but do

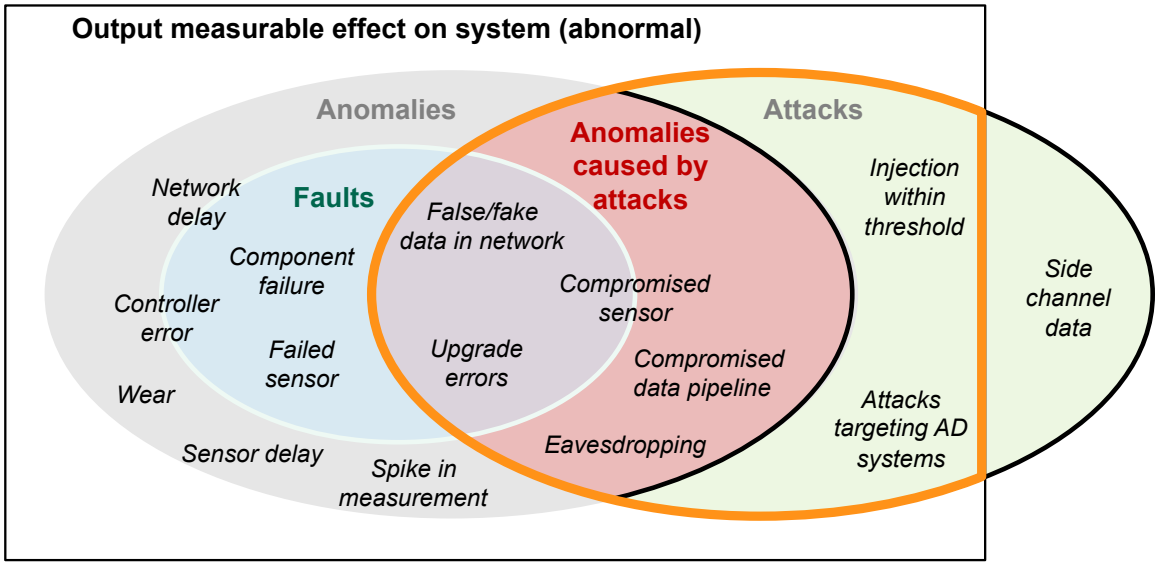


Figure 6.1: Illustration of the subspaces for observable abnormalities, anomalies, faults, and attacks considered in this work. The scope of this chapter is outlined with orange borders in the figure. AD: Anomaly detection.

not necessarily cause anomalies, and attacks that are output measurable and cause anomalies. Within the scope of this work, we are focusing on attacks that have output measurable effects on the system. Thus, the goal of our proposed DT is to detect the aforementioned output measurable attacks. Since anomalies are not necessarily caused by attacks, an effective methodology should be able to distinguish anomalies caused by attacks from the inherent anomalies that we expect to see in the system, which we term as *expected anomalies* in further discussions.

Remark 6.4. *Within the context of attacks on CPMS, we do not necessarily require malicious intent. For example, we consider a miscalibrated sensor as a non-malicious attack. Attacks in the context of this chapter denote all types, regardless of the intent.*

Additionally, we note that the physical system is a controlled CPMS resource, thus the operational characteristics of the system may be modified by a controller. Transient behavior and multiple setpoint references must be analyzed in run-time to mitigate false-positives in attack detection. Furthermore, due to our assumption on the presence of anomalies in the system, expected anomalies and attacks may be *inseparable* in the output as they may result in similar output effects.

Remark 6.5. *We note that our work differs from the past literature as we do not rely on a specific system model or analysis tool to provide our results. Instead, we present a general-*

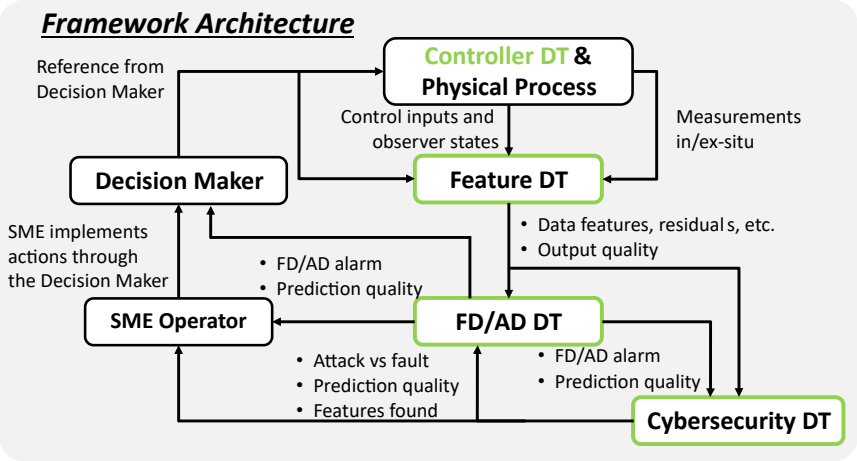


Figure 6.2: The framework architecture including all the DTs and physical components. The architecture provides a basis for further extensions based on the needs of a certain physical process. The decision-maker in the architecture may be autonomous or purely advisory depending on the application domain. The color green indicates the DTs in the framework.

purpose DT framework where data-driven and physics-based information about the CPMS may be utilized efficiently to detect cyber-attacks in an extensible and systematic manner.

We formally state our problem as *"How do we develop an effective methodology to identify cyber-attacks with output measurable effects in the presence of anomalies in a controlled CPMS?"* In this work, we propose a DT-based method to address this problem and present case studies to illustrate our proposed method.

6.3 Proposed DT-Based Methodology

In this section, we present the proposed methodology to utilize DTs for cyber-attack detection in the context of anomalies and controllers in the system. We start by introducing the framework architecture with some of the existing DTs in place. After we situate our DTs in this context, we introduce the proposed methods for anomaly and attack detection.

6.3.1 Framework Architecture

Figure 6.2 illustrates the architecture of the controlled CPS framework with the proposed DTs considered in this work. To avoid confusion of terminology, we use the term *process* instead of *system* in this chapter (e.g., a physical system is a physical process). The physical process in the top block is the manufacturing process we control and analyze utilizing the proposed framework. The physical process may be discrete or continuous

depending on the application domain. The execution of discrete manufacturing processes is often considered in terms of *runs* where a single unit (or batch) is manufactured. We consider the data collected during the run as in-situ and the data collected after a run is completed as ex-situ (e.g., for post-process quality control). The framework architecture presented in Fig. 6.2 is largely based on augmenting existing feature-based anomaly and fault detection systems in the literature (e.g., [15, 99, 101, 159, 193]).

In the rest of this section we provide details of the blocks in Fig. 6.2 and present definitions, purpose, assumptions, inputs, outputs, and possible extensions for the proposed DTs.

6.3.1.1 Physical Process

We assume that the physical process (referred to as *process* for the rest of the chapter) is a manufacturing process that has sensors in place to collect in- and ex-situ data and the measurements are available to the DTs in the framework for data analysis purposes in run-time as well as in the form of historical data through a database. A discrete-time *temporal* representation of the process is then given in a general form as

$$\boldsymbol{x}(t+1) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t), t) \tag{6.1a}$$

$$\boldsymbol{y}(t) = g(\boldsymbol{x}(t), \boldsymbol{v}(t)), \tag{6.1b}$$

where $\boldsymbol{x}(t)$ denotes the process states, $\boldsymbol{u}(t)$ is the process input, $\boldsymbol{y}(t)$ is the measurement, $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are process and measurement noise respectively, t is the discrete-time index, and f and g are the process and measurement models, respectively. The time index t represents either the in-situ time index or the run-to-run (R2R) index based on the applications of interest. We utilize t to represent the in-situ time index for the rest of this chapter. We can augment the notation of (6.1) to include both time indices if needed (e.g. $\boldsymbol{x}_j(t)$ where j is the R2R index and t is the in-situ time index). Note that $\boldsymbol{y}(t)$ may represent an in- or ex-situ measurement in this context and it is referred to as the process *output* when the interpretation is clear from the context. Additionally, note that in contrast to the spatial dynamical models presented in Chapters III-IV, the model in (6.1) is a temporal model of the process.

An SME monitors the process through the DTs in the framework as illustrated in Fig. 6.2, and implements reconfigurations or changes to the process through a decision-maker. If the decision-maker is purely advisory, the SME may implement actions and prescribe references directly to the process (i.e., without a decision-maker). We assume that a decision-maker exists in the framework without loss of generality. The decision-maker

provides setpoint references $\boldsymbol{r}(t)$ for the process to track (in the sense that $||\boldsymbol{r}(t) - \boldsymbol{y}(t)||$ is as small as possible in a suitable norm).

While we assume that the process has the form in (6.1) for our further discussions and developments, systems of various forms and dynamics can be considered here (e.g., discrete event systems). Additionally, the process itself can be modeled as a separate DT to perform simulation-based analysis on the process.

6.3.1.2 Controller DT

The Controller DT houses the run-time controller with the control logic, as well as observers, process models, and simulation tools. The Controller DT employs various control methods (e.g., feedback, feedforward, rule-based, hybrid, etc.) to regulate the process measurements $\boldsymbol{y}(t)$ toward the reference setpoints $\boldsymbol{r}(t)$ provided by the decision-maker. To perform state-based control, the Controller DT may incorporate various types of filters and estimators to estimate the current and future states of the process by using the measurements and information such as historical data, or model adaptation information provided by other DTs in the framework (e.g., models of the noises $\boldsymbol{v}(t)$ and $\boldsymbol{w}(t)$). Control inputs $\boldsymbol{u}(t) \in \mathcal{U}$ are implemented on the process, where $\mathcal{U}$ denotes an input constraint set. In practical implementations, there may be additional safety control loops that bypass the control input implementation (e.g., emergency stop switch for a robotic manipulator).

- The inputs are the reference setpoints $\boldsymbol{r}(t)$ from the decision-maker, and run-time data (sensor measurements including but not limited to $\boldsymbol{y}(t)$) from the process.
- The outputs are the control input $\boldsymbol{u}(t)$ to be implemented on the process, states of the process (estimated via observers), process indicators, model states considered by the control logic, and the measurement signal $\boldsymbol{y}(t)$.

We do not limit the type of control that may be employed by the Controller DT, and instead focus on its purpose, control goals, (e.g., output regulation to a given setpoint or optimization of a given objective function), and output metrics (e.g., the control signal, observer states, measured signals, and performance indices). In addition to the existing process control solutions, the Controller DT may have process simulation, data analytics, and adaptation capabilities to consider external data sources from available databases and other DTs. The adaptation and integration capability of the Controller DT enables novel control applications for CPMS.

The Controller DT may utilize information from other DTs, e.g., to simulate system dynamics for what-if analyses of the physical process [159] or estimate remaining useful life to detect anomalies and optimize end-of-life control strategies [15].

6.3.1.3 Feature DT

The Feature DT provides uniform data streams to the DTs in the framework to improve the interoperability of the framework. Existing run-time anomaly detection methods often rely on residual analysis to provide threshold-based decisions. The residual signal is evaluated via the given reference $\boldsymbol{r}(t)$, system model (e.g., (6.1)), or a DT-based simulation of the process that runs alongside the physical twin to provide run-time analytics. We assume that an SME defines the desired residual signals with specific features, and implements them as part of the Feature DT so that the residual information is shared with other DTs for further data analysis.

Another important task of the Feature DT is to evaluate key process indicators (KPIs) for the process. KPIs are widely used in manufacturing processes to assess run-time performance. Various types of KPIs include health indicators, performance indicators, and efficiency indicators [159, 192]. Similarly, the Feature DT may be tasked to pre-process or partition large scale or high sampling-rate measurement data for another DT that performs statistical learning on the measurement data.

- The inputs are the data streams from the decision-maker, process, and the Controller DT. There inputs are aggregated, and pre-processed by the Feature DT.

- The outputs are the processed data streams with the SME designed data features, residuals, and an indication of output quality whenever applicable.

In many practical applications, the physical process and its Controller DT are on a different interface and platform than the data analytics platform. In such cases, the Feature DT is tasked with implementing the appropriate interfaces for data communication and storage to a local database. While the Feature DT is implemented in the framework based on the specific needs of other components and DTs, we leverage existing Feature DTs if available, and implement the additional capabilities that are needed by our cybersecurity solution on top of the existing solutions.

6.3.1.4 FD/AD DT

The FD/AD DT performs fault and anomaly detection on run-time data streams. Preliminary detection capabilities are included in most CPS for reliable run-time performance. Such detection mechanisms are considered as part of the FD/AD DT here. The detection systems are often built to monitor operation-critical system components and corresponding KPIs in a limit-checking fashion. The purpose of the FD/AD DT is to represent the

existing detection mechanism as part of our framework and implement additional detection mechanisms if necessary. The FD/AD DT is usually built to perform threshold-based limit-checking on the physical process. The FD/AD DT may include safety monitoring and performance monitoring systems to detect anomalies and faults. A review of various model-based anomaly detection methods for control systems is given in [99] and more specific anomaly types for smart manufacturing systems with possible detection methods are discussed in [125].

- The FD/AD DT takes inputs from the Feature DT to perform its analysis. Historical data provided by a database may also be utilized for analysis. Additionally, the attack detection predictions of the Cybersecurity DT may be utilized to refine threshold parameters in the FD/AD DT.

- The FD/AD DT provides its outputs for the detection of an anomaly with an indication of prediction quality (i.e., confidence in detection) to the decision-maker and the operator in the system. We further utilize the outputs of the FD/AD DT to implement our Cybersecurity DT. We consider an alarm as an indication of a fault or anomaly at the DT output. The FD/AD DT may also share the corresponding data traces for the predicted faults and anomalies.

While we assume a threshold-based limit checking method for the FD/AD DT here, additional methods that adapt and learn anomalous or fault behavior of the process over time may be implemented as extensions. The use of FD/AD DT outputs in the Cybersecurity DT provides additional insight on what types of extensions may improve and extend the framework performance.

6.3.1.5 The Cybersecurity DT

The DT-based cybersecurity approach proposed in this work is implemented by the Cybersecurity DT. Specifically, the Cybersecurity DT provides predictions about attacks on the system in the context of anomalies and transient response of the controlled process. We assume that the Cybersecurity DT is designed by an SME knowledgeable of the cybersecurity of the process, and we focus on attacks with output measurable effects as stated earlier. The Cybersecurity DT is a novel contribution of this work to distinguish cyber-attacks from expected anomalies for a controlled process, and we provide a detailed analysis of the Cybersecurity DT in later sections.

- The output data streams and FD/AD indications of the Feature DT and the FD/AD DT

are the inputs for the Cybersecurity DT. Additional historical data available through a database is also used as inputs for training data models.

- The predictions of attacks versus expected anomalies with an indication of the prediction quality and key features found in the analyzed signal (e.g. features indicating the type and/or source of an attack) are the outputs of the Cybersecurity DT. Additionally, the attack features found in the analyzed data are shared with the FD/AD DT and the SME Operator for further analysis.

6.3.1.6 SME Operator

The operator monitors the outputs of the FD/AD DT and the Cybersecurity DT to further analyze if the physical process has an anomaly or is under a cyber-attack. For this purpose, the DTs report their prediction quality and the features found in the data so that a human SME may further investigate any abnormalities.

6.3.1.7 Decision Maker

The role of the decision-maker is to provide an interface between the SME and the plant floor. Many CPMS in practice utilize a supervisory control and data acquisition (SCADA) layer as a decision-maker. The decision-maker may have a supervisory role where it takes actions on the plant floor by making autonomous decisions. If the decision-maker is purely advisory, the SME may implement actions and prescribe references directly to the controlled plant, bypassing the decision-maker. In our context, the decision-maker provides details and updates on the reference signal $\boldsymbol{r}(t)$ for the process.

The presented framework forms a basis for the analysis of cyber-attacks for CPMS in the context of closed-loop controllers and expected anomalies. However, we can consider various extensions to our framework based on the expected behavior of the CPMS under investigation. For example, depending on the type of application, the AD-DT and the Detector DT may be merged into a single unit, providing predictions about the system and communicating with additional DTs on the plant floor. It is important to note that the presented framework and the DTs may be extended to accommodate solutions from the literature (e.g., [101, 116, 154, 176, 181, 193, 206]). Furthermore, the presented framework may be aggregated into a system-level DT that operates within or outside of the four walls of operation (e.g., at the supply chain level) [11, 138].

6.4 The Cybersecurity DT

The architecture of the Cybersecurity DT is illustrated in Fig. 6.3. The Cybersecurity DT utilizes a Detector DT and a Consistency DT to analyze run-time and historical data, and perform online data analysis. In this section, we present the architecture of the Cybersecurity DT and details of the proposed attack detection methods to distinguish attacks in the context of anomalies for a controlled process.

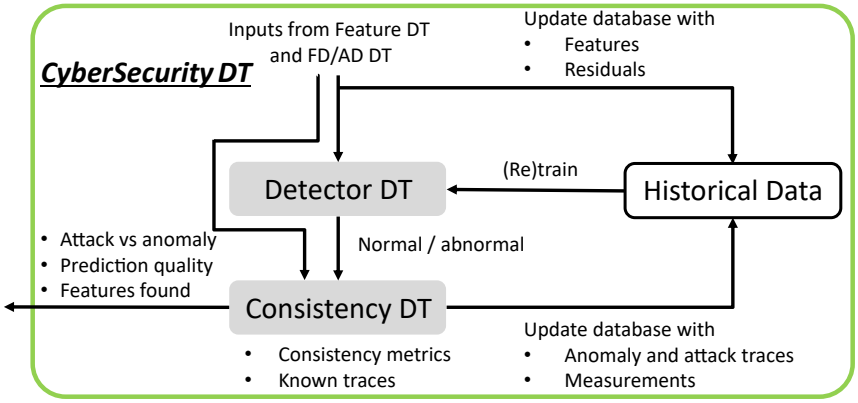


Figure 6.3: The architecture of the Cybersecurity DT. The Detector DT and the Consistency DT are used for detecting abnormalities and attacks on the physical process. The historical data is stored in a database for model training as well as knowledge storage and SME data mining of the types of expected anomalies, attacks, etc.

6.4.1 Cybersecurity DT Architecture

We propose a Cybersecurity DT that utilizes two DTs to perform abnormality detection and attack detection, so that it can predict the presence of a cyberattack on the physical process. The Cybersecurity DT also has a database that includes historical data used for model training, data mining, and data analysis. Note that while the Cybersecurity DT uses run-time data to predict attacks, the DT may or may not run synchronously with the physical twin. We assume that the data streams within the framework are time-stamped such that asynchronous DT predictions that indicate predicted time-instance of an attack onset are possible. The actual time frame of the DT versus the physical process for a practical implementation depends on the application domain of the process.

6.4.1.1 Detector DT

Noting that our goal is to detect attacks that have output measurable effects, we first need to identify if a measurement is abnormal. The Detector DT is tasked with performing abnormality detection on the process data by leveraging the anomaly prediction from the FD/AD DT. A key problem with anomaly and attack detection is the scarcity of abnormal process data versus the abundance of *normal* process data, leading to an unbalanced data set. For this purpose, machine learning models such as one-class discriminators [79, 139, 181, 207] and auto-encoders [118, 208, 210] are often utilized in the literature for abnormality detection to represent the normal data sufficiently well in a projected space such that abnormal data can be detected efficiently.

We utilize data-driven models in the detector DT and assume the availability of sufficient normal process data to train data-driven models. Additionally, we utilize the FD/AD DT predictions in our framework to improve the abnormality detection in the Detector DT. We present a Detector DT leveraging one-class discriminators in later sections and demonstrate our approach in the experimental study.

- Inputs to the Detector DT are the outputs from the Feature DT (e.g., features of the process measurements) and the FD/AD DT (e.g., prediction of an anomaly, anomaly traces found in the analyzed data). Historical data from a database is used to train data-driven models in the Detector DT.

- The Detector DT provides an indication of abnormality for the processed data. If the abnormality detection has an associated detection threshold, the threshold value is reported as well.

If the physical twin undergoes modifications that affect the dynamics of the process (e.g., physical wear, maintenance event, re-calibration of sensors, new data streams, software updates), the Detector DT may be re-trained (given that sufficient data in the historical database is available), adjusted to another model in its library (if the new context environment has already been modeled), or modeled to track certain slow-varying dynamics such as slow drifts as part of the new normal behavior. We focus on data-driven discriminator based abnormality analysis in our work. Extensions to the presented approach include physics-based analysis utilizing models such as (6.1), residual-based analysis (e.g., using a *golden trace*), and rule-based analysis.

6.4.1.2 Consistency DT

If a measurement is labeled as abnormal by the Detector DT, or alternatively if an anomaly is detected by the AD-DT, further analysis is performed by the Consistency DT to understand whether the abnormality is an attack or an expected anomaly. To understand if an abnormal measurement is due to an attack or an anomaly, we utilize the notion of *consistency metrics* on the process data. A consistency metric for the physical process (6.1) characterizes the expected behavior of the system during expected anomalies. We provide a formal definition of consistency metrics and how they are used for attack detection in later sections.

We assume that specifications defining the behavior of the system under *expected anomalies* are provided to the Consistency DT. Therefore, the Consistency DT monitors these specifications on the run-time process data to detect inconsistencies that predict the presence of an attack on the process. We focus on formal methods-based approaches to encode specifications for expected anomalies for the CPMS process. Within this context, the specifications are represented in terms of the progression of consistency metrics in a formal language that is used by the Consistency DT to monitor the process for attacks.

- Outputs from the Feature DT, Detector DT, and the FD/AD DT are utilized for evaluating consistency metrics and monitoring the metrics in run-time. Process specifications based on the consistency-metrics for expected anomalies are also provided to the Consistency DT.

- The Consistency DT outputs the prediction that an attack has occurred with the prediction quality and the features found in the data traces. The anomaly and attack traces found by the Consistency DT are also stored as part of the historical data for further analysis.

We assume an expert analyzes historical data of the process to evaluate consistency metrics and specifications for expected anomalies. Therefore, a new anomaly (one that is not considered in the set of expected anomalies) is predicted to be an attack by the Consistency DT. As the Consistency DT reports the features found on the data as well as data traces, an SME may design additional consistency metrics and specifications for a new anomaly by analyzing the process data. We present example consistency metrics and corresponding formal specifications for a CPMS process in the experimental study. Extensions of our approach could utilize data-driven methods to identify consistency metrics and corresponding formal specifications. Developing such methods is a subject for future work.

6.4.1.3 Historical Data

The historical database stores process data, the expected anomaly features, and data traces, as well as historical outputs from the Feature DT and the FD/AD DT. The database is updated with the outputs of the DTs and the SME updates the database with new expected anomalies encountered on the process.

6.4.2 Proposed Methods for Attack Detection

We present the theoretical background of the detection methods used in the Detector DT and the Consistency DT for abnormality detection and consistency metrics based attack detection, respectively, in this section.

6.4.2.1 Abnormality Detection

To implement abnormality detection, the Detector DT is trained on the historical process data $D = \{\boldsymbol{y}(t), \boldsymbol{x}(t), \boldsymbol{u}(t), \eta(t) \mid t = t_0, t_0 + 1, \ldots, t_0 + n_w\}$, where $\eta(t) \in \{0, 1\}$ is a label for abnormality of a data point, to recognize features of normal process data. Note that in practice for an unbalanced dataset, we utilize only a single class label (i.e., the normal data) and define everything else as abnormal (i.e., η becomes trivial as all data in D corresponds to a single class). We denote the *normal* data boundaries trained by the Detector DT using the data D as $\mathcal{B}(D) \subset \mathcal{F}$, where $\mathcal{F}$ is a possibly nonlinear feature space where the Detector DT operates.

The Detector DT utilizes its trained model $\mathcal{B}(D)$ to monitor run-time data provided by the Feature DT and FD/AD DT and detect if current measurements of the physical process are normal, i.e., if $\psi(\boldsymbol{y}(t')) \in \mathcal{B}(D)$, where $\psi : \mathcal{Y} \to \mathcal{F}$ is a map from the measurement space $\mathcal{Y}$ to the feature space $\mathcal{F}$ of the Detector DT. Based on this analysis the Detector DT outputs its prediction as a label $\hat{\eta}(t') \in \{0, 1\}$ of normal versus abnormal. Additionally, probabilistic predictions and prediction quality measures may be provided.

The training for the Detector DT utilizes historical process data of the steady-state operation at a predefined setpoint reference, e.g., $\boldsymbol{r}(t) = \bar{\boldsymbol{r}}$, to train $\mathcal{B}(D)$. However, the setpoint of the process may be altered either by a decision-maker or by a closed-loop controller on the physical process. The setpoint changes result in transient dynamic behavior on the system represented by (6.1), which may cause false positives by the Detector DT.

To mitigate false positives of the Detector DT during transients, we utilize the solution map of the system represented by (6.1), $\phi : \mathcal{X} \times \mathcal{U}^\infty \times \mathbb{Z}_+ \to \mathcal{X}$, where $\mathcal{X}$ is the state space of (6.1) and $\mathcal{U}^\infty$ is the space of sequential control inputs on (6.1). Given an initial state $\boldsymbol{x}(t_0)$ and a control sequence $\boldsymbol{u} \in \mathcal{U}^\infty$ over a time interval including the interval $[t_0, t_c]$, we

have

$$\phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_c) = \boldsymbol{x}(t_c), \tag{6.2}$$

where $\boldsymbol{x}(t_c)$ is the state at time t_c (i.e., the current state). Our motivation for the proposed abnormality detection method is to utilize the trained data boundaries $\mathcal{B}(D)$ during transient response. Roughly speaking, as $\mathcal{B}(D)$ is trained for the process at a given setpoint, we define a projection using ϕ to estimate state of the process at a previous setpoint given the transient observations (i.e., as the process moves away from the said setpoint) and the control inputs. If the process is normal, (i.e., no attacks or anomalies), the projected state should be within $\mathcal{B}(D)$.

Remark 6.6. *Forward projections of the set $\mathcal{B}(D)$ for the transient control inputs can also be used for abnormality detection. However defining such projections may in general be computationally expensive as $\mathcal{B}(D)$ may be control and state dependent, and new computations are needed at each control step. Therefore, we focus on the proposed projection type method for abnormality detection in this work.*

Formally, the goal of the Detector DT during transients is to estimate the initial state $\bar{\boldsymbol{x}}(t_0)$ of the process at time t_0 based on the observed sequence of states and control input $\boldsymbol{u}$ until the current time t_c. Let us denote the model of the state progression as

$$\Phi(\boldsymbol{x}(t_0)) = \begin{bmatrix} \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_0) \\ \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_0 + 1) \\ \vdots \\ \phi(\boldsymbol{x}(t_0), \boldsymbol{u}; t_c) \end{bmatrix}. \tag{6.3}$$

Additionally, let $\boldsymbol{x}$ denote the sequence of estimated states of the process between the times $[t_0, t_c]$. Then, the Detector DT solves the following minimization to estimate the initial state $\bar{\boldsymbol{x}}(t_0)$ by using the control input $\boldsymbol{u}$ and the state sequence $\boldsymbol{x}$.

$$\bar{\boldsymbol{x}}(t_0) = \min_{z} \{||\Phi(z) - \boldsymbol{x}||\}, \tag{6.4}$$

where z is an intermediate variable for the notation. For a normal process (i.e., process outputs with $\psi(\boldsymbol{y}(t')) \in \mathcal{B}(D)$), the solution of (6.4) is close (in the normed distance sense) to the actual initial state $\boldsymbol{x}(t_0)$. Therefore, the Detector DT evaluates the abnormality of the projected state $\bar{\boldsymbol{x}}(t_0)$ to evaluate the label $\hat{\eta}(t_c)$ for the current state $\boldsymbol{x}(t_c)$. Namely, if $\bar{\boldsymbol{x}}(t_0) \in \mathcal{B}(D)$, then the current state $\boldsymbol{x}(t_c)$ is predicted as *normal* by the Detector DT.

6.4.2.2 Consistency Metrics for Attack Identification

As mentioned earlier, the Consistency DT monitors the progression of a set of consistency metrics to understand if the abnormal data traces belong to a known anomaly. Since there may be many types of anomalies in the system, the design of the appropriate consistency metrics is often a challenging task. Following (6.1) we define the anomalous states and measurements as $\tilde{\boldsymbol{x}}(t)$ and $\tilde{\boldsymbol{y}}(t)$, respectively. Due to the nature of anomalies, true models of the anomalous process and measurements are often unknown, but we do have historical data of $\tilde{\boldsymbol{x}}(t)$ and $\tilde{\boldsymbol{y}}(t)$ for known anomalies. Let us define a combined run-time state as

$$\boldsymbol{\zeta}(t) = vec(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{u}(t), \boldsymbol{r}(t)). \tag{6.5}$$

Our goal is to develop a consistency metric of type

$$\xi_i(t) = c_i(\boldsymbol{\zeta}; \boldsymbol{\theta}), \tag{6.6}$$

where $\boldsymbol{\theta} = \{(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{y}}_i, \boldsymbol{r}_i, \boldsymbol{u}_i) \mid i = 1, \ldots, n_h\}$, is the data set of length n_h from previous known anomalous process measurements, and $\boldsymbol{\zeta} = \{\boldsymbol{\zeta}(t), \boldsymbol{\zeta}(t-1), \ldots, \boldsymbol{\zeta}(t-n_w)\}$, for a window of size n_w. The Consistency DT monitors the progression of $\xi_i(t)$ with run-time data $\boldsymbol{\zeta}(t)$. Suppose we design our consistency metric such that $||\xi(t)|| \leq \delta(t)$, for some $\delta(t) \in \mathbb{R}$ during expected anomalies. Then, any abnormal measurement that results in $||\xi(t)|| > \delta(t)$ is logged as inconsistent. In our setting, an inconsistent measurement is a possible attack on the system. The design engineer for the consistency metric may utilize models of any kind (e.g., data-driven, physics-based, statistical, rules-based, etc.) to define a function c_i to evaluate consistency metrics. There may be multiple c_i in the system and the consistency DT may utilize an ensemble approach to detect inconsistencies.

A key challenge is defining $\delta(t)$ dynamically for a temporally measured signal. While specific $\delta(t)$ may be developed for individual use cases, the scalability of the design process becomes a prohibiting factor for using the proposed DT-based approach. An effective way to monitor consistency metrics may utilize signal temporal logic (STL) to develop logical predicates that prescribe the expected behavior of the measured signal over predefined measurement-time windows. STL is a widely used formalism to specify properties of a signal that is measured from a process. STL predicates for anomaly detection on an additive manufacturing (AM) process in a similar DT setting are presented in previous work [15]. We omit a detailed background on STL and refer interested readers to [66]. An

STL formula π is formed by the following syntax:

$$\pi \triangleq \quad \top \quad | \quad p \quad | \quad \neg\pi \quad | \quad \pi_i \wedge \pi_j \quad | \quad \pi_i \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi_j \tag{6.7}$$

where, $\top$ is logical true, p is a predicate, $\neg\pi$ is the logical negation of the proposition π, $\pi_i \wedge \pi_j$ is the logical conjunction of two propositions, and $\pi_i \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi_j$ is the *until* operator defined as the proposition π_i being true at least until the proposition π_j is true in the time interval $[t+a, t+b]$, where t is the current time. A signal $\boldsymbol{s}(t)$ at time t is satisfied by a predicate p if $\ell(\boldsymbol{s}(t)) > 0$ for some function ℓ (i.e. $\boldsymbol{s}(t) \models p \iff \ell(\boldsymbol{s}(t)) > 0$). Here the operator $\models$ is used to indicate that the condition on the left side satisfies the condition on the right side. Additionally, $\bot = \neg\top$ is the logical false, the *eventually* operator is $\Diamond_{[a,b]}\pi \triangleq \top \, \boldsymbol{\mathcal{U}}_{[a,b]} \pi$, and the *always* operator is $\Box_{[a,b]}\pi \triangleq \neg(\Diamond_{[a,b]} \neg\pi)$.

We utilize signal temporal logic (STL) to encode the consistent temporal response of the system for expected anomalies. Let $\Pi = \{\pi_1, \ldots, \pi_{n_s}\}$ denote the set of consistency specifications to monitor. We want the process to satisfy all the specifications $\pi_i \in \Pi$, thus the Consistency DT monitors if the conjunction of all specifications is satisfied (i.e., evaluates to true (top)). Thus, the consistency DT monitors the proposition

$$\boldsymbol{\zeta}(t) \models \bigwedge_{\forall \pi_j \in \Pi} \pi_j, \tag{6.8}$$

where we require the consistent run-time state measurement $\boldsymbol{\zeta}(t)$ (or a subset of the signals) to satisfy the conjunction of n_s propositions. While the proposed framework utilizes STL for consistency monitoring, extensions of the proposed framework may utilize various techniques including static and adaptive limit checking. We provide examples of inconsistency metrics and how they can be used for attack detection in the experimental demonstration.

As mentioned previously, unexpected anomalies, i.e., anomalies that are not considered in the set of anomalies during the design of the Consistency DT, are also predicted as inconsistent. When such new anomalies are encountered, the design engineer utilizes the new data traces $\boldsymbol{\theta}'$ containing data from the new anomalies to potentially design new consistency metrics and propositions to be monitored by the Consistency DT.

6.4.3 Integration of the DTs for Attack Detection

Figure 6.4 illustrates the information flow between the DTs for attack detection in the presence of expected anomalies in the system. For the purposes of illustration, we denote a flowchart of how the prediction of the DTs inform each other and note that Fig. 6.4 does not illustrate the data shared between the DTs. The Feature DT continuously provides

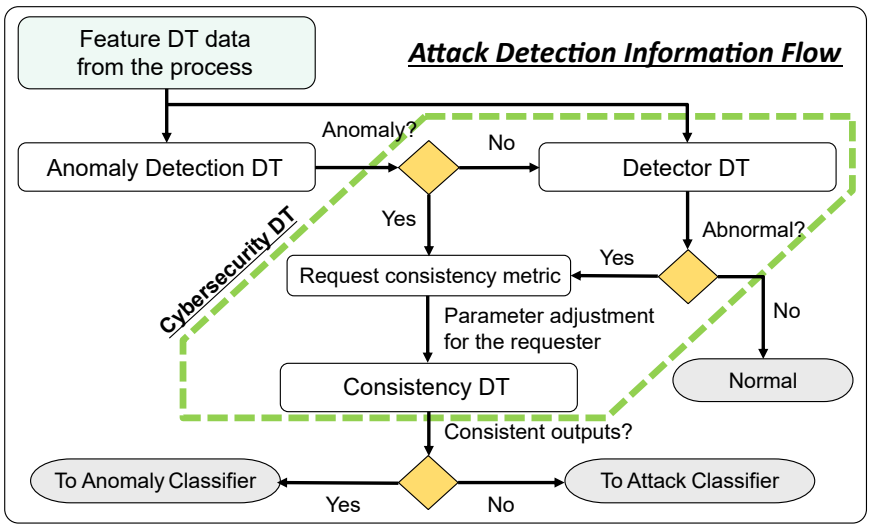


Figure 6.4: Illustration of the information flow in the framework for attack detection with the Cybersecurity DT. The boundaries of the Cybersecurity DT are outlined with dashed lines.

data to all other DTs in the framework. If the Feature DT includes an event trigger in its outputs, other DTs may use the trigger to perform analysis, or they may continuously perform analysis on the streaming process data.

First, the AD-DT performs threshold-based limit-checking to predict if there are any anomalies in the process. As we treat the AD-DT as part of existing detection mechanisms on the CPMS, we expect that its threshold limits are tuned at a desired operating characteristic by an expert. For our case study, we assume that the AD-DT has "wide" threshold limits set by an expert to reduce the false-positive rate (i.e., cost of a false-positive is more expensive than a missed-positive). Consequently, if the AD-DT detects an anomaly, we utilize its detection to conclude that the data is abnormal and move to request necessary consistency metrics from the Consistency DT.

If the AD-DT does not detect an anomaly, the Detector DT is utilized to predict abnormality. If there is no abnormality, the data is labeled as normal and no further action is taken. If the Detector DT predicts an abnormality, a consistency metric is requested from the Consistency DT. The consistency DT may utilize multiple consistency metrics with various parameter settings based on which DT requests a given metric. After the Consistency DT performs its analysis on the data, a consistent output is predicted to be an anomaly and this prediction is shared with an anomaly classifier or decision-maker for further analysis. If the data is inconsistent, an attack is predicted and the prediction is shared with an attack classifier or decision-maker for further analysis.

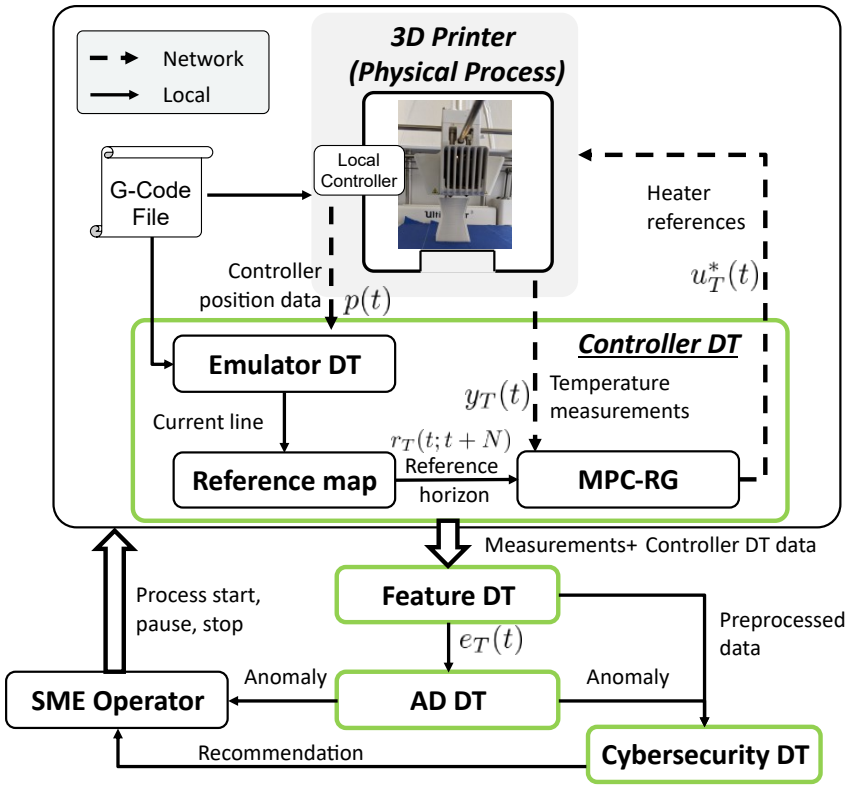


Figure 6.5: Illustration of the DT architecture with the off-the-shelf 3D printer. Data communication over the network is shown with dashed lines and local data communication is shown with solid lines.

6.5 Experimental Demonstration on an Off-the-Shelf 3D Printer

In this section, we provide an experimental demonstration of our proposed DT solution on an off-the-shelf 3D printer as an illustrative CPMS resource. Experimental data collected from a printer under normal operation, anomalous operation, and attacks are collected and analyzed by the Cybersecurity DT to present results of attack detection. The overview of the experimental setup is shown in Fig. 6.5.

6.5.1 Controller DT over a Network

For our experimental demonstration, we focus on the heating system of an off-the-shelf fused deposition modeling (FDM) 3D printer. In FDM, a thermoplastic material is extruded onto a build bed via a numerically controlled extruder with a heated nozzle. A G-Code file is an input to the printer's local controller, and the local controller executes each line of G-Code in sequence to deposit material at each layer to create a 3D geometry in a bottom-up, layer-by-layer fashion. Thus, a physical process is operated by purely cyber inputs.

6.5.1.1 Motivation

Heating the deposited material within the desired temperature range is crucial for an extrusion process. The local controller includes a PID loop that ensures robust tracking of a temperature reference $r_T(t)$ prescribed by the G-Code file; however, dynamic updates to the printing temperature are of interest for several reasons. Dynamic adjustment of printing temperatures is shown to greatly improve dimensional performance [69] as well as layer-to-layer material adhesion and part strength [172, 191]. To enable such applications of interest, we implement a network Controller DT that adjusts the printing temperature of the 3D printer based on a *reference map* that is designed by an engineer for a specific printing process. Run-time communication of the heater inputs over a network induces potential cyber-attack vulnerabilities that may cause the failure of the printed part or the machine itself. For this purpose, we demonstrate how our framework enables DT-based cybersecurity solutions for the controlled physical process in the context of expected anomalies.

6.5.1.2 Controller Implementation

Since we do not have direct access to the nozzle heaters in the printer, we model the closed-loop heating system (i.e., heaters controlled by the local controller) and develop a model predictive controller (MPC) scheme to prescribe heater references so that the system output $y_T(t)$ tracks a reference temperature $r_T(t)$. To implement a controller, the heating system is modeled as a discrete-time second order linear time invariant (LTI) system

$$\boldsymbol{x}(t+1) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u_T(t) \tag{6.9a}$$

$$y_T(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{6.9b}$$

where the system matrices $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$, $\boldsymbol{B} \in \mathbb{R}^{2\times 1}$ are identified from the step response of the closed-loop heating system with $\boldsymbol{C} = [1 \quad 0]$. We define the control input limits as $\mathcal{U} = [160, 220]$ °C. Then, the goal of the MPC controller is to solve the following optimization problem in a receding horizon fashion

$$\min_{\boldsymbol{u}} \sum_{\tau=t}^{t+N-1} ||\hat{\boldsymbol{x}}(\tau) - \boldsymbol{x}^r(\tau)||_Q^2 + ||u_T(\tau) - u_T^r(\tau)||_R^2 \tag{6.10a}$$

$$+ ||\hat{\boldsymbol{x}}(t+N) - \boldsymbol{x}^r(t+N)||_P^2 \tag{6.10b}$$

$$\text{s.t.: } \hat{\boldsymbol{x}}(t+1) = \boldsymbol{A}\hat{\boldsymbol{x}}(t) + \boldsymbol{B}u_T(t) \tag{6.10c}$$

$$u_T(\tau) \in \mathcal{U}, \tag{6.10d}$$

where we have Q, R, P as positive definite controller gains, $||x||_Q^2 = x^T Q x$, $\boldsymbol{x}^r$ and u_T^r as the state and control input references, respectively, $\hat{\boldsymbol{x}}$ as the state estimate, N as the controller horizon, and $\boldsymbol{u} = \{u_T(t), u_T(t+1), \ldots, u_T(t+N-1)\}$. We use the solution of the corresponding Discrete-time Algebraic Ricatti Equation (DARE), i.e., the LQR gain, for defining the weight matrix P. We denote the optimal solution of (6.10) with $\boldsymbol{u}^*$. After the controller implements $u_T^*(t)$ on the physical system over the network, the optimization (6.10) is solved over an updated horizon with updated process data. We utilize a standard Kalman filter observer update to estimate the current state $\hat{\boldsymbol{x}}(t)$ and omit the observer formulation here for brevity.

6.5.1.3 Reference handling

As the formulation (6.10) suggests, the controller operates in the temporal domain. However, G-Code references executed on the printer are inherently spatial and event-based. To remedy this mismatch, we utilize an Emulator DT that emulates the printing process by analyzing the G-Code file. During run-time, the Emulator DT queries the position data of the four axes (e.g., x,y,z location of the extruder head, and the position of the extrusion (E) axis), $p(t) \in \mathbb{R}^4$, from the local controller. Then, the Emulator DT utilizes $p(t)$ to estimate the current line of G-Code executed by the printer.

For our case study, we utilize a temperature reference that alternates between $205°$C and $210°$C every five layers in the printing process.

6.5.2 Attack and Anomaly Scenarios

We consider two attack scenarios and two anomaly scenarios in our case study.

6.5.2.1 Anomalies

Two types of anomalies are considered:

A1 The first anomaly is caused by the use of a cooling fan on the extruder head. The fan increases the airflow over the extruder nozzle, which reduces its temperature. The cooling effect is an exogenous disturbance that is unknown to the controller and causes an anomaly in the temperature measurements.

A2 The second anomaly is the degradation of the heating system performance. As the heating system is used over time, its components undergo thermal and mechanical wear that cause the system response to be slower (in terms of settling time) than expected for a given temperature reference $r_T(t)$. As this effect occurs gradually over a long time

horizon (a matter of months of use), we instead simulate the degradation by updating the local controller gains to deliberately slow down the closed-loop response of the local heating system.

As shown in Fig. 6.5, an anomaly detection (AD) DT is implemented as a threshold-based limit-checking procedure on the temperature error $e_T(t) = r_T(t) - y_T(t)$. Thus, the AD-DT checks if the error is larger than a predefined threshold level $\beta_{AD} \in \mathbb{R}_+$, i.e., $|e_T(t)| > \beta_{AD}$. The value of β_{AD} is preset by a designer based on the expected system response characteristics (e.g., expected maximum temperature error, robustness margins, etc.). Note that, the AD-DT does not consider the compensation by the controller to minimize temperature error, but rather just monitors the output. An increased activity in the control signal when compared to the baseline normal conditions may indicate further abnormalities that may not be observable through output monitoring. Investigating detection methods utilizing the control signal itself via the Controller DT is a promising future research direction.

6.5.2.2 Attacks

As the DT framework communicates with the printer over a network, the measurement signals may be prone to attacks. To simulate network attacks on the measurements, we consider two attack types as $y_T(t) + w(t)$, where $w(t)$ is the attack signal we implement on the measurement.

T1 Injection of a constant offset to the measurement signal, e.g. $w(t) = c_1$ for some $c_1 \in \mathbb{R}$.

T2 Injection of a temporally cyclic signal to the measurement signal, e.g. $w(t) = c_2 \sin(t)$ for some $c_2 \in \mathbb{R}$.

In this case study, we are focused on the attacks that compromise the temperature measurements $y_T(t)$. Attacks on the transmission of heater reference input ($u_T(t)^*$) from the Controller DT to the 3D printer are not considered.

6.5.3 Cybersecurity DT

Following the architecture illustrated in Fig. 6.3, the Cybersecurity DT is designed for abnormality detection and consistency checking.

6.5.3.1 Detector DT

As previously stated, the temperature reference alternates between the two setpoints every five layers. We utilize two one-class support vector machines (OSVM) [48, 79, 203] to model the normal behavior of the process at the two setpoints, one for each setpoint. An OSVM utilizes training data that correspond to the same class, also named as *positive training samples* e.g., measurements under normal operation. Let us denote the training data as $D^+ = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_{n_z}\}$, where $\boldsymbol{z}_i \in \mathcal{Z}$ denote individual measurements. Utilizing a mapping $\phi : \mathcal{Z} \to \mathcal{F}$, the OSVM trains its data boundaries with the following optimization

$$\boldsymbol{\alpha}^* = \min_{\boldsymbol{\alpha}} \{\boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha} \mid 0 \leq \alpha_i \leq \frac{1}{v n_z}, \textstyle\sum_i \alpha_i = 1\}, \tag{6.11}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{n_z}]$ is the decision variable, $v \in \mathbb{R}_+$ is a user-defined regularizer parameter, and $\boldsymbol{Q}[i,j] = k(\boldsymbol{z}_i, \boldsymbol{z}_k) = \phi(\boldsymbol{z}_i) \cdot \phi(\boldsymbol{z}_j)$ with $k(\boldsymbol{z}_i, \boldsymbol{z}_k)$ representing a kernel function, which is in turn given by the dot product $\phi(\boldsymbol{z}_i) \cdot \phi(\boldsymbol{z}_j)$. The optimal threshold value is evaluated as

$$\rho^* = \textstyle\sum_i \alpha_i k(\boldsymbol{z}_j, \boldsymbol{z}_i). \tag{6.12}$$

The decision function for one-class classification is given as

$$h(\boldsymbol{z}_*) = sgn(\textstyle\sum_i \alpha_i k(\boldsymbol{z}_i, \boldsymbol{z}_*) - \rho^*). \tag{6.13}$$

Furthermore, we denote the trained model of the OSVM as

$$\mathcal{B}(D^+) = \{\boldsymbol{z} \mid h(\boldsymbol{z}) \geq 0\}. \tag{6.14}$$

If we have a sample $\boldsymbol{z} \in \mathcal{B}(D^+)$, the Detector DT predicts that the sample is normal, and abnormal otherwise. Note that if we are training only on the measurement outputs $\boldsymbol{y}(t)$, we may utilize $\phi = \psi$, where the map ψ is given previously in Section 6.4.2.1. For further details on the derivation of (6.11), see [48, 207]. For our implementation, we utilize the output measurements to train our OSVM, i.e., $\boldsymbol{z}_t = y_T(t)$.

To collect training data D^+, we run the process at the given setpoint temperatures (denoted with T_1^s and T_2^s) and with the MPC providing heating references in closed-loop. We train two OSVMs on the collected data at two different setpoints, denoted with $D^+(T_1^s)$ and $D^+(T_2^s)$ to evaluate the two models as $\mathcal{B}_1 := \mathcal{B}(D^+(T_1^s))$ and $\mathcal{B}_2 := \mathcal{B}(D^+(T_2^s))$.

To deal with controlled transient behavior, we utilize the solution map ϕ of the linear

system model (6.9). Since the Cybersecurity DT is provided with data from the Controller DT, the discrete-time index when the system is driven to a new setpoint is tracked as $t = n_{sp}$. Then, by querying the sequence of previous inputs $\boldsymbol{u} = \{u_T^*(t-1), u_T^*(t), \ldots, u_T^*(t - n_{sp})\}$ from the Feature DT, the Detector DT evaluates the projected state $\bar{\boldsymbol{x}}(t - n_{sp})$ by utilizing the state estimates $\hat{\boldsymbol{x}}$ (see (6.4)). Since the OSVM is trained on the output measurements, we further get the corresponding projected output measurement as $\bar{y}_T(t - n_{sp}) = \boldsymbol{C}\bar{\boldsymbol{x}}(t - n_{sp})$. If we have $\bar{y}_T(t - n_{sp}) \in \mathcal{B}_i$, where i denotes the corresponding previous setpoint, then the Detector DT predicts that the current state estimate $\hat{\boldsymbol{x}}(t)$ is normal and abnormal otherwise.

Remark 6.7. *The abnormality detection checks the condition $\bar{y}_T(t - n_{sp}) \in \mathcal{B}_i$. If the volume of $\mathcal{B}_i$ is too large (in a multidimensional sense), projections of certain attacked process measurements may still be within $\mathcal{B}_i$, resulting in false negatives. Additional models to refine the abnormality predictions of the Detector DT may be utilized to reduce false negatives in such cases.*

6.5.3.2 Consistency DT

We present a consistency DT designed by utilizing expertise knowledge about the controlled physical process. By including the Controller DT in our framework, we have additional information about the expected system behavior under closed-loop control. Namely, the controller (6.10) provides near offset-free tracking under perfect model and state knowledge (see [151] for further details). Since we have inherent uncertainties in our model (6.9) as well as state estimation, we expect the controller to have a small steady-state tracking offset (in the normed sense), which we evaluate experimentally (an over-approximation of this offset is denoted with δ_1). Then we define a consistency metric

$$\xi_1(t) = c_1(y_T(t), r_T(t)) = |y_T(t) - r_T(t)|, \tag{6.15}$$

which provides us with the norm of the output measurement residual signal. For a consistent physical system, the residual should converge to a neighborhood of the empirically determined tracking offset, e.g. $\xi_1(t) \leq \delta_1$ as $t \to \infty$. However, monitoring the asymptotic response of the system is often not feasible or desirable. Let $\tau(t) \in \{0, 1\}$ denote a consistency metric request, such that we have $\tau(t) = 1$ if either the AD-DT or the Detector DT requests a consistency metric and $\tau(t) = 0$ otherwise. We define $\xi_2(t, t_0, t_f)$ as another consistency metric to count the number of requests in a given time interval $t \in [t_0, t_f]$. We use ξ_2 to define specifications that are robust to transient response in the state estimation, which often occur due to disturbances on the process and data communication. Using STL,

the monitoring logic is given as

$$\pi_1 : \tau(t) \implies (\tau(t+1) \vee \Diamond_{[t+1,t+t_s]}(\xi_1(t) \leq \delta_1)), \tag{6.16a}$$

$$\pi_2 : \tau(t) \implies \Diamond_{[0,\beta]}(\neg\tau(t)), \tag{6.16b}$$

$$\pi_3 : \tau(t-1) \wedge \neg\tau(t) \implies \Box_{[0,\beta_2]}(\xi_2(t,0,\beta_2) \leq 1), \tag{6.16c}$$

$$\pi_4 : \tau(t) \wedge (\xi_1(t) \geq 1) \implies \Diamond_{[t+1,t+t'_s]}(\xi_1(t) \leq \delta_1), \tag{6.16d}$$

where (6.16a) denotes that whenever the consistency metric is requested, (i.e., $\tau(t) = 1 = \top$); either the consistency metric is requested again in the next time-step ($t+1$), or the temperature error norm eventually converges to the tracking offset δ_1 within t_s time-steps. Similarly, (6.16b) requires that whenever the consistency metric is requested, i.e., the measurement is abnormal, the new measurements should be normal eventually in the next β time steps, where the value of β is determined empirically based on previous process data. Next, (6.16c) requires that whenever the output measurement turns normal (i.e., request is made in the previous time step but not the current one), it should stay that way for the next β_2 time-steps based on the ξ_2 consistency metric (i.e., there can be at most one consistency metric request in the given time interval). Similar metrics may be developed for various applications of interest. Lastly, (6.16d) denotes that if there is a large deviation in the signal at the time of consistency metric request, the signal should converge to the δ_1 limit within t'_s timesteps. Note that π_3 and π_4 may be merged for a succinct representation, but we utilize two separate specifications here for clarity. We denote the STL specification

$$\pi = \pi_1 \wedge \pi_2 \wedge \pi_3 \wedge \pi_4, \tag{6.17}$$

which is satisfied (`SAT`) if and only if all the propositions in (6.16) are satisfied (i.e., they are $\top$). The Consistency DT monitors the process data to check if the specification (6.17) is `SAT`. The monitoring process may be performed in run-time with robust satisfaction monitoring techniques [60]. Here we instead utilize a retrospective analysis on the collected process data stored in the Cybersecurity DT to perform the monitoring task. We expect a consistent process to have the output measurement stable under the closed-loop control implemented by the Controller DT, resulting in `SAT`. When the measurement signal is compromised, the controller will not be able to track the desired reference, which causes large tracking errors or instability on the temperature signal that result in (6.17) to be unsatisfied (`UNSAT`).

We determine the value of the parameter t'_s based on the expected system performance as a constant parameter during the process. However, the parameter t_s should be adjusted

based on the initial condition of the mismatch between $y_T(t)$ and $r_T(t)$ dynamically, since the system dynamics have a non-trivial time constant. To evaluate the parameter t_s in (6.16a) we utilize a user-defined parameter mapping for t_s, which is determined based on the analysis of previous anomalous data by an expert.

6.5.4 Implementation Details

We use an Ultimaker 3 FDM printer with Polylactic Acid (PLA) material for our case study. The printer has a network API that allows for monitoring of extruder temperatures $y_T(t)$, stepper motor counts on the controller $p(t)$, and extruder heater input updates $u_T(t)$. The Controller DT is implemented on a personal computer using python. For reliable network communication between the Controller DT and the printer, a sampling time of 0.5 seconds is used in the case study. In each 0.5 second cycle, the Controller DT execution sequence is repeated and a new heater reference $u_T^*(t)$ is sent to the printer over the network. The linear model in (6.9) is adjusted for actuator delays, which were approximated to be 10 time steps for the implementation of the control law using (6.10).

We collect experimental data for (i) the nominal system without the Controller DT running, (ii) the controlled system with the Controller DT without attacks or anomalies, (iii) the controlled system with anomaly cases (***A1*** and ***A2***), and (iv) the controlled system with sensor attack cases (***T1***, ***T2***).

6.5.5 Results

In this section, we present experimental study results from the implementation of the proposed framework. Figures in this section are the results of multiple runs of the printing process for various anomalies and attacks. Therefore the time axes on the processes do not illustrate times from a single printing process but rather show the time scale of the dynamic behavior.

6.5.5.1 Detector DT performance

We use nominal process data when the Controller DT is running to train the OSVMs for the two temperature setpoints $T_1^s = 205°\text{C}$ and $T_2^s = 210°\text{C}$. We use radial basis function kernels for our initial OSVM training. The transient analysis in (6.4) is sensitive to model mismatches, which results in false-positives on normal data traces. Thus, we analyze the performance of the OSVMs on the nominal process measurement and transient responses to improve their robustness by perturbing their data boundaries. We perturb the boundaries based on the performance of the Detector DT on the historical data to mitigate

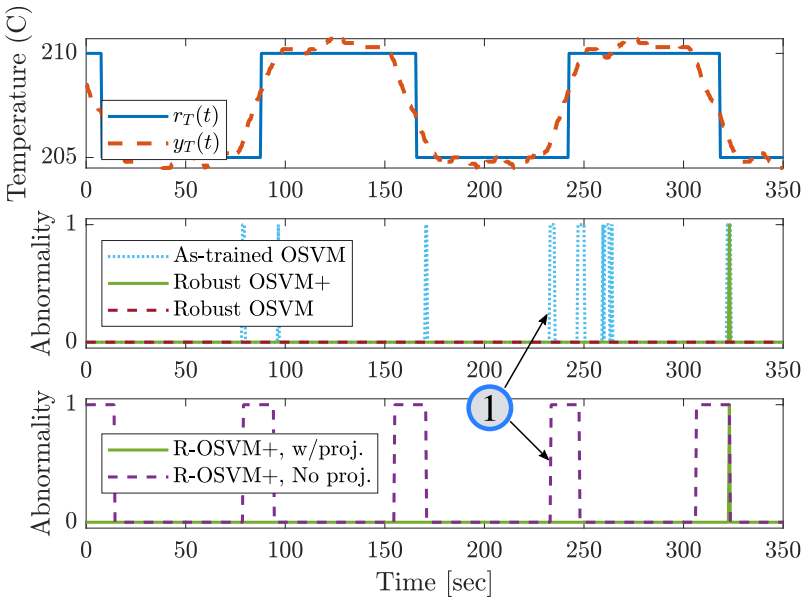


Figure 6.6: Illustration of the Detector DT abnormality detection during normal operation using the robust bounds. The marker 1 indicates the data signals with the false positives discussed in the text.

false positives (e.g., include historical data points that are known to be normal but predicted abnormal by the Detector DT in the training set D). This procedure reduces the false positive rates of the Detector DT at the expense of reduced sensitivity (see Remark 6.7). The perturbation procedure is designed such that the resulting boundaries of the *Robust OSVM* are an over-approximation of the original *as-trained OSVM*.

To remedy the reduced sensitivity of the Detector DT with the Robust OSVM due to the perturbation procedure, we utilize additional OSVMs to provide a solution to the scenario outlined in Remark 6.7. We train two new additional OSVMs (one for each setpoint) on the features of the Robust OSVM solutions for normal data points. The output of the Detector DT with these new additional OSVMs is denoted as *Robust OSVM+* in Fig. 6.6 and the indication $+$ is used in subsequent figures for comparisons to the two different Robust OSVM solutions. To train the Robust OSVM+, we extract three features from the Robust OSVM solutions on the normal dataset. Specifically, we train the Robust OSVM+ on the

- Root-mean-squared error of the estimation (6.4)
- Time elapsed since the last setpoint change
- The absolute estimation error $|\bar{y}_T - y_T|$,

of the Robust OSVM solutions on the normal dataset. We pass the datapoints that are

predicted to be normal by the Robust OSVM through the Robust OSVM+ for additional analysis and to improve the Detector DT performance.

Figure 6.6 illustrates the performance of the Detector DT with the updated robust data boundaries on normal process data. The top plot illustrates the temperature reading and the reference temperature for the process. The middle plot shows the comparison of the predictions using the As-trained OSVM, Robust OSVM, and Robust OSVM+. We observe that the As-trained OSVM results in a high rate of false positives in the middle plot, annotated with 1 on the figure. The Robust OSVM+ solution in the middle plot has slightly increased sensitivity that still rejects most of the As-trained OSVM false positives. We show how the increased sensitivity, when compared to the Robust OSVM, provides better attack detection performance in the following results.

On the bottom plot of Fig. 6.6 we illustrate the importance of the projection method given in (6.4) utilizing the Robust OSVM+ (R-OSVM+) procedure. The output without the projection (R-OSVM+, No proj.) has excessive false positives, annotated with 1 on the figure, since it predicts all the transient responses as abnormal datapoints. It is important to note that our proposed method for abnormality detection during controlled transient response works with minimal false positives in our experimental setup. The data trace (R-OSVM+, w/proj.) correctly identifies the transient behavior of the process as normal behavior and has minimal false positives (e.g. the one spike at time ≈ 225 seconds on the bottom plot of Fig. 6.6).

6.5.5.2 Cybersecurity DT performance

Here we analyze the performance of the Cybersecurity DT for the attack and anomaly cases considered in our case study. All the figures have rectangles around certain axis labels indicating the ground truth for the signal illustrated in the figure. Additionally, we provide numbered annotations on the figures to highlight some discussion points that we provide in the text.

Figure 6.7 illustrates the Detector DT and Cybersecurity DT output for the anomaly case of ***A2***. On the top plot, we see that the system response is altered due to the change in the local controller parameters. Specifically, the system response is slower than what is anticipated by the model in the controller. Consequently, the temperature signal significantly over- and undershoots the reference signal as the controller has a large model mismatch, which is illustrated by the process response on the top plot of Fig. 6.7. The Detector DT predicts abnormalities throughout the process and requests consistency metrics from the Consistency DT. In the middle plot we see the effect of the Robust OSVM+ versus the Robust OSVM, where the predicted abnormalities are slightly different for the two cases. Due

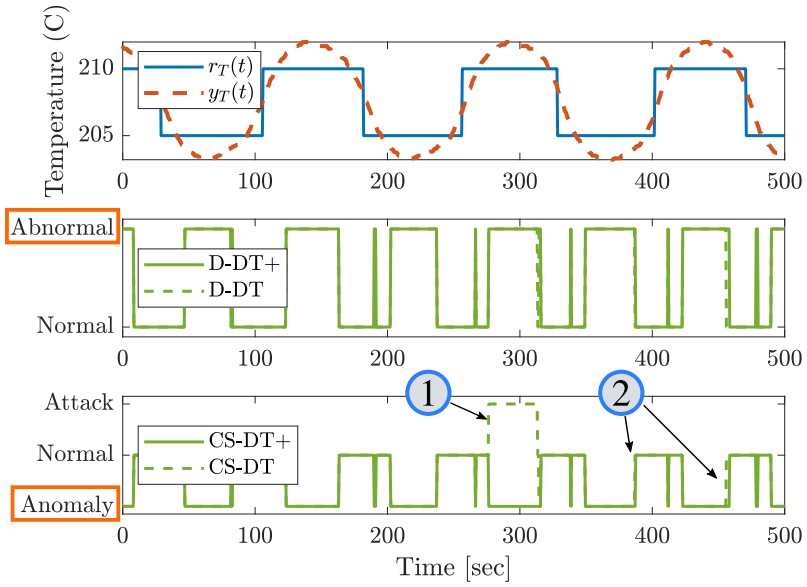


Figure 6.7: Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the anomaly ***A2***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Anomaly indicate the ground truth for the signals (i.e., if the signal is actually, normal/abnormal, anomalous/attacked, etc.). Annotations 1 and 2 are discussed in the text.

to the anomaly, the datapoints throughout Fig. 6.7 are abnormal and anomalous, which is indicated by the rectangles in the axis labels. The detector DT identifies the abnormalities as shown in the middle plot, during the setpoints and partially during the transient response between the setpoints. The transient response of the system is similar to the normal case, which results in the Detector DT predicting normal outputs during parts of the transient response.

Since the process is not attacked, we see that the temperature output is still bounded around the reference trajectory and thus the predictions of the Cybersecurity DT are *normal* where the temperature signal begins to converge towards the reference. On the bottom plot of Fig. 6.7, we see that the Cybersecurity DT correctly identifies the anomaly in the signal for the abnormal predictions identified by the Detector DT. Additionally, we see that the Robust OSVM, due to its reduced sensitivity in abnormality detection, results in a false postive of the Cybersecurity DT, which can be seen on the bottom plot of the Fig. 6.7, annotated with 1. In comparison, the Robust OSVM+ has no false positives (e.g. all the

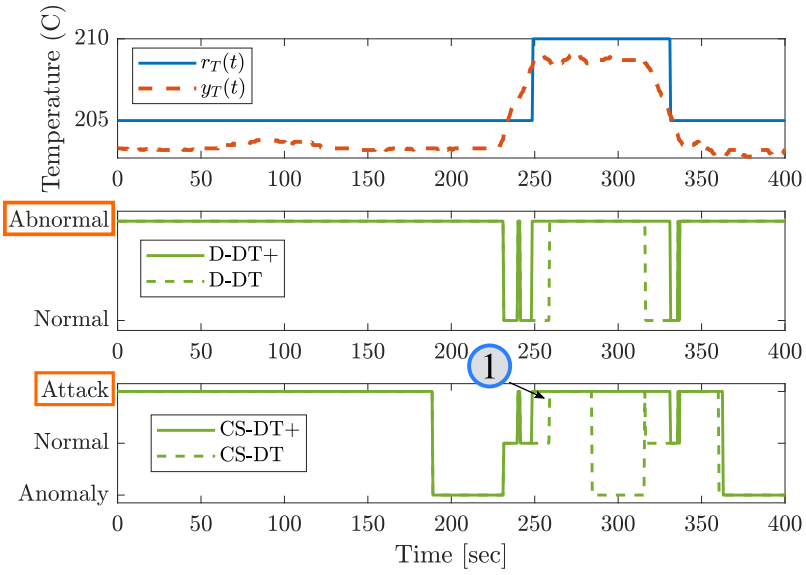


Figure 6.8: Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the attack ***T1***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Attack indicate the ground truth for the signals. Annotation 1 is discussed in the text.

predictions are either anomalous or normal, with no predictions of an attack). Similar to the case illustrated in Fig. 6.7, the Cybersecurity DT was able to correctly detect the anomaly ***A1*** for all the experimental data we have collected.

Figure 6.8 illustrates the Detector DT and Cybersecurity DT output for the attack case ***T1***. Due to the attack on the system, the controller stabilizes the temperature outputs at an offset away from the desired setpoints. The middle plot shows the abnormality predictions of the two OSVM procedures. The Robust OSVM+ has better abnormality prediction (shown with D-DT+) when compared to the Robust OSVM (shown with D-DT). Using the consistency measures and the STL monitoring, the Consistency DT analyzes the measurements and identifies the offset in the signal. On the bottom plot of Fig. 6.8 we see that the Cybersecurity DT identifies attacks on the inconsistent measurements from the process. During transients in the process, the signal behaves consistently with the nominal transient behavior, which causes the Cybersecurity DT to predict normal measurements. The annotation 1 on the bottom plot highlights the attack prediction of the CS-DT, which is later and more inconsistent when compared to the CS-DT+ outputs that use the Robust-OSVM+. We clearly see that the Robust OSVM+ in the Detector DT improves the abnormality detection

and consequently attack detection performance of the Cybersecurity DT.

Figure 6.9 illustrates the Detector DT and Cybersecurity DT output for the attack case ***T2***. Due to the attack on the system, the controller is not able to stabilize the system to any reference temperature, which can be observed on the top plot. Thus, the temperature measurements fluctuate irregularly causing the Detector DT to detect abnormality most of the time, shown in the middle plot. Using the consistency measures and the STL monitoring, the Consistency DT analyzes the measurements due to the requests from the AD-DT and the Detector DT. On the bottom plot of Fig. 6.9 we see that the Cybersecurity DT identifies attacks on the inconsistent measurements from the process. It is important to note that the Cybersecurity DT finds normal or anomalous measurements in the data stream at times where the measurement signal is similar to the transient response of the nominal process or the abnormal signals of the anomalous process. Annotation 1 on the bottom plot shows the missed positive by the Robust OSVM when compared to the Robust OSVM+. While the effect of Robust OSVM+ is attenuated in this case, we still see that when compared to the Robust OSVM, the Robust OSVM+ provides slightly better attack detection performance. The anomalous predictions between $100 - 150$ seconds and $170 - 210$ seconds exhibit similar behavior to the anomaly case ***A2***, which results in the Cybersecurity DT predicting anomalies instead of attack signals in those intervals. As mentioned earlier, the recommendation of an attack on the system is shared with an attack classifier (either to an SME or to additional data analysis DT) for further analysis.

6.5.5.3 Discussion

The results in this section show that the Cybersecurity DT identifies inconsistencies throughout the signal, which predicts that the signal is most likely compromised, i.e., attacked. Furthermore, when used for anomalous signals and attack signals interchangeably, the Cybersecurity DT was able to correctly analyze and identify an attack on the system without any need for parameter adjustment or model re-training. Due to the way we have implemented the STL specifications for consistency monitoring, we have observed that some attacked signals are partially missed or identified as anomalous especially around setpoint changes. This is an important tradeoff in the framework design that we favored to provide robustness when the process is in nominal condition (no anomalies or attacks). AM processes often take multiple hours to manufacture a 3D object. Therefore, false alarms that would result in stopping a process prematurely could be costly (e.g., time and material lost due to a false alarm at the end of a multi-hour print job). Consequently, our design is aimed toward reducing false alarms on the physical process. Various extensions that allow for higher false alarm rates may be utilized for processes where the cost of missing an attack

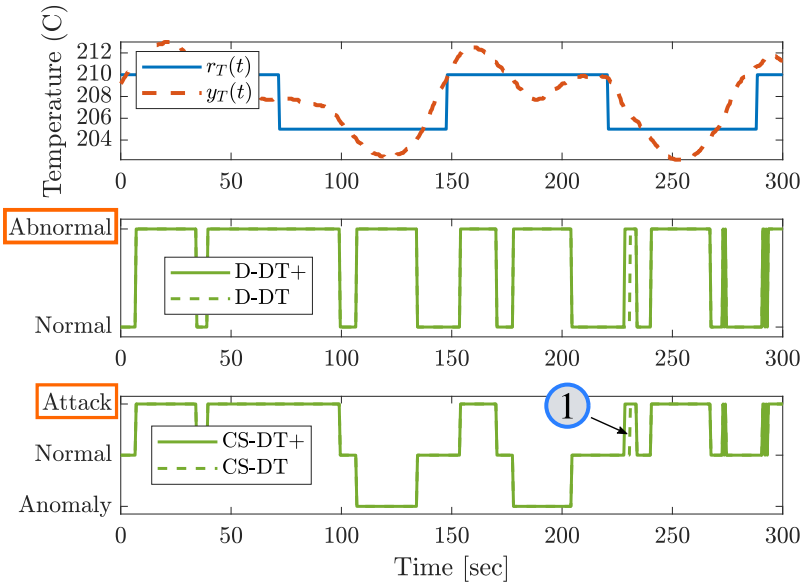


Figure 6.9: Illustration of the Detector DT abnormality detection and the Cybersecurity DT attack detection during the attack ***T2***. **Top** plot shows the temperature response of the process and the reference temperatures. **Middle** plot shows the Detector DT outputs for the Robust OSVM and Robust OSVM+ procedures with D-DT and D-DT+ respectively. **Bottom** plot shows the Cybersecurity DT outputs when the Robust OSVM and Robust OSVM+ procedures are utilized with CS-DT and CS-DT+ respectively. The rectangles around Abnormal and Attack indicate the ground truth for the signals. Annotation 1 is discussed in the text.

is much higher than prematurely stopping the process.

An important observation about the attack detection results indicate that the detection performance may be improved by considering a latched process detection. In our presented work, all detections are conducted by comparing the datapoints to the expected normal boundaries; however, by incorporating the state of the process as anomalous, normal, or attacked, further methods can be developed to instead look at the transition from one state to another. Such detection methods may require additional modeling and data analysis, and would be an extension of the proposed framework.

6.6 Further Applications on Spatial DTs for AM Processes

The Cybersecurity DT study illustrates a general purpose framework to monitor cyber-physical systems to detect anomalies, and cyber-attacks in the context of closed loop controllers. In this section, we utilize the presented DT framework architecture to demonstrate further implementations for performance monitoring of spatiotemporal dynamics of an AM process. Performance monitoring is a key application for AM processes to assess

the process efficiency and the expected finished part quality. Therefore, we describe a complementary application domain to the cybersecurity application and demonstrate how the proposed DT framework with multiple DTs may be utilized in industry to monitor and analyze high-performance spatiotemporal AM processes.

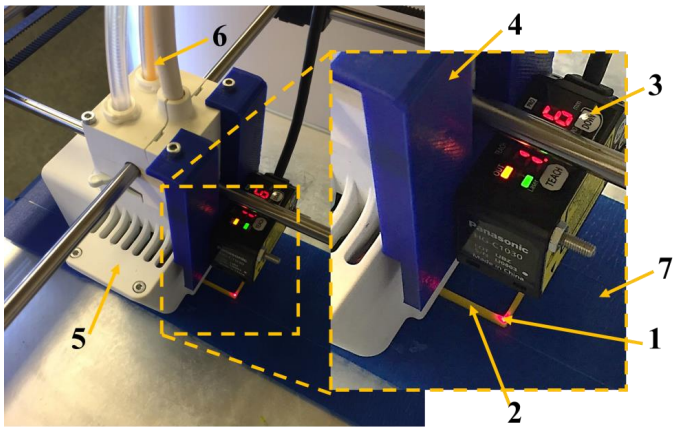


Figure 6.10: Implementation of the laser distance measurement sensor. This setting is also presented in Fig. 3.5 in Chapter III. **1** - laser measurement point, **2** - square shell build geometry, **3** - laser distance measurement sensor, **4** - mounting piece for the sensor, **5** - extruder head of the FDM printer, **6** - PLA filament used in the experiment, **7** - heated build plate with the painters tape.

The spatial dynamics of the process represent the height of the printed part at each layer, as a function of space. To measure the spatial state of the system at each printed layer, a customized sensor integration is utilized illustrated in Fig. 6.10. A laser distance measurement sensor is mounted on the extruders and the GCode is modified to scan a layer after it is printed. Fig. 6.10 shows the setup while measuring a layer of a shell geometry. For spatial measurement, a painter's tape is used to minimize reflections from the build plate, thus minimizing measurement noise.

The laser measurements are passed to the Feature DT, which translates the temporal signal into a spatial representation to model the height of the top layer as a function of space. The top panel on Fig. 6.11 illustrates real-time measurements of given shell geometry, implemented through a MATLAB interface. The spatial DT evaluates the measured profile as well as a projected layer height after five layers into the future based on the model presented in [13]. The projection model is implemented as a DT replacing the AD DT and Cybersecurity DT in Fig. 6.5. The spatial analysis DT utilizes a model with the spatial measurements from the Feature DT to predict the height profile of the process at a future layer. A simple noise model is used to provide uncertainty for the predicted layer

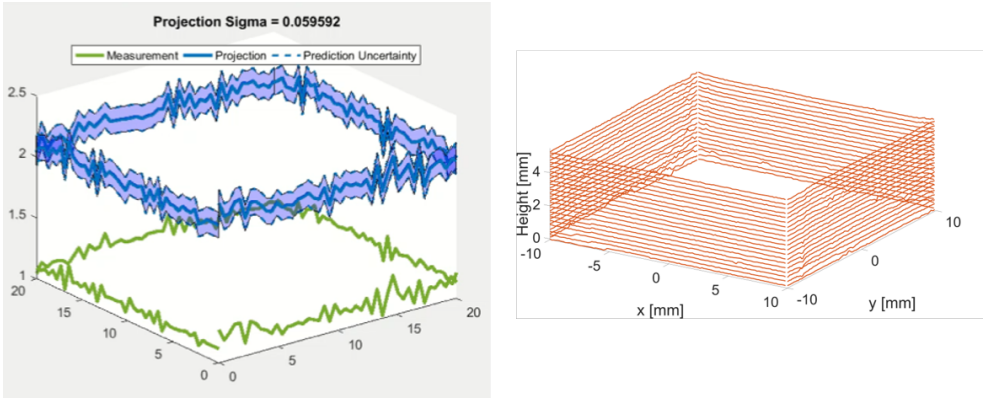


Figure 6.11: **Left:** Real-time spatial height measurement of the laser sensor and projection of the spatial height measurements to five layers into the future with prediction uncertainty data panel in MATLAB. **Right:** Spatial height measurements of a 20 layer shell geometry for offline data analysis.

height, shown around the projected spatial state. The bottom panel on Fig. 6.11 illustrates the data collected from a 20 layer printing process after some additional offline data processing. This data is used to analyze part performance and derive control actions for the printing process for future prints. Additionally, the layer-to-layer height profile data enables detailed analysis on the internal structures and micro-defects that may be invisible in the finished part. Therefore, spatial monitoring enables great insight on the part quality and functionality, which is essential for industrial applications. This type of analysis is of great importance for in-line quality control and virtual metrology applications. Additionally, by utilizing future projections of the process, it is possible to predict if an AM process is expected to conform to design tolerances and take preemptive action accordingly. Tolerance conformance prediction problem is studied in detail in Chapter IV.

Additional DTs for performance monitoring on the spatiotemporal dynamics of an AM process with low-level implementation details are presented in Appendix C. We present a real-time implementation study for a number of monitoring DTs working with the spatiotemporal process to present performance metrics in real-time. We note that the DT framework presented in this chapter is capable of accommodating the DTs presented in Appendix C as they are specializations of the general purpose DT architecture and framework in this chapter.

6.7 Chapter Conclusions

In this chapter, we presented a general-purpose DT framework for cyber-physical manufacturing systems and presented a detailed application for cyberattack detection on CPMS

in the context of closed-loop controllers and anomalies in the physical process. The main contribution of this chapter is a digital twin framework that is flexible and extensible to incorporate various models and data structures for run-time analysis of cyber-physical manufacturing systems (*C4*). The proposed Cybersecurity DT is able to detect and distinguish attacks and anomalies on the system while the process is controlled to switching setpoints. Furthermore, the proposed framework is capable of integrating with existing solutions in practice and leveraging subject matter expertise. Our approach is platform agnostic and modular thanks to its DT-centric design. We demonstrated our approach on an off-the-shelf 3D printer on which we implemented a novel network controller. Our demonstration illustrates the utility of the approach and proposes an architecture for scalable implementations in the context of AM Fleet applications. Then, we presented a specialization of the presented framework in the context of spatiotemporal AM processes. Experimental results illustrate how DT technology can enable monitoring and online analysis for AM processes, which in turn enables higher visibility, security, and reliability in industrial applications. The monitoring and analysis steps provide a baseline for practical implementations of the developments presented in the previous chapters since monitoring provides us with interfaces to interact and possibly control the spatial process. The components of the framework can be extended for multiple resources in a manufacturing system and aggregated into a system-level DT framework. The system-level DT framework solution can be utilized within the SDC-AM control architecture presented in Chapter V.

CHAPTER VII

Conclusions and Future Directions

As described in Chapter I, spatial processes emerge in many engineering and science applications as a powerful modeling formalism to consider system dynamics evolving as a function of space. An important and exciting application of spatial processes is additive manufacturing (AM), where a 3D object is manufactured in a layer-wise fashion. This dissertation focuses on modeling and control of spatial processes with a specific emphasis on AM as a motivating example. The unique manufacturing capabilities provided by AM have enabled a new paradigm of digital manufacturing, where production capabilities may be highly customized and decentralized since AM processes produce near-net-shape final products. When multiple AM processes are utilized together for improved capacity, efficiency, and yield, they are named AM Fleets. We consider AM Fleets as system-level extensions and generalizations of spatial processes in the form of spatially distributed systems. AM Fleets have been successfully utilized in industry to improve production capabilities. Furthermore, AM has been a crucial decentralized manufacturing capability worldwide during the COVID-19 pandemic, where individuals with AM capabilities have contributed to the production of crucial safety equipment during the catastrophic disruptions in industry. The utilization of AM processes in such critical and disruptive scenarios illustrates how AM capabilities in manufacturing can enable novel manufacturing systems that are agile, robust, and highly reconfigurable in the face of disruptions.

However, AM processes are, in general, difficult to model, analyze, and control due to the complex physics involved in the process. Additionally, there is a lack of consensus on practical considerations such as sensors, control architectures, control authority, and performance measures. The lack of knowledge on the process analysis leads to poor reliability and repeatability of AM processes in industry, which is often addressed by tuning individual machines to process a single product. While effective in practice, such approaches are difficult to scale and contradict the main promise of AM processes enabling greater flexibility and customizability with minimal setup costs. To address the aforementioned issues,

this dissertation proposes a comprehensive approach to model, control, and analyze AM processes. Furthermore, a system-level framework to efficiently model, monitor, and control AM Fleets is presented to complement the developments at the process level. In the rest of this chapter, we restate the main contributions of the dissertation, identify limitations, provide future research directions, and discuss the greater impact of the contributions.

7.1 Contributions

The four primary contributions of this dissertation are stated below.

C1 - A unifying control-oriented modeling framework for spatial AM processes: The first main contribution of this dissertation is the development of a modeling framework and a notion of layer-to-layer stability to characterize the performance of the layer-to-layer spatial AM processes. The spatial modeling framework and layer-to-layer stability metrics are presented in Chapter III. We present a spatial modeling framework for the layer-to-layer spatial dynamics of an AM process, which is later used for closed-loop controller development. Additionally, the layer-to-layer stability metrics provide a spatial performance measure to quantify if a printed part conforms to physical specifications such as dimensional tolerances. We demonstrate the concepts on a fused deposition modeling process and provide experimental results.

C2- Novel layer-to-layer control methods that utilize spatial models of AM processes: The second contribution is the development of novel control architectures that utilize the presented modeling framework to ensure layer-to-layer reachability, stabilizability, and reference tracking in the context of control constraints. Novel approaches leveraging the positivity and monotonicity of spatial height dynamics are presented for controllers that are scalable to large spatial domains. The layer-to-layer controllers and their applications are presented in Chapter IV. The novel approaches presented in Chapter IV provide layer-to-layer closed loop controllers to track desired spatial height references and to layer-to-layer stabilize the spatial process.

C3 - A system-level control approach for spatially distributed AM processes: The third contribution of this dissertation is a system-level centralized control framework that employs novel control and analysis methods for run-time closed-loop scheduling control and knowledge transfer/reuse in AM Fleets. A system-level centralized control framework and its applications in scheduling and knowledge transfer are presented in Chapter V. We present a system-level centralized control framework that utilizes digital twins of the resources to obtain a run-time centralized view of an AM Fleet. We then illustrate how the centralized knowledge may be utilized for run-time closed-loop scheduling applications

and knowledge transfer applications.

C4 - An extensible digital twin framework for monitoring and analysis of spatio-temporal processes: Many of the presented contributions are enabled by the availability of run-time models and representations of the physical processes, which are utilized for multiple purposes. To this end, we develop purpose-driven digital twins to efficiently synthesize run-time data to model, analyze, and control cyber-physical systems. The fourth main contribution is a digital twin framework that is flexible and extensible to incorporate various models and data structures for run-time analysis of cyber-physical manufacturing systems. We demonstrate the digital twin framework on an extensive cybersecurity application implemented on an AM process and provide further implementations for analyzing and performance monitoring on spatio-temporal process data. The digital twin-based process monitoring and analysis framework is presented in Chapter VI. We demonstrate several applications implemented on an off-the-shelf 3D printer to illustrate the utility and flexibility of the presented framework. The presented experimental results enable run-time verification of cyber-physical manufacturing systems.

There exist numerous research directions and open questions for future work building on the knowledge synthesized in this dissertation. The rest of the chapter outlines the limitations of the presented work, three future research directions, and the greater impact of this dissertation on the science and engineering community.

7.2 Limitations

This dissertation has explored modeling and control for spatial processes and spatially distributed systems within the context of the four main contributions *C1-C4*. A number of limitations and future work remain related to all contributions.

The control-oriented modeling framework in Chapter III presents a layer-to-layer model for a general AM process. While spatial input-output (material input to spatial height output) models for many AM processes have been proposed in the literature, system identification remains a challenge for many practical applications. Developing a framework for systematic system identification methods for spatial models in AM processes is an important challenge to be addressed. Additionally, spatial location and layer dependence of the layer-to-layer models may be necessary depending on the applications of interest and the spatial geometry in the process. Furthermore, we provide examples of layer-to-layer stability bounds defined based on the dimensional tolerances of simple geometries, but in practice, the dimensioning and tolerancing for the printed parts may be layer and geometry-dependent. Further studies on how to develop effective layer-to-layer stability bounds to

ensure part dimensionality and possibly other mechanical properties through further modeling efforts will greatly improve the applicability of the proposed models in practice.

Another important challenge is the practical application of layer-to-layer controllers for AM, such as the ones presented in Chapter IV. As also outlined in Chapter V, current AM systems lack a uniform architecture to interface and control the process in run-time. For many processes, closed-loop control applications are not possible due to a lack of in-situ sensing, or sufficient control authority. To enable a more general framework for AM spatial control in practice, necessary components (for sensing and actuation) must be identified and standardized in the industry. Establishing a more uniform landscape for control authority and sensing of various AM processes will enable new applications for process control in practice. Additionally, the controllers proposed in Chapter IV have been demonstrated in simulation studies due to the difficulties in implementing experimental controllers on off-the-shelf FDM machines. Implementation, experimentation, and validation of the proposed controllers on commercially available machines (e.g., off-the-shelf FDM printers) is a practical limitation of the presented work, and is a promising direction for future work.

Another limitation of the presented work is the increased computational overhead due to the data-intensive and large-scale datasets produced by AM processes and spatial processes in general. In practice, a tradeoff between modeling fidelity and computational availability is made based on the application of interest. AM processes and DTs can produce large amounts of data to be processed and analyzed since the dimensionality of the spatial data and models scale by the spatial resolution. For example, for a spatial resolution of 100 microns, a square spatial domain of just 10 millimeters results in 10201 spatial locations per layer, which scales further with the number of layers in the process. Thus, data monitoring, analysis, and control for the spatial locations become a challenging big data analysis problem that needs to be handled efficiently. Developing efficient computational tools to analyze run-time data streams and control applications is a crucial step to enable the next generation of online analysis tools.

An important limitation for the centralized system-level framework for AM fleets is the applicability and availability of necessary data protocols to collect data and implement control on AM processes. As discussed earlier in Section 5.1, proprietary data and control architectures of various AM machines make it difficult to collect uniform data streams from all machines in an AM Fleet with heterogeneous machines. Additionally, the knowledge transfer methods described in Chapter V rely on the necessary maps being defined across processes in the AM Fleet, which is a challenging research task in practice. Further research work in this direction will enable practical implementations of the presented approaches in industry.

The DT framework and cybersecurity applications presented in Chapter VI provide a baseline framework that is extensible for further applications. The presented cyber-security application relies on certain logical propositions provided by a subject matter expert. However, developing such propositions may be difficult and labor-intensive in practice. Developing efficient methods to systematically build monitoring conditions for on-line analysis applications applications and leveraging data-driven approaches will help to scale the proposed methods to multiple machines and attack types. Additionally, the reusability of monitoring specifications across machines in a fleet, and how the proposed methods scale for detecting system-level anomalies are promising future research directions. Furthermore, specifications for monitoring spatial process data such as the example given in Chapter VI should be developed to detect attacks and anomalies on the spatial domain.

7.3 Future Research Directions

While a number of limitations and research questions can be identified for the main contributions of this dissertation, three promising research directions are presented next as potential directions that can leverage the research in this dissertation.

7.3.1 In-situ Verification and Control of AM for Online Quality Assessment

An important practical requirement for manufactured parts is to conform to specified mechanical properties, e.g., tensile strength, dimensional tolerance, porosity, etc. Quality assessment and assurance with AM is often challenging since the layer-to-layer process may be prone to imperfections. Additionally, due to the issues in repeatability and reliability, AM-produced parts may have high reliability, which makes it difficult to do batch sampling (in many cases, the parts are bespoke and not even in batches). The difficulty in verifying AM printed parts is an important challenge for utilizing AM in the industry. While certain designs may be repeatedly tested and verified by batch producing on a given machine and material combination, verification for customized parts remains difficult. Additionally, many testing routines may depend on destructive testing of printed parts which is cost-intensive. The DT frameworks presented in Chapter VI and Appendix C illustrate how DTs may be utilized for online monitoring in conjunction with formal specification methods to prescribe desired behavior for the process and the produced part. These approaches may be extended to develop online verification methods that can verify an AM printed part in-situ. To build such tools, extensive research must be done on understanding the key parameters that influence mechanical properties. By monitoring the key parameters

and building DTs to estimate and monitor the related mechanical properties, in-situ quality verification may be possible.

While monitoring enables quality assessment for the processes, the spatial modeling and control methods may be utilized in a similar manner to ensure part quality. Chapter III and Chapter IV discuss layer-to-layer stability as a measure of dimensional performance of the printed part. By extending the definition of a state from a heightmap to other properties of interest and providing layer-to-layer measurements of the process, it may be possible to build controllers that ensure conformance to mechanical properties for AM-produced parts. Developing fundamental knowledge on the control models for key parameters and developing an efficient method to implement such controllers in the context of varying control authority and sensing availability is an important challenge. Addressing research questions in this direction will enable the next generation of high-performance and efficient AM processes.

7.3.2 Modeling and Control of AM Spatial Dynamics as Monotone Systems

The use of monotone operators for efficient controller computation is presented in Chapter IV. Understanding the additional structure of monotonicity in the model enables efficient computation methods for large-scale nonlinear system dynamics. Additionally, since these methods utilize the nonlinear dynamics equations without additional approximations, they provide an exact solution of the nonlinear system, which is expected to outperform linear approximations in practice. In fact, monotone systems and monotone operator theory both have rich literature with many novel applications that enable scalable control and verification, which may be leveraged in the AM research domain. Considering AM spatial dynamics as monotone operators, we may scale control approaches to large-scale applications by leveraging monotone operator theory. Immediate research applications are decentralized and distributed modeling frameworks to consider the spatial AM process in multiple spatial sub-domains to develop efficient computation methods. By leveraging the computational efficiency of the monotone dynamics, it may be possible to develop scalable model predictive control approaches with the nonlinear dynamics to provide additional robustness to the process. This is an interesting research direction since model-based control with nonlinear dynamics is a difficult problem with the computation of the exact solutions not always feasible due to computational complexity.

Within this scope, developing frameworks that account for model uncertainty and noise with the nonlinear models is an important extension of the presented work. Extensions in such directions may be leveraged from solutions in the literature on related problems in monotone operator theory and optimization theory. Therefore, understanding and de-

veloping new models that satisfy the monotonicity or positivity (in the linear case) is a promising research direction. Especially, characterization of uncertainty in the developed models to preserve monotonicity and developing theory on layer-to-layer stability-ensuring control applications to control for the functional properties of the produced part will enable practical and efficient control methods for AM processes that scale well. Future research directions could also utilize ILC methods similar to those presented in Chapter IV on the monotone spatial dynamics to provide theoretical guarantees on robustness and stability for the closed-loop system. Lastly, layer-to-layer stability and finite stability in the context of finitely many layers poses an interesting research challenge on designing controllers that provide guarantees in a finite regime. Further research on the implications and possible guarantees that can be provided for such applications will, in turn, enable enhanced stabilizability and reachability analysis on the spatial dynamics. Applications in this direction, leveraging underlying system properties such as monotonicity, will enable intelligent decision-makers that monitor, analyze, learn, and predict process behavior in future layers in an accurate manner. By utilizing such next-level decision-makers for AM Fleets, an under-performing process is either controlled to be layer-to-layer stable or if the process is not stabilizable, it is stopped prematurely to avoid time and cost. Such applications are essential, especially in system-level AM Fleet scenarios.

7.3.3 Knowledge Transferring Control for Data-rich Systems

As mentioned previously, parameter tuning for AM processes is a labor-intensive and difficult process. Chapter V presents the SDC-AM framework as a centralized system-level control framework that utilizes digital twins in conjunction with a knowledge base for systematic modeling and storage of process models and past measurements. The application of knowledge transferring control leverages the data from the previous runs of the same process or similar processes to improve controller evaluation. The application may be extended to other data-rich processes such as power networks, multi-robot systems, and autonomous driving. Developing theoretical foundations on the performance analysis of utilizing model information alone versus model information enriched with the knowledge transferring control approaches will enable a new control paradigm where measurements from similar processes may be utilized for control to improve controller performance. Within this context, measures such as convergence rate and domains of attraction for the closed-loop controllers utilizing transferred data versus model-only evaluation should be quantified for practical applications.

By utilizing knowledge transferring control on spatially distributed systems, distributed learning applications may be developed such that the group of resources (e.g., the AM

Fleet, group of robots over a spatially distributed domain) collectively learn optimal control policies from each other. Similar concepts have been explored in the context of exploration and cooperation in multi-agent systems. Here, we can extend the application domain in manufacturing by explicitly utilizing redundancies in the system to ensure product quality and reliability. For example, if we have a mapping to translate the measurements from one machine to another as an application of knowledge transfer, we may utilize a relatively inexpensive process to print a certain high-value product to collect data. Then, the collected data, i.e., the knowledge, can be utilized on the high-performance machine by utilizing the mapping between the processes. In practice, evaluating such maps is not always feasible. Thus, an important research direction is to explore structures for such maps and develop a fundamental understanding of how they can be evaluated. Machine learning methods may be utilized to evaluate mappings between the dynamics of various processes in a fleet. Learning such mappings, in turn, will improve the reliability and repeatability of each resource and improve the yield of the fleet.

7.4 Outlook and Greater Impact

This dissertation has focused on AM processes and AM Fleets to provide examples and developments on spatial processes and spatially distributed systems. The contributions of the dissertation enable efficient modeling, control, and online monitoring methods for AM processes, which in turn improve the quality, yield, repeatability, and reliability of AM processes. The development of digital twin architectures for cyber-physical manufacturing processes enables the detection of anomalies and attacks for secure and verified AM processes. The system-level centralized modeling and control framework provides a baseline architecture to utilize multiple AM processes for improved utilization and throughout industrial applications. Overall, the contributions of this dissertation improve the knowledge in the field of modeling, monitoring, analysis, and control for AM processes.

The developments and contributions in this dissertation have a greater impact on the literature as the developed methods and knowledge apply to a broader range of processes. Since the presented work in this dissertation has focused on applications in additive manufacturing, the application of the presented methods to related fields requires the development of proper extensions for the related fields. Spatial processes and the theory developed in Chapter III and Chapter IV apply to other manufacturing processes such as welding, milling, and hybrid manufacturing (material addition and removal) by considering the layer-to-layer process in terms of a pass-to-pass process. While there are inherent differences between the layer-to-layer dynamics of additive manufacturing and the pass-to-pass

material removal process, extensions of the proposed methods can be developed to apply novel modeling and control applications that perform spatial reference tracking. By extending the proposed modeling and control methods for spatial processes, novel control applications for various manufacturing processes may be developed. Additionally, industrial robotics applications may be considered as spatial processes that are repetitive in nature. Extensions of the contributions in Chapters III-IV may result in novel modeling and control methods for industrial robotics. The concept of spatially distributed systems is also general enough to apply to many types of multi-agent systems, supply chains, connected automated vehicles, and smart power grids. The centralized control framework and the control approaches developed in Chapter V may be utilized for these other systems with appropriate extensions. Lastly, the DT framework and applications presented in Chapter VI apply to general cyber-physical systems with applications of monitoring, analysis, verification, control, and cybersecurity. While DTs have been largely used in manufacturing applications, they are gaining attention in other fields as well. Utilizing the proposed frameworks and architecture in other application domains such as aerospace, automotive, autonomous vehicles, robotics and robot fleets, etc., creates novel applications and research directions. As an example, a fleet of connected autonomous vehicles for transportation applications may be considered as a spatially distributed system where instead of resources in an AM Fleet, we now consider the individual vehicles and how they accomplish transportation tasks. The centralized control framework with run-time scheduling applications developed in Chapter V may be utilized in this setting. Additionally, digital twins of the vehicles may be utilized to gather run-time information to understand vehicle status, capability, range, etc. Formalizing the presented work from this dissertation in the context of these application domains and developing a fundamental understanding and theory about the underlying concepts will potentially have a high impact on multiple applications, resulting in improved system performance and efficiency.

APPENDICES

APPENDIX A

Details of the Chapter III Case Study Model

In this appendix we present additional details of the spatial model from the case study of Chapter III. We illustrate the use of shape functions to build matrix representations of the cross-sectional geometry of the FDM process over a spatial domain of interest. The contents of this appendix are presented in [16].

A.1 Shape function and input for FDM

An ellipsoid shape function with the major (a) and minor (b) radii and $\boldsymbol{\theta} = [a, b]$ is chosen for the FDM process. The shape function for the FDM process is given by

$$c(p, \boldsymbol{\theta}, b, y') = \frac{1}{2b}\left[\sqrt{b^2(1 - \frac{\Delta y}{a^2})} + b\right] \tag{A.1}$$

where $y' \in [y - a, y + a]$ and $\Delta y = ||y - y'||_2^2$ is the Euclidean distance from a deposition point $y \in p$ in the $\hat{\boldsymbol{j}}_{\boldsymbol{P}}$ direction, and the function is zero everywhere else.

Single bead width is measured as $0.36mm$ on average with caliper measurements. The square deposition path on Λ is given by the following spatial deposition points.

$$\begin{aligned}\bar{p}(k, \bar{\gamma}(\Lambda)) = \{&\boldsymbol{\lambda}(1,1), \ldots, \boldsymbol{\lambda}(1,126),\\ &\boldsymbol{\lambda}(2,126), \ldots, \boldsymbol{\lambda}(126,126),\\ &\boldsymbol{\lambda}(126,125), \ldots, \boldsymbol{\lambda}(126,1),\\ &\boldsymbol{\lambda}(125,1), \ldots, \boldsymbol{\lambda}(2,1)\}.\end{aligned} \tag{A.2}$$

Define $\boldsymbol{P} \in \mathbb{R}^{n_i \times n_j}$ with ones for the spatial locations in Λ with deposition, defined by $\bar{p}(k, \bar{\gamma}(\Lambda))$ and zeros elsewhere, and define $\boldsymbol{p} = vec(\boldsymbol{P})$.

A.2 Kernel Bases for deposition Bead Modeling

The concept of kernel basis matrices is introduced in [13] to evaluate $\bar{\boldsymbol{c}}_i$ with the correct spatial location and orientation. A kernel basis matrix has the nonzero entries that correspond to the discretized heights of the shape function at a specific orientation (e.g. $0, \pi/2$) based on the deposition path. Leveraging the adjacency and the square shape of the deposition path, following 3×3 kernel basis matrices are given. The shorthand $c(y') = c(p, \boldsymbol{\theta}, b, y')$ is used for notational brevity.

$$\boldsymbol{K}_1 = \begin{bmatrix} 0 & 0 & 0 \\ c(y+\alpha_i) & c(y) & c(y-\alpha_i) \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{K}_2 = \begin{bmatrix} 0 & c(y-\alpha_j) & 0 \\ 0 & c(y) & 0 \\ 0 & c(y+\alpha_j) & 0 \end{bmatrix} \quad \boldsymbol{K}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & c(y) & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Since $\alpha_i = \alpha_j$ and the bead function is symmetric, $\boldsymbol{K}_1 = \boldsymbol{K}_2^T$ in this example. To understand the effect of corner overflow in the case study, the deposited parts are observed under a microscope. As a result, $\boldsymbol{K}_3$ represents the corners of the square shell.

The $\tilde{\boldsymbol{c}}(\boldsymbol{s}_m)$ matrices for spatial inputs are evaluated in the following way for each deposition point.

1. For a point p on the deposition path $\bar{p}$, a corresponding kernel basis matrix $\boldsymbol{K}_i$ is determined.

2. Taking the center entry of $\boldsymbol{K}_i$ as the spatial location corresponding to p in $\boldsymbol{\Lambda}$, the matrix is padded with zeros to have the appropriate dimensions $\mathbb{R}^{n_i \times n_j}$.

Using the $\tilde{\boldsymbol{c}}(\boldsymbol{s}_m)$ matrices, input matrix $\boldsymbol{B}_k$ is evaluated as given in (3.5). For the spatial locations with multiple overlapping $\boldsymbol{K}_i$ matrices, a saturation function is utilized to ensure that the control input is appropriately applied to the spatial dynamics.

APPENDIX B

Details of the First-order Logic Representation in Section 5.2

This appendix overviews the methods used to express a priced timed automaton (PTA) in first-order logic (FOL) in Chapter V, Section 5.2. Details of the model development for the optimization problem for the model predictive control application in Chapter V, Section 5.2 are given in the following.

B.1 First-order Logic Representation of the Graph

Suppose each location in Q is represented as the j^{th} standard basis vector in $\mathbb{B}^{n_q}$, $\mathbf{q}^j$, where $n_q = |Q|$ and $\mathbb{B} = \{0, 1\}$. Similarly, each edge in E can be represented as the jth standard basis vector in $\mathbb{B}^{n_e}$, $\mathbf{e}^j$ where $n_e = |E|$. Then, let $A \in \{-1, 0, 1\}^{n_q \times n_e}$ represent the incidence matrix of $\mathcal{A}$ where an element $A(i, j) = 1$ (positive incidence) indicates that $\mathbf{e}^j$ transitions to $\mathbf{q}^i$ and $A(i, j) = -1$ (negative incidence) indicates that $\mathbf{e}^j$ transitions from $\mathbf{q}^i$.

Define the input transition matrix $B_{in} \triangleq \max(A, 0)$, where $\max$ is computed element-wise for each element of the first argument and $\triangleq$ denotes a definition for the left-hand side, that maps an edge to the location it transitions to. Then, $\mathbf{q}_{i+1} = B_{in}\mathbf{e}_{i+1}$ where $\mathbf{e}_{i+1}$ is the edge from a location $\mathbf{q}_i$ to $\mathbf{q}_{i+1}$, and $\mathbf{q}_{i+1}$ is the location following the discrete transition. Similarly, define the output transition matrix $B_{out} \triangleq \max(-A, 0)$ that maps an edge to the location from where it transitioned. Therefore, $\mathbf{q}_i = B_{out}\mathbf{e}_{i+1}$.

Next, define $\tilde{A} \triangleq \max(-A^T B_{in}, 0)$ as the matrix that maps an edge $\mathbf{e}^j$ into the set of edges that are enabled after the execution of $\mathbf{e}^j$. Here, enabling is defined in terms of connectivity and not the guards and invariants, so that if an edge is negatively incident to

a location, then that edge is enabled at that location. A vector of edges enabled following the traversal of any edge $\mathbf{e}_i$ can thus be defined as $\tilde{\mathbf{e}}_{i+1} = \tilde{A}\mathbf{e}_i$ with initial condition $\tilde{\mathbf{e}}_1 = \max(-A^T, 0)\mathbf{q}_0$.

To constrain the discrete transition to one of the possible choices in the vector $\tilde{\mathbf{e}}$, the "multiple exclusive or" Boolean function is introduced:

$$\texttt{mXor}(\mathbf{a}, \mathbf{b}) \triangleq \mathbf{a}^T\mathbf{b} == 1 \tag{B.1}$$

where $\mathbf{a}$ and $\mathbf{b}$ are binary column vectors of equal length. This function enables the encoding of the automaton connectivity as a set of first-order logical constraints on the edges taken during discrete transitions: each edge $\mathbf{e}_i$, denoting the i^{th} discrete transition in a path, must satisfy $\texttt{mXor}(\mathbf{e}_i, \tilde{\mathbf{e}}_i)$.

Finally, a first-order logic MPC horizon, N_{mpc}, is defined as the length of the horizon for the MPC formulation. The decision variables of the first-order logic MPC are the discrete transitions of a path on a PTA. Then, the sequence $U = \{u_1, \ldots, u_{N_{mpc}}\}$ denotes the decision variables for the first-order logic MPC problem, so that $u_i \in \mathbb{B}^{n_e}$.

B.2 First-order Logic Representation of Time

The guards and invariants of the set $\mathcal{B}(C)$ of the PTA must also be represented in first-order logic. The PTA has two clocks: a global clock and a local clock. Local clock valuations are denoted with $c_l(\mathbf{q}^j)$ and represent the time spent at a given location. Valuation of a global clock at state $\mathbf{q}^j$ is denoted with $c_g(\mathbf{q}^j)$. A global clock valuation represents the time spent since the beginning of a path. The vector of all clock valuations is defined as $val(\mathbf{c}) = [c_l(\mathbf{q}^0), c_g(\mathbf{q}^0), \ldots, c_l(\mathbf{q}^{n_q}), c_g(\mathbf{q}^{n_q})]^T$. This schema allows the encoding of the algebraic constraints in $\mathcal{B}(C)$ in an algorithmic way for each state in the PTA.

By assumption, the global clock of a PTA is never reset. This poses a consistency constraint on the valuations of the global clock at different $\mathbf{q}^i$, such that the global clock valuation at $\mathbf{q}^i$ must be $c_g(\mathbf{q}^i) = c_l(\mathbf{q}^i) + c_g(\mathbf{q}^m)$ if and only if a discrete transition from $\mathbf{q}^m$ to $\mathbf{q}^i$ is taken in a path (i.e. $(c_g(\mathbf{q}^i) = c_l(\mathbf{q}^i) + c_g(\mathbf{q}^m)) \Leftrightarrow e^j$, where e^j is the edge from $\mathbf{q}^m$ to $\mathbf{q}^i$). Algorithm 3 presents the global clock valuation constraint assignment. The assignment is performed between all the locations in the PTA except for the initial location q_0. Let $\delta_{in} \in \mathbb{B}^{n_e}$ denote a vector that corresponds to the edges with positive incidence to $\mathbf{q}^i$. Then, each 1 in the vector δ_{in} corresponds to an $\mathbf{e}^j \in E$. So, if $\delta_{in} \in E$, there is a single incoming edge (positive incidence) to the location $\mathbf{q}^i$. The constraint given in line 6 is assigned for the single incoming edge case. If δ_{in} contains multiple ones, the

Algorithm 3 Global clock valuation constraint assignment

1: **function** GLOCLOCK($E, B_{in}, B_{out}, Q, val(\mathbf{c}), U$)
2: *Initialize*: $\Phi \leftarrow \{\}, \Lambda \leftarrow \{\}$
3: **for all** $\mathbf{q}^i \in Q \setminus \mathbf{q_0}$ **do**
4: $\delta_{in} \leftarrow B_{in}^T \mathbf{q}^i$
5: **if** $\delta_{in} \in E$ **then** $\mathbf{q}_m = B_{out}\delta_{in}$
6: $\Lambda \leftarrow \Lambda \cup \{c_g(\mathbf{q}^i) = c_l(\mathbf{q}^i) + c_g(\mathbf{q}^m)\}$
7: **else**
8: Decompose $\delta_{in} = \sum_j \mathbf{e}^j$
9: **for all** $\mathbf{e}^j \in \delta_{in}$ **do** $\mathbf{q}^m = B_{out}\mathbf{e}^j$
10: $\phi_m \leftarrow \{c_g(\mathbf{q}^i) = c_l(\mathbf{q}^i) + c_g(\mathbf{q}^m)\}$
11: $\Phi \leftarrow \Phi \cup \{\bigvee_{k\in[1,N]} u_k(\mathbf{e}^j) \Leftrightarrow \phi_m\}$
12: **end for**
13: **end if**
14: **end for**
15: **Output:** $\mathcal{C} \leftarrow \Phi \cup \Lambda$
16: **end function**

vector is decomposed into the unit basis vectors $\mathbf{e}^j$ and the proposition ϕ_m is created for each $\mathbf{e}^j$. After evaluating the proposition ϕ_m, the constraint given in line 11 is evaluated. The constraint reads as: the proposition ϕ_m is true if and only if the discrete transition belonging to the edge e^j is executed within the MPC-horizon N_{mpc}. By using Algorithm 3, the consistency constraint on the global clock valuations is maintained.

Algorithm 4 presents the guard condition assignment for the clock valuations in $val(\mathbf{c})$. The assignment is performed for all locations in the PTA except for the initial location $\mathbf{q}_0$. Let the vector $\delta_{out} \in \mathbb{B}^{n_e}$ denote a vector that corresponds to the edges with negative incidence to $\mathbf{q}^i$. If δ_{out} contains more than one edge, it is decomposed into the edges it corresponds to ($\delta_{out} = \sum_j \mathbf{e}^j$). The algorithm first checks if the multiple negatively incident edges have the same guard condition g^{e^j}.

The proposition in line 11 of Algorithm 4 is assigned to ensure that $c_l(\mathbf{q}^i)$ will satisfy the guard condition if a discrete transition belonging to edge e^j is executed. To have a feasible satisfaction problem in first-order logical constraints, all the Boolean equation constraints of a system must evaluate true. The constraints evaluated at line 11 of Algorithm 4 make the equivalent guard conditions of two outgoing edges (negative incidence) infeasible when one of the edges has an associated discrete transition executed. To circumvent this situation for equivalent guard conditions on outgoing edges, the proposition in line 15 is evaluated. The proposition states that a local clock value satisfies the equivalent guard condition if and only if either of the edges with the equivalent guard condition is executed.

Finally, the invariants of the PTA can be encoded with first-order logic via linear con-

Algorithm 4 Guard condition assignment

function ASSGNGUARD($E, B_{in}, B_{out}, Q, val(\mathbf{c}), U$)
  Initialize: $\Xi \leftarrow \{\}, \Omega = \{\}$
  for all $\mathbf{q}^i \in Q \setminus \mathbf{q_0}$ **do**
    $\delta_{out} \leftarrow B_{out}^T \mathbf{q}^i$
    if $\delta_{out} \notin E$ **then** $\delta_{out} = \sum_j e^j,\ \Theta \leftarrow \{\}$
      for all $e^j \in \delta_{out}$ **do** $\theta \leftarrow e^j$
        if $g^{e^j+1} = g^\theta$ **then** ▷ equivalent guards
          $\Theta \leftarrow \Theta \cup \theta$
        else
          $\varepsilon \leftarrow (c_l(\mathbf{q}^i) \models g^{e^j})$
          $\Xi \leftarrow \Xi \cup \{\bigvee_{k\in[1,N]} u_k(e^j) \Leftrightarrow \varepsilon\}$
        end if
      end for
      $\varepsilon \leftarrow (c_l(\mathbf{q}^i) \models g^{e^j})$
      $\Xi \leftarrow \Xi \cup \{\bigvee_{e^j\in\Theta} \bigvee_{k\in[1,N]} u_k(e^j) \Leftrightarrow \varepsilon\}$
    else $\delta_{out} \in E$
      $\Omega \leftarrow \Omega \cup \{c_l(\mathbf{q}^i) \models g^{e^j}\}$
    end if
  end for
  Output: $\mathcal{G} \leftarrow \Xi \cup \Omega$
end function

straints. Each location of the PTA has a nonnegative number of invariants, η^i, associated with it. The invariants are put in the standard form as:

$$\Psi^i \begin{pmatrix} y \\ x \end{pmatrix} \leq \mathbf{t}^i \tag{B.2}$$

where x and y are the global and local clock valuations at a location, $\Psi^i \in \mathbb{R}^{\eta^i \times 2}$, and $\mathbf{t}^i \in \mathbb{R}^{\eta^i}$. The vectors representing all of the invariants can be built by letting $\Psi = diag(\Psi^1, \ldots, \Psi^{n_q})$ and $\mathbf{t} = [\mathbf{t}^1 \ldots \mathbf{t}^{n_q}]^T$ satisfying:

$$\mathbf{\Psi}\ val(\mathbf{c}) \leq \mathbf{t} \tag{B.3}$$

where $\Psi \in \mathbb{R}^{\eta \times 2n_q}$, $\mathbf{t}^i \in \mathbb{R}^\eta$, and η is the number of invariant constraints. This forms the first-order logic representation of the invariants of the PTA.

The proposed encoding scheme for the first-order logic representation of time allows for an efficient formulation of the MPC problem in which the clock valuations only at the discrete transitions are considered. Presented time encoding scheme differs from the existing first-order logic time encoding schemes in literature such as [30]. The existing encoding schemes encode the time of delay and discrete transitions of a PTA in fixed dis-

cretized intervals, resulting in an increased number of constraints to represent time when compared to the proposed scheme here. The first-order logic formulation of the PTA-MPC problem utilizing the proposed graph and time representations is described in Chapter V, Section 5.2.

APPENDIX C

DT Implementation Studies

This appendix presents an implementation study based on a specialized digital twin architecture presented in our work [15]. The implementation and the experimental results are given as further references to the DT framework presented in Chapter VI.

C.1 Further Applications in Performance Monitoring and Anomaly Detection for Spatiotemporal AM Processes

In this section we specialize the general DT framework proposed in Chapter VI for a specific AM process and present results on spatiotemporal monitoring. There has been recent work in modeling FDM processes, developing measurement technologies, and establishing verification techniques to improve the quality and reliability of AM manufactured parts [13, 160, 171]. Most of the current literature relies on customized sensing and measurement technologies to build DTs of AM systems. There has been little DT work focused on a unified approach to handle different types of data available through the machine, control system and the design data of an AM process to model the cyber-physical nature of an AM machine for anomaly detection and performance monitoring.

This section presents a specialization of the DT architecture presented in Chapter VI for performance monitoring and anomaly detection of AM processes by utilizing spatiotemporal data streams. Focused examples of the presented approach on FDM technology are presented to illustrate practical use cases of the proposed DT.

We utilize STL to define propositions to be monitored by the DTs, similar to the presented methodology in Section 6.4.2.2. Propositions for an AM process may specify properties of the end-product. Additionally, a proposition may define allowable working conditions of the AM machine or the materials in the process. For example, an AM process

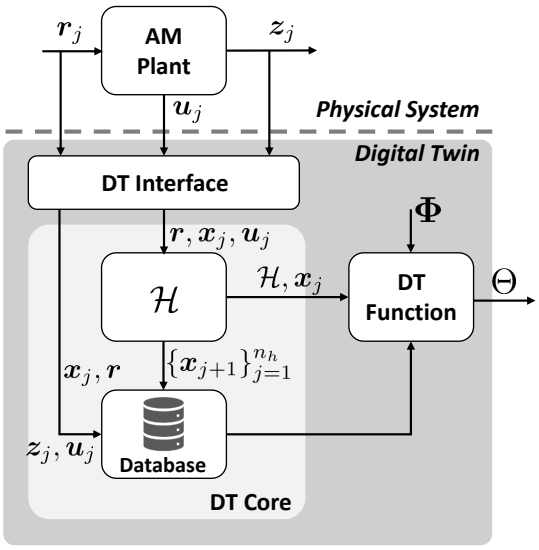


Figure C.1: Structure of the DT application for spatiotemporal monitoring.

proposition π_j may define allowable printing temperatures specific to each material in an AM process. To check if the proposition is satisfied, a measurement signal $\boldsymbol{s}(t)$ of the corresponding material temperature may be monitored in addition to other measurement data.

C.2 Digital Twin Framework Application for Spatiotemporal AM Monitoring

Figure C.1 illustrates the specialized DT architecture for spatiotemporal AM process monitoring. Notice that the components of Fig. C.1 are similar to the general framework presented in Fig. 6.2 and Fig. 6.5. Previously, we presented a system-level framework where multiple DTs were interacting to deliver results. Here we specialize to the case when we have a single purpose-driven DT implemented for an AM process for illustrative purposes of how the framework may be utilized in several practical applications. Three main components of the proposed DT are the *DT Interface*, *DT Functions*, and *DT Core*. The reference list $\boldsymbol{r} = [\boldsymbol{r_1}, \ldots, \boldsymbol{r}_{n_f}]$ is the list of reference inputs for the AM plant. An example of $\boldsymbol{r}$ is GCode instructions, used to prescribe actions for the components of a numerically controlled machine, which are commonly used in AM. Each line of a GCode file describes a set of actions for the machine to execute using its actuators. Consequently, $\boldsymbol{r}_j$ denotes a single line of a GCode command executed at time-step j.

C.2.1 DT Interface for AM Processes

Real-time data coming from the physical system is managed by the DT Interface. A conceptual AM plant is shown in Fig. C.2. An AM plant in this work is considered as a closed-loop controlled AM process in which the only allowable input to the closed-loop AM plant is the reference list $\boldsymbol{r}$. This consideration is a practical one since most AM machines in practice have OEM control systems and sensors that are not accessible to the user during the AM process. The reference list input (GCode) is analyzed by an interpreter and a controller input $\hat{\boldsymbol{r}}_j$ is generated for the OEM controller of the AM plant. The OEM controller uses the controller input and the measurements from the AM process coming from the OEM sensors to generate the actuator inputs $\boldsymbol{u}_j$. Based on the actuator inputs, the AM process takes place and a physical part is manufactured. The physical outputs of the AM process are denoted with $\boldsymbol{y}_j$ in Fig. C.2. The OEM sensors measure the physical output $\boldsymbol{y}_j$ and send $\boldsymbol{z}_j^{OEM}$ back to the OEM controller. External sensors such as cameras, laser scanners, and temperature readers are often instrumented on an AM machine to measure the physical output $\boldsymbol{y}_j$ [46, 160]. External output measurements are shown with $\boldsymbol{z}_j^{ext}$ in Fig. C.2. Therefore, $\boldsymbol{z}_j^{ext}$ denotes the sensory measurement signal for the process output $\boldsymbol{y}_j$ The output data $\boldsymbol{z}_j = [\boldsymbol{z}_j^{ext}, \boldsymbol{z}_j^{OEM}]^T$ is computed as a vector output of measurements from OEM sensors and external sensors, at the output of the AM Plant. In practice, some of the OEM sensor measurements may be unobservable to the user, in which case $\boldsymbol{z}_j^{OEM}$ represents a partial data stream from the OEM sensor measurements.

The output data $\boldsymbol{z}_j$ can be collected in real-time, during the printing of a single layer (temperature measurement) and in between layers (layer-to-layer metrology). The presented framework can accommodate measurements with different time scales. For consistency of presentation, this work will focus on real-time data collected continuously during the printing of layers. As an illustrative example, FDM machines have heating actuators for the extruders and the heated print bed. A GCode command at time-step j ($\boldsymbol{r}_j$) is interpreted and a reference temperature for an extruder ($\hat{\boldsymbol{r}}_j$) is sent to the OEM controller. The OEM controller has a control algorithm that takes the control input and the measurement $\boldsymbol{z}_j^{OEM}$ from the OEM sensors to compute an appropriate actuator input $\boldsymbol{u}_j$ which, in turn changes the temperature of the extruder ($\boldsymbol{y}_j$). External sensors can be used to measure this temperature change ($\boldsymbol{z}_j^{ext}$).

The output data $\boldsymbol{z}_j$ of an AM Plant is transferred through a predefined transmission protocol ($e.g.$ TCP/IP) in real-time (at a sampling rate) to the DT Interface. In theory, the reference list $\boldsymbol{r}$, the current-time (j) reference to the AM plant $\boldsymbol{r}_j$, and the inputs for the actuators of the AM plant $\boldsymbol{u}_j$ are communicated through the DT Interface. However, in practice, the continuous states of the machine (e.g. position and extruder temperature),

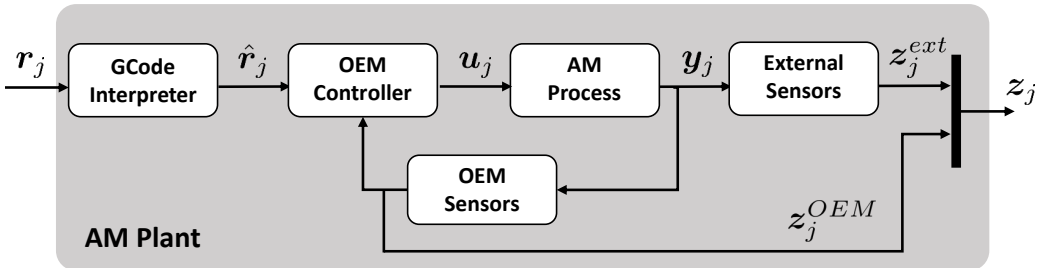


Figure C.2: Block diagram for an AM plant using GCode reference list inputs.

controller input $\hat{\boldsymbol{r}}_j$, and the actuator inputs $\boldsymbol{u}_j$ are generally not observable from the AM plant. For this purpose, pre-processing, filtering, state estimation, and event detection on the original data streams $\boldsymbol{z}_j, \boldsymbol{u}_j$, and $\boldsymbol{r}$ (based on the availability on a specific AM plant) are implemented in the DT interface, to generate useful data for the models in the DT Core. An example of the different data streams and their use in a DT is shown in the case study.

Streaming data are sampled at a sampling rate τ and the data at time-step j are prepared as a state $\boldsymbol{x}_j = [\boldsymbol{q}_j, \boldsymbol{x}_j^r]^T$, where the real-valued continuous states are denoted with $\boldsymbol{x}_j^r \in \mathbb{R}^{n_r}$, and the discrete states denoted with $\boldsymbol{q}_j \in \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{n_d}\}$. State estimation and event detection on the data streams from the AM plant are used for evaluating $\boldsymbol{x}_j$. The data streams from the AM plant and the state $\boldsymbol{x}_j$ are shared with a database at the sampling rate τ.

C.2.2 DT Core for AM Processes

The most important purpose of a DT is to utilize models of the physical system with real-time updated state information to provide KPIs. For this purpose, an AM hybrid automaton (AM-HA) model that captures both the continuous and discrete-event dynamics of the AM process is included in the DT core. A hybrid system model for the continuous and discrete-event dynamics of a micro-AM deposition system is proposed in [177]. Here, we propose a general-purpose hybrid automaton for DT development with various AM-processes. Definition of the AM-HA extends the definition of the functional state model in Chapter V [14].

Definition 3.1. [AM-Hybrid Automaton] An AM-HA is a tuple $\mathcal{H} = (Q, X, U, \Sigma, f, G, R, Init)$ where:

- $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{n_d}\}$ is the set of discrete states
- $X \subseteq \mathbb{R}^{n_r}$ is the space of real-valued states

- $U \subseteq \mathbb{R}^{n_u}$ is the space of admissible actuator inputs
- $\Sigma \subseteq Q \times Q$ is the set of discrete transition events (edges)
- $f : Q \times X \times U \rightarrow \mathbb{R}^{n_r}$ is a vector field for the discrete time dynamics of the system, such that $\boldsymbol{x}^r_{j+1} = f(\boldsymbol{q}_i, \boldsymbol{x}^r_j, \boldsymbol{u}_j), \boldsymbol{q}_i \in Q, \boldsymbol{x}^r_j \in X, \boldsymbol{u}_j \in U$
- $G : \Sigma \rightarrow 2^X$ is a set of guards
- $L : \Sigma \times X \rightarrow 2^X$ is a reset map for the continuous states
- $Init = \{(\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{x}}) |\ \tilde{\boldsymbol{q}} \in Q, \tilde{\boldsymbol{x}} \in X\}$ is the initial state.

The hybrid state of $\mathcal{H}$ at time instant j is given as $\boldsymbol{x}_j \in Q \times X$. Streaming data (with the estimated and observed states) from the DT Interface is pushed to the DT Core and the state $\boldsymbol{x}_j$ is updated in each time-step j to track the discrete and continuous states of the system. The hybrid states of $\mathcal{H}$ are assumed to be observable. By updating the hybrid state of the system in real-time, transitions of the model ($e_j \in \Sigma$) are also tracked. If some events are observable through the output $\boldsymbol{z}_j$, the events are used for updating the hybrid state as well. The hybrid system $\mathcal{H}$, is encoded in a predefined format ($e.g.$ XML, JSON). The $\mathcal{H}$ is shared with the DT Function and the up-to-date state $\boldsymbol{x}_j$ is shared every time-step with the DT Function (Fig. C.1).

The AM-HA in the DT Core has the capability to predict the future state progression $\{\boldsymbol{x}_{j+1}\}_{j=1}^{n_h}$ for a horizon of length n_h based on the current state, given that $\boldsymbol{r}_j$ and $\boldsymbol{u}_j$ are provided for the prediction horizon. The prediction task for a given horizon is done by evaluating traces of $\mathcal{H}$, with the initial condition as $Init = \boldsymbol{x}_j$ and the actuator inputs $\boldsymbol{u}_j$ derived from the reference list $\boldsymbol{r}$. A detailed analysis on the simulation of the AM-HA $\mathcal{H}$ is subject of future work.

DT Core includes a database to store the evolution of state trajectories $\boldsymbol{x}_j$ as well as the data in the AM-workflow. The reference list $\boldsymbol{r}$ of the AM process is also stored in the database. This way, DT Function may request a real time data stream and the reference data from the DT Core.

C.2.3 DT Function for AM processes

DT Function makes use of the data from the DT Core to perform various tasks including performance analysis and anomaly detection. DT Function allows various applications to be integrated with the DT, given that proper data types are defined. Based on the available data streams inside the DT, a generic DT Function g has the following data streams available.

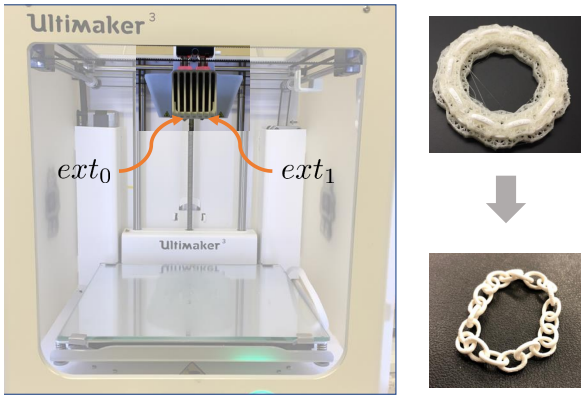


Figure C.3: The Ultimaker 3 used in the case study (left). Printer part using dual extrusion FDM (right top) and the resulting chain geometry with the support material removed (right bottom).

- $\mathbf{\Phi} = \bigwedge_{\forall \phi_j \in \Phi} \phi_j$: a conjunction of n_s propositions for the AM process
- $\bar{\boldsymbol{x}} = \{\boldsymbol{x}_i\}_{i=t_0}^{t_0+j+n_h}$: sequence of hybrid states starting from initial (initialization of the DT) time-step t_0 up to current time-step j and the prediction horizon n_h
- $\bar{\boldsymbol{z}}, \bar{\boldsymbol{u}}$: sequences of measured AM process outputs and actuator inputs between time-steps $[t_0, j]$

Utilizing the available input data streams, DT function outputs Θ (see Fig. C.1). Depending on the specific application, the structure of the output Θ may differ. Two illustrative DT Functions are discussed in the case study. A DT Function g may use all the available data streams although it is not required to do so. Next, we present several DT applications that implement different functions in the DT function block.

C.3 Experimental Demonstrations for Spatiotemporal Analysis

In this section we present several experimental results with the DT framework on an FDM process. An Ultimaker 3 printer is used, shown in Fig. C.3. The printer has two extruders. Left extruder in the picture, ext_0, extrudes the structural material (PLA), while the right extruder, ext_1, extrudes the support material (PVA). A 3D chain geometry is printed for part of the experimental results that utilize two extruders, which requires the use of both structural and support material during the process (Fig. C.3). Process data is collected from the Jedi API of Ultimaker[1]. A real-time data collection pipeline is set up

[1] http://software.ultimaker.com/jedi/api/

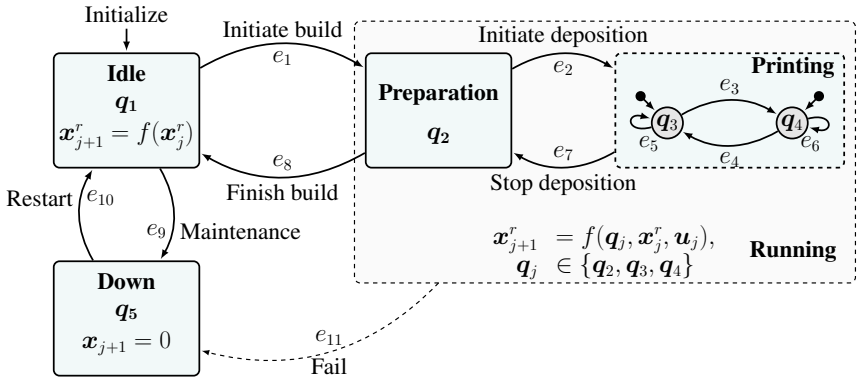


Figure C.4: The AM-HA model for dual extruder printer used in the case study.

using ADEPT[2] framework of Applied Dynamics International (ADI). A data server is set up to collect data during the print via the API at a fixed sampling rate of $\tau = 200\ ms$. The sampling rate is set based on the rate of data availability through the Ultimaker API. The ADI data collection framework is used for collecting material usage and temperature data about the two extruders shown in Fig. C.3.

The AM-HA for the experimental setup is shown in Fig. C.4. The system is initialized as idle ($\boldsymbol{q}_1$), and as real-time data is streamed through the DT Interface, the hybrid state $\boldsymbol{x}_j$ of the hybrid automaton is updated. The super-state ***Printing*** has two sub-states that model the use of two extruders shown in Fig. C.3. The model shown in Fig. C.4 has eleven transitions $\Sigma = \{e_1, \ldots, e_{11}\}$ and five discrete states $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_5\}$.

For an FDM process, let $M = \{m_1, \ldots, m_n\}$ be the set of materials (m_i) available for a specific machine. In addition, define $T^i(t) \in \mathbb{R}$ as the temperature reading at the i^{th} extruder, where $t \in \mathbb{R}$ is the time argument. Discrete time readings of the temperature are denoted with subscripts $e.g.\ T^i_j, t \in j\tau, j = 0, 1, \ldots$. Then the continuous states for the temperature of two extruders are given as $\boldsymbol{x}^r_j = [T^0_j\ T^1_j]^T$, and the inputs to heating actuators of two extruders are given as $\boldsymbol{u}_j = [u^0_j\ u^1_j]^T$. The heating dynamics of the FDM machine are given by $\boldsymbol{x}_{j+1} = f(\boldsymbol{q}_j, \boldsymbol{x}^r_j, \boldsymbol{u}_j)$, where $\boldsymbol{q}_j \in Q$. Since the temperature readings of the two extruders are directly measurable using the data collection framework, the heating dynamics of the FDM machine can be computed, meaning that we can directly evaluate the states $\boldsymbol{x}^r_j$ and the heating input $\boldsymbol{u}_j$ of the system from the output $\boldsymbol{z}_j$. The state and input measurements are used in this case study for KPI and STL monitoring.

There is no heating input in the **Idle** state ($\boldsymbol{q}_1$), thus the dynamics of $\boldsymbol{q}_1$ is the autonomous cooling dynamics with $\boldsymbol{u}_j = 0$. Similarly, the system has no dynamics in the **Down** state ($\boldsymbol{q}_5$).

[2]https://www.adi.com/products/adept-framework/

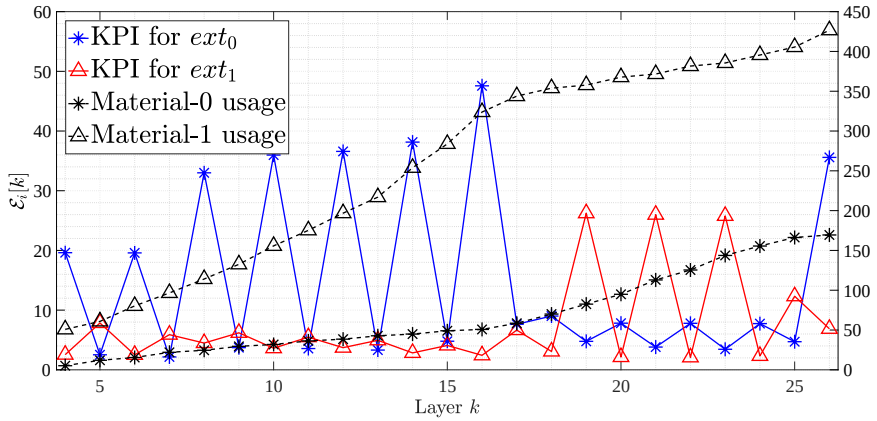


Figure C.5: Star markers show the KPI (on the left) and cumulative material usage (on the right) for the ext_0. Triangle markers show the KPI (on the left) and cumulative material usage (on the right) for the ext_1.

C.3.1 Performance Monitoring

The AM process is subject to exogenous disturbances, such as material impurities, noise in the environment, and mechanical wear of the AM plant and actuators. To understand the performance of the AM process, certain key performance indicators (KPIs) must be devised. By evaluating KPIs in real-time and between different runs of an AM plant, the performance of the AM plant is analyzed.

A KPI may be evaluated during a layer deposition, per layer, or per process to keep track of historical performance of an AM process. A DT Function evaluates the KPIs of the AM process and reports the progression of a certain KPI in the output Θ.

A novel KPI to track the energy efficiency of the AM plant is proposed for the experimental study. Define $\boldsymbol{\mu}^i[k] = [u^i(t_0), u^i(t_0+\tau), \ldots, u^i(t_f)]^T$ as the vector of actuator input sequence in $t \in [t_0, t_f]$ for the i^{th} extruder at layer k. Additionally, define $\ell_i(m_j, k)$ as the total length of material m_j used by the i^{th} extruder at layer k. Then the energy efficiency KPI for extruder i at layer k is defined as $\mathcal{E}_i[k] = ||\boldsymbol{\mu}^i[k]||_1/\ell_i(m_j, k)$.

The majority of the energy consumption in an FDM process is due to the heating of materials. The KPI $\mathcal{E}_i[k]$ measures how much energy is consumed for heating versus the amount of actual material extruded for the printing process.

Fig. C.5 shows the energy efficiency KPI $\mathcal{E}_i[k]$ for the layers $k \in [4, 26]$ in the experiment. The signal $\boldsymbol{\mu}^i[k]$ is calculated using the heater input signal monitored using the ADI data extraction framework. The heater input signal is normalized to be in the range of $[0, 1]$ and divided by the material use of the certain extruder in a layer to compute the KPI $\mathcal{E}_i[k]$ for each layer, for both extruders.

From the analysis of results shown in Fig. C.5, it is concluded that the fluctuation of $\mathcal{E}_i[k]$ between layers is largely affected by the changing lengths of material used in each layer and the energy consumed for reheating of materials between material changes. Thus, using the same material for longer in a single layer is more efficient since less re-heating energy is consumed per length of printed material in a layer. The value of the $\mathcal{E}_i[k]$ for both extruders are presented to a user through the output of the DT. By tracking this KPI between different runs of the same AM plant, it is possible to get insight on degradation in the heating system. Degradation will lead into lower efficiency, which results in an increasing trend of $\mathcal{E}_i[k]$ between different runs of the AM plant.

C.3.2 Formal Logic-Based Anomaly Detection

Though it is possible to monitor the output of the AM plant $\boldsymbol{z}$, functional dynamics information is not apparent from the data streams in $\boldsymbol{z}$. For this purpose, the traces of the states of AM-HA $\mathcal{H}$ are monitored by the DT Function ($\boldsymbol{s}(t)$ is taken as $\bar{\boldsymbol{x}}$) to check the satisfaction of STL formulas given in $\boldsymbol{\Phi}$. Examples of simple STL specifications, such as allowable working temperature ranges for specific material selections such as polylactic acid (PLA) and (polyvinyl alcohol) PVA for an FDM process, are given below. The bounds on the STL properties can be set according to different physical phenomena such as a desired viscosity of a material in a certain temperature range, or the maximum temperature that a certain extruder system allows. The STL properties used in this study are defined based on the preferred working ranges of temperatures for the structural and support materials m_1 and m_2 respectively and not as a result of an additional abnormality detection process as given in Chapter VI. Additional STL properties are defined for the performance of the heating actuators on the two extruders as the following.

$$
\begin{aligned}
\phi_1 &= \square_{[0,\beta]} \left[\boldsymbol{q}_3 \rightarrow |\Delta T^0| \leq \alpha_1(m_1)\right] \\
\phi_2 &= \square[\neg \boldsymbol{q}_3 \, \boldsymbol{\mathcal{U}}_{[0,\beta]} \boldsymbol{q}_3 \rightarrow \lozenge_{[0,\tau_s^1]} \left(\square_{[0,\tau_s^2]} |\Delta T^0| \leq \alpha_2(m_1)\right)] \\
\phi_3 &= \square_{[0,\beta]} \left[\boldsymbol{q}_4 \rightarrow |\Delta T^1| \leq \alpha_3(m_2)\right] \\
\phi_4 &= \square[\neg \boldsymbol{q}_4 \, \boldsymbol{\mathcal{U}}_{[0,\beta]} \boldsymbol{q}_4 \rightarrow \lozenge_{[0,\tau_s^1]} \left(\square_{[0,\tau_s^2]} |\Delta T^1| \leq \alpha_4(m_2)\right)]
\end{aligned}
$$

where $\Delta T^k = T^p(m_i) - T_j^k$ is the temperature error for extruder k at time j, $T^p(m_i)$ is the printing temperature for the material m_i, $\tau_s^1 = 10$ is the settling time, $\tau_s^2 = 15$ is the steady-state time, $|\cdot|$ is the L_1-norm (absolute value), m_1, m_2 are the materials used in extruders 0 and 1 respectively, and $\alpha_i(\cdot)$ are material dependent bounds for the satisfactory execution of the FDM process. The printing temperature for the structural material is $T^p(m_1) = 205°$C,

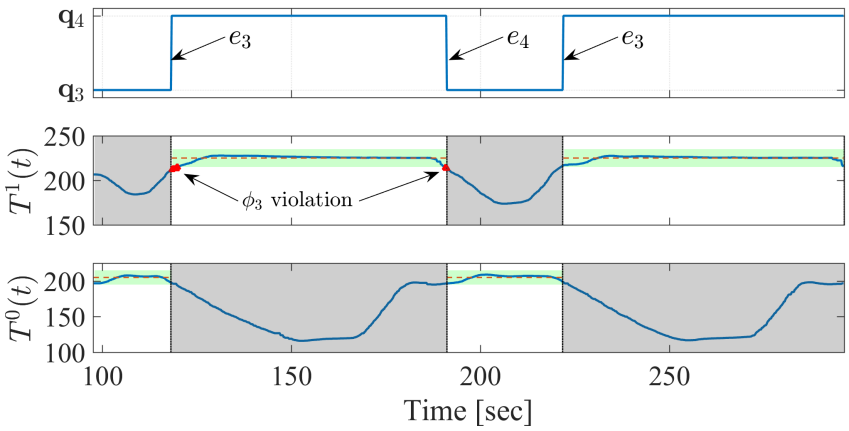


Figure C.6: *Top:* Discrete states and transitions of the hybrid system given in Fig. C.4. *Middle/Bottom:* The temperature of ext_1/ext_0 versus time with inactive periods grayed-out and printing temperatures are shown with red dashed lines. Temperature bounds are shown with green fills. Violation of ϕ_3 is shown with the red triangle markers.

and the printing temperature for the support material is $T^p(m_1) = 225°\text{C}$.

The property ϕ_1 reads as; whenever the extruder 0 is active, the L_1 norm between the temperature reading and the printing temperature should be always bounded by $\alpha_1(m_1)$. The bound on the structural material (PLA) is set as $10°\text{C}$, thus $\alpha_1(m_1) = 10°\text{C}$. The property ϕ_3 defines the same bound for the support material as $\alpha_3(m_2) = 10°\text{C}$.

The property ϕ_2 describes that whenever the extruder 0 is switched from inactive to active in the time interval $[0, \beta]$, the temperature of the extruder should eventually reach the bound $\alpha_2(m_1)$ within τ_s^1 seconds and stay within the bound for τ_s^2 seconds. This property is similar to the rise-time and settling time of a dynamic system. The bound $\alpha_2(m_1)$ is defined based on the 2% bound around the printing temperature, thus $\alpha_2(m_1) = 4.1°\text{C}$. Similarly, we have $\alpha_4(m_2) = 4.5°\text{C}$. The temperature evolution of the FDM is measured during the case study and the satisfaction of the conjunction of all the STL properties are evaluated for the measured signal.

The proposed DT framework (Fig. C.1) with the hybrid model is required to monitor the data with the discrete event and continuous states to detect anomalies. STL properties with the hybrid model define a formal infrastructure between the temporal arguments, discrete states, and continuous states to perform an efficient anomaly detection. Various STL specifications may be devised for specific needs in an anomaly detection application, utilizing the DT framework proposed in this work.

Figure C.6 shows the signals $T^i(t), i = 1, 2$ from the data collected during the experiments, and the evolution of discrete states $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$ with respect to time. A pre-

processing step is implemented in the DT Interface to detect the times when each extruder is active. The events that result in transition between discrete states, temperature readings, and bounds are shown on the top plot of Fig. C.6. As shown in Fig. C.6, the measured signals are analyzed in the time intervals on which each extruder is active based on the STL specifications. Bounds shown in green are set by α_2 and α_4, and the printing temperatures are shown with red dashed lines.

The intervals on times $[118.4, 120]$ and $[190.8, 191.2]$ violate the proposition ϕ_3, as shown in Fig. C.6. The violation is caused by the temperature of extruder 1 being outside of the bounds set by $\alpha_3(m_2)$. Note that the bounds set by $\alpha_3(m_2)$ are violated multiple times through the experiment, but since the STL properties define the time intervals in which a property must be satisfied with respect to the states of the hybrid model, an anomaly is accurately detected. The detected anomaly is reported to the user through the output Θ of the DT.

C.3.3 Functional State DTs for AM Processes

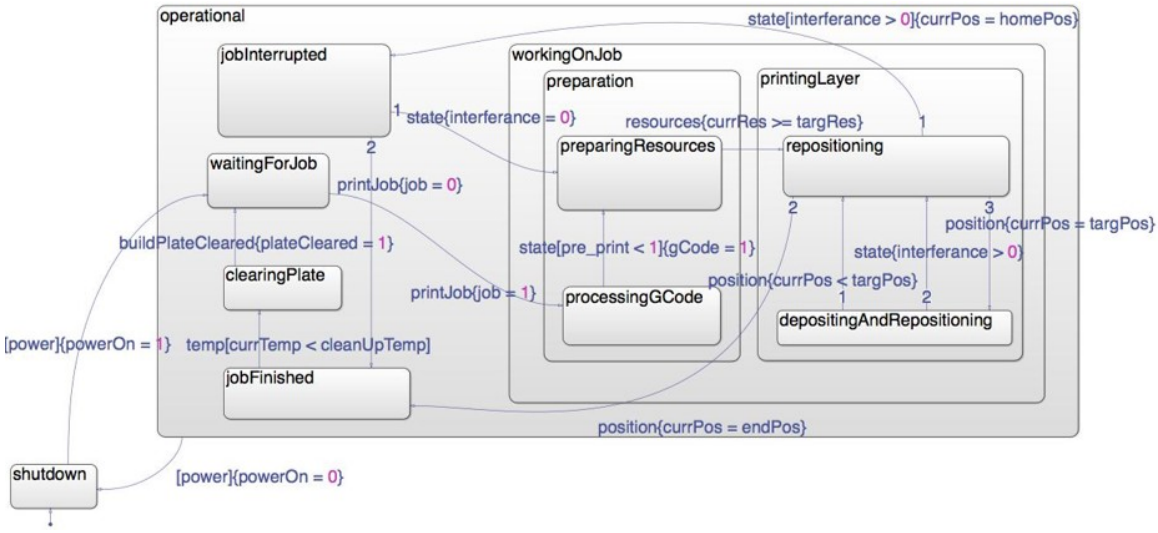


Figure C.7: The functional state finite state machine model of the 3D printer built with Simulink.

The efficiency KPI and temperature errors of each extruder are evaluated within the ADEPT framework and illustrated through a visualization interface. The functional state DT, given in Fig. C.7 is implemented based on the AM-HA model given in Fig. C.4. The process has shutdown, running, and workingOnJob states, which have substates to indicate the arrival of a job, preparation of resources, GCode preprocessing by the machine,

repositioning of axes, and the printing process itself. After a printing job is finished, the machine state is set to wait for the build plate to be cleared and eventually returning to waiting for a job. The model is implemented in the Simulink environment. The functional state DT has great importance for scheduling and dispatch applications in an AM fleet where a decision-maker utilizes the real-time availability of the machines to schedule jobs.

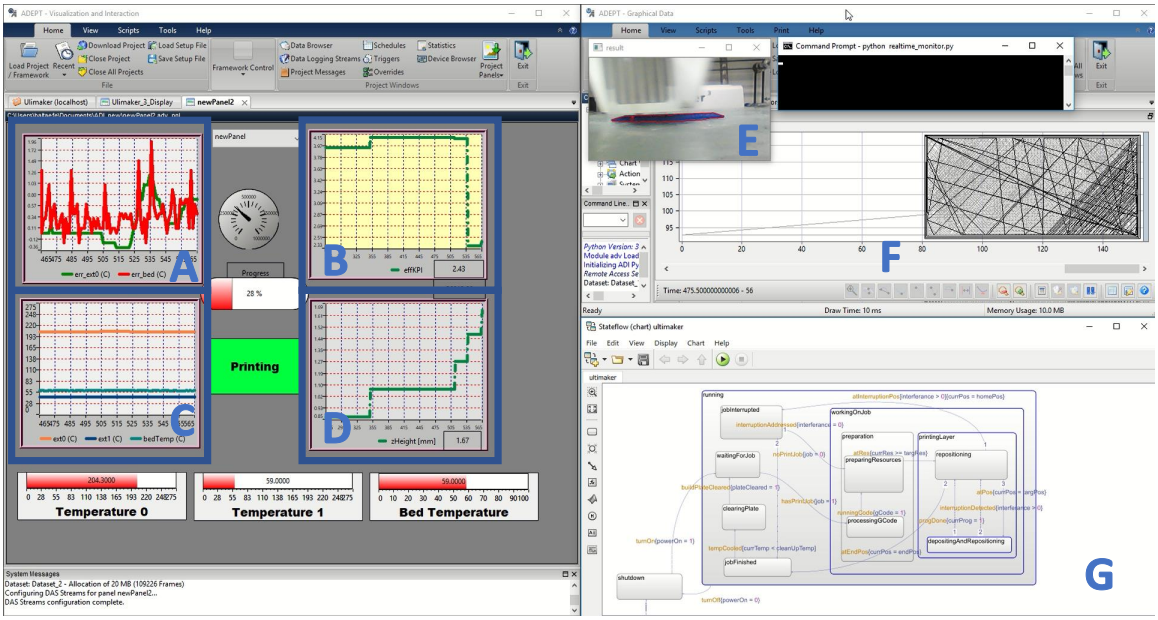


Figure C.8: Screen-shot of the implemented DTs working in real-time during a printing process. Panel A: temperature error. Panel B: efficiency metric. Panel C: tempreatures for the two extruders and the build bed. Panel D: current height. Panel E: computer vision output for nozzle clog detection. Panel F: travel of the extruder head form GCode information. Panel G: functional states of the 3D printer.

A screen-shot from the real-time implementation of the DTs for AM is shown on Fig. C.8. Panel A shows the temperature error on the extruders and the build bed. Panel B evaluates the efficiency KPI. Panel C shows the temperatures for the two extruders and the build bed. Panel D shows the current height of the print process. Additional information for the data collection rate, job completion progress and high-level functional state for the process are also shown on the left. Panel E shows the computer vision output used for detecting nozzle clogs via computer vision and additional sensor information on material feed to the extruders. Panel F shows the travel of the extruder head, extracted through the web-API (this information is based on the GCode data rather than a position encoder on the motion axes.) Panel G shows the AM-HA model of the process, illustrating the functional

states of the 3D printer. This implementation presents how the proposed DT framework for AM processes may be utilized to capture multiple DT applications and data streams to provide a comprehensive overview of the spatiotemporal attributes of the AM process.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Daekeon Ahn, Jin Hwe Kweon, Soonman Kwon, Jungil Song, and Seokhee Lee. Representation of surface roughness in fused deposition modeling. *Journal of Materials Processing Technology*, 209(15-16):5593–5600, 2009.

[2] Doruk Aksoy, Efe C. Balta, Dawn M. Tilbury, and Kira Barton. A control-oriented model for bead cross-sectional geometry in fused deposition modeling. In *American Control Conference (ACC)*, 2020.

[3] Berk Altin and Kira Barton. Exponential stability of nonlinear differential repetitive processes with applications to iterative learning control. *Automatica*, 81:369–376, 2017.

[4] Berk Altin, Zhi Wang, David J Hoelzle, and Kira Barton. Robust monotonically convergent spatial iterative learning control: Interval systems analysis via discrete fourier transform. *IEEE Transactions on Control Systems Technology*, 2018.

[5] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[6] David Angeli and Eduardo D Sontag. Monotone control systems. *IEEE Transactions on automatic control*, 48(10):1684–1698, 2003.

[7] Gilles Audemard, Alessandro Cimatti, Artur Kornilowicz, and Roberto Sebastiani. Bounded model checking for timed systems. In *International Conference on Formal Techniques for Networked and Distributed Systems*, pages 243–259. Springer, 2002.

[8] Efe C. Balta, Kshitij Jain, Yikai Lin, Dawn Tilbury, Kira Barton, and Z. Morley Mao. Production as a service: A centralized framework for small batch manufacturing. In *Automation Science and Engineering (CASE), 2017 13th IEEE Conference on*, pages 382–389. IEEE, 2017.

[9] Efe C. Balta, Ilya Kovalenko, Isaac A. Spiegel, Dawn M. Tilbury, and Kira Barton. Details of the First-Order Logic Representation in "Model Predictive Control of Priced Timed Automata Encoded with First-Order Logic". Technical report, University of Michigan Barton Research Group, 2020.

[10] Efe C. Balta, Ilya Kovalenko, Isaac A. Spiegel, Dawn M. Tilbury, and Kira Barton. Model predictive control of priced timed automata encoded with first-order logic. *IEEE Transactions on Control Systems Technology*, 2021.

[11] Efe C. Balta, Yikai Lin, Kira Barton, Dawn M. Tilbury, and Z. Morley Mao. Production as a service: A digital manufacturing framework for optimizing utilization. *IEEE Transactions on Automation Science and Engineering*, 15(4):1483–1493, Oct 2018.

[12] Efe C. Balta, Michael Pease, James Moyne, Kira Barton, and Dawn M. Tilbury. Cyber-attack detection digital twins for cyber-physical manufacturing systems. Under review, 2020.

[13] Efe C. Balta, Dawn M. Tilbury, and K. Barton. Control-oriented modeling and layer-to-layer stability for fused deposition modeling: A kernel basis approach. In *2019 American Control Conference (ACC)*, pages 4727–4733, July 2019.

[14] Efe C. Balta, Dawn M. Tilbury, and Kira Barton. A centralized framework for system-level control and management of additive manufacturing fleets. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1071–1078, Aug 2018.

[15] Efe C. Balta, Dawn M. Tilbury, and Kira Barton. A digital twin framework for performance monitoring and anomaly detection in fused deposition modeling. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 823–829, Aug 2019.

[16] Efe C. Balta, Dawn M. Tilbury, and Kira Barton. Layer-to-layer stability of linear layerwise spatially varying systems: Applications in fused deposition modeling. *IEEE Transactions on Control Systems Technology*, 2021.

[17] Goran Banjac, Paul Goulart, Bartolomeo Stellato, and Stephen Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *Journal of Optimization Theory and Applications*, 183(2):490–519, 2019.

[18] Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi. Monitoring mobile and spatially distributed cyber-physical systems. In *Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pages 146–155. ACM, 2017.

[19] Ezio Bartocci and Pietro Lió. Computational modeling, formal analysis, and tools for systems biology. *PLoS computational biology*, 12(1):e1004591, 2016.

[20] Kira Barton, Douglas A Bristow, David Hoelzle, and Sandipan Mishra. Mechatronics advances for the next generation of am process control. 2019.

[21] Kira L Barton and Andrew G Alleyne. A cross-coupled iterative learning control design for precision motion control. *IEEE Transactions on Control Systems Technology*, 16(6):1218–1231, 2008.

[22] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[23] Gerd Behrmann. UPPAAL CORA: UPPAAL for Planning and Scheduling, 2014.

[24] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. Developing UPPAAL over 15 years. *Software: practice and experience*, 41(2):133–142, 2011.

[25] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced time automata. In *International Workshop on Hybrid Systems: Computation and Control*, pages 147–161. Springer, 2001.

[26] Gerd Behrmann, Kim G Larsen, and Jacob I Rasmussen. Priced timed automata: Algorithms and applications. In *International Symposium on Formal Methods for Components and Objects*, pages 162–182. Springer, 2004.

[27] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. dr0wned–cyber-physical attack with additive manufacturing. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[28] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[29] Julian Besag, Jeremy York, and Annie Mollié. Bayesian image restoration, with two applications in spatial statistics. *Annals of the institute of statistical mathematics*, 43(1):1–20, 1991.

[30] Devendra Bhave, Shankara Narayanan Krishna, and Ashutosh Trivedi. On nonlinear prices in timed automata. In *Electronic Proceedings in Theoretical Computer Science*, volume 232, pages 65–78, 2016.

[31] Nikolaj Bjørner and Anh-Dung Phan. νz-maximal satisfaction with z3. In *6th International Symposium on Symbolc Computation in Software Science (SCSS)*, pages 1–9, 2014.

[32] Alexander Bleakie and Dragan Djurdjanovic. Feature extraction, condition monitoring, and fault modeling in semiconductor manufacturing systems. *Computers in Industry*, 64(3):203–213, 2013.

[33] Luca Bortolussi and Laura Nenzi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, pages 66–73. ICST (Institute for Computer Sciences, Social-Informatics and , 2014.

[34] Radu Ioan Boţ and Ernö Robert Csetnek. Forward-backward and tsengs type penalty schemes for monotone inclusion problems. *Set-Valued and Variational Analysis*, 22(2):313–331, 2014.

[35] Michael S Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on automatic control*, 43(4):475–482, 1998.

[36] Alessandro Busachi, John Erkoyuncu, Paul Colegrove, Filomeno Martina, Chris Watts, and Richard Drake. A review of additive manufacturing technology and cost estimation techniques for the defence sector. *CIRP Journal of Manufacturing Science and Technology*, 19:117–128, 2017.

[37] Defense Use Case. Analysis of the cyber a nist sp 800-28 version 2 - nist pagettack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388, 2016.

[38] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer US, 2008.

[39] Stephanie E Chang, Masanobu Shinozuka, and James E Moore. Probabilistic earthquake scenarios: extending risk analysis methodologies to spatially distributed systems. *Earthquake Spectra*, 16(3):557–572, 2000.

[40] George HG Chen and R Tyrrell Rockafellar. Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997.

[41] Yiyang Chen, Bing Chu, and Christopher T Freeman. Generalized iterative learning control using successive projection: Algorithm, convergence, and experimental verification. *IEEE Transactions on Control Systems Technology*, 2019.

[42] Longwei Cheng, Fugee Tsung, and Andi Wang. A statistical transfer learning perspective for modeling shape deviations in additive manufacturing. *IEEE Robotics and Automation Letters*, 2(4):1988–1993, 2017.

[43] Yuan Cheng and Mohsen A Jafari. Vision-based online process control in manufacturing applications. *IEEE Transactions on Automation Science and Engineering*, 5(1):140–153, 2008.

[44] Sujit Rokka Chhetri and Mohammad Abdullah Al Faruque. Side channels of cyber-physical systems: Case study in additive manufacturing. *IEEE Design & Test*, 34(4):18–25, 2017.

[45] Sujit Rokka Chhetri, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Kcad: kinetic cyber-attack detection method for cyber-physical additive manufacturing systems. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, pages 1–8. IEEE, 2016.

[46] Sujit Rokka Chhetri, Sina Faezi, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Quilt: quality inference from living digital twins in iot-enabled manufacturing systems. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 237–248, 2019.

[47] Ronghu Chi, Zhongsheng Hou, and Jianxin Xu. Adaptive ilc for a class of discrete-time systems with iteration-varying trajectory and random initial condition. *Automatica*, 44(8):2207–2213, 2008.

[48] Young-Sik Choi. Least squares one-class support vector machine. *Pattern Recognition Letters*, 30(13):1236–1240, 2009.

[49] Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. An experimental spatio-temporal model checker. In *SEFM 2015 Collocated Workshops*, pages 297–311. Springer, 2015.

[50] Luis F Combita, Alvaro A Cardenas, and Nicanor Quijano. Mitigating sensor attacks against industrial control systems. *IEEE Access*, 7:92444–92455, 2019.

[51] Raphal Comminal, Marcin P. Serdeczny, David B. Pedersen, and Jon Spangenberg. Numerical modeling of the strand deposition flow in extrusion-based additive manufacturing. *Additive Manufacturing*, 20:68–76, 2018.

[52] JM Gomes da Silva Jr and Sophie Tarbouriech. Anti-windup design with guaranteed regions of stability for discrete-time linear systems. *Systems & Control Letters*, 55(3):184–192, 2006.

[53] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Virkumar Vazirani. *Algorithms*. McGraw-Hill Higher Education, 2008.

[54] Damek Davis and Wotao Yin. A three-operator splitting scheme and its optimization applications. *Set-valued and variational analysis*, 25(4):829–858, 2017.

[55] Jim Davis, Thomas Edgar, James Porter, John Bernaden, and Michael Sarli. Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers & Chemical Engineering*, 47:145–156, 2012.

[56] B De Schutter and TJJ Van Den Boom. MPC for discrete-event systems with soft and hard synchronization constraints. *International Journal of Control*, 76(1):82–94, 2003.

[57] Bart De Schutter and Ton Van Den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37(7):1049–1056, 2001.

[58] T. DebRoy, W. Zhang, J. Turner, and S. S. Babu. Building digital twins of 3D printing machines. *Scripta Materialia*, 135:119–124, 2017.

[59] Kun Deng, Yushan Chen, and Calin Belta. An approximate dynamic programming approach to multiagent persistent monitoring in stochastic environments with temporal logic constraints. *IEEE Transactions on Automatic Control*, 62(9):4549–4563, 2017.

[60] Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1):5–30, 2017.

[61] Stefano Di Cairano, Abraham Goldsmith, and Scott A Bortoff. Mpc and spatial governor for multistage precision manufacturing machines. *IFAC-PapersOnLine*, 48(23):398–403, 2015.

[62] Moritz Diehl, Hans Georg Bock, and Johannes P Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, 43(5):1714–1736, 2005.

[63] Peter J Diggle, Paula Moraga, Barry Rowlingson, Benjamin M Taylor, et al. Spatial and spatio-temporal log-gaussian cox processes: extending the geostatistical paradigm. *Statistical Science*, 28(4):542–563, 2013.

[64] Dimos V Dimarogonas, Emilio Frazzoli, and Karl H Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2011.

[65] Donghong Ding, Zengxi Pan, Dominic Cuiuri, Huijun Li, Stephen Van Duin, and Nathan Larkin. Bead modelling and implementation of adaptive MAT path in wire and arc additive manufacturing. *Robotics and Computer-Integrated Manufacturing*, 39:32–42, 2016.

[66] Alexandre Donzé, Thomas Ferrere, and Oded Maler. Efficient robust monitoring for STL. In *International Conference on Computer Aided Verification*, pages 264–279. Springer, 2013.

[67] Molong Duan, Deokkyun Yoon, and Chinedum E Okwudire. A limited-preview filtered b-spline approach to tracking control–with application to vibration-induced error compensation of a 3d printer. *Mechatronics*, 2017.

[68] Jonathan Eckstein and Michael C Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 10(2):218–235, 1998.

[69] Deniz Sera Ertay, Alexander Yuen, and Yusuf Altintas. Synchronized material deposition rate control with path velocity on fused filament fabrication machines. *Additive Manufacturing*, 19:205–213, 2018.

[70] Haijun Fang and Zongli Lin. Stability analysis for linear systems under state constraints. *IEEE Transactions on Automatic Control*, 49(6):950–955, 2004.

[71] Panagis Foteinopoulos, Alexios Papacharalampopoulos, and Panagiotis Stavropoulos. On thermal modeling of Additive Manufacturing processes. *CIRP Journal of Manufacturing Science and Technology*, 20:66–83, 2018.

[72] Walter J Freeman. Characterization of state transitions in spatially distributed, chaotic, nonlinear, dynamical systems in cerebral cortex. *Integrative Physiological and Behavioral Science*, 29(3):294–306, 1994.

[73] Daniel Gabay. Chapter ix applications of the method of multipliers to variational inequalities. In *Studies in mathematics and its applications*, volume 15, pages 299–331. Elsevier, 1983.

[74] Wei Gao, Yunbo Zhang, Devarajan Ramanujan, Karthik Ramani, Yong Chen, Christopher B Williams, Charlie CL Wang, Yung C Shin, Song Zhang, and Pablo D Zavattieri. The status, challenges, and future of additive manufacturing in engineering. *Computer-Aided Design*, 69:65–89, 2015.

[75] Kévin Garanger, Eric Feron, Pierre-Loïc Garoche, Julian J Rimoli, John D Berrigan, Martha Grover, and Kerianne Hobbs. Foundations of intelligent additive manufacturing. *arXiv preprint arXiv:1705.00960*, 2017.

[76] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *International Conference on Protocol Specification, Testing and Verification*, pages 3–18. Springer, 1995.

[77] Ian Gibson, David W Rosen, and Brent Stucker. *Additive Manufacturing Technologies: Rapid Prototyping to Direct Digital Manufacturing*. Springer-Verlag New York, 2009.

[78] Ebru Aydin Gol, Ezio Bartocci, and Calin Belta. A formal methods approach to pattern synthesis in reaction diffusion systems. In *53rd IEEE Conference on Decision and Control*, pages 108–113. IEEE, 2014.

[79] Vanessa Gómez-Verdejo, Jerónimo Arenas-García, Miguel Lázaro-Gredilla, and Ángel Navia-Vázquez. Adaptive one-class support vector machine. *IEEE Transactions on Signal Processing*, 59(6):2975–2981, 2011.

[80] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, pages 85–113. Springer, 2017.

[81] Yijie Guo and Sandipan Mishra. A predictive control algorithm for layer-to-layer ink-jet 3D printing. *Proceedings of the American Control Conference*, 2016-July:833–838, 2016.

[82] Yijie Guo, Joost Peters, Tom Oomen, and Sandipan Mishra. Distributed model predictive control for ink-jet 3d printing. In *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 436–441. IEEE, 2017.

[83] Yijie Guo, Joost Peters, Tom Oomen, and Sandipan Mishra. Control-oriented models for ink-jet 3D printing. *Mechatronics*, 56:211–219, 2018.

[84] Sohrab Haghighat, Stefano Di Cairano, Dmytro Konobrytskyi, and Scott Bortoff. Coordinated control of a dual-stage positioning system using constrained model predictive control. In *ASME 2014 Dynamic Systems and Control Conference*, pages V001T02A005–V001T02A005. American Society of Mechanical Engineers, 2014.

[85] Iman Haghighi, Austin Jones, Zhaodan Kong, Ezio Bartocci, Radu Gros, and Calin Belta. Spatel: a novel spatial-temporal logic and its applications to networked systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 189–198. ACM, 2015.

[86] Petter Hagqvist, Almir Heralić, Anna-Karin Christiansson, and Bengt Lennartson. Resistance based iterative learning control of additive manufacturing with wire. *Mechatronics*, 31:116–123, 2015.

[87] Darald J Hartfiel. *Nonhomogeneous matrix products*. World Scientific, 2002.

[88] John Haslett, Graham Wills, and Antony Unwin. Spideran interactive statistical tool for the analysis of spatially distributed data. *International Journal of Geographical Information System*, 4(3):285–296, 1990.

[89] Mohammad Hekmatnejad, Giulia Pedrielli, and Georgios Fainekos. Task Scheduling with Nonlinear Costs using SMT Solvers. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 183–188, Vancouver, Canada, 2019.

[90] Almir Heralić, Anna-Karin Christiansson, and Bengt Lennartson. Height control of laser metal-wire deposition based on iterative learning control and 3d scanning. *Optics and lasers in engineering*, 50(9):1230–1241, 2012.

[91] Almir Heralić, Anna Karin Christiansson, Mattias Ottosson, and Bengt Lennartson. Increased stability in laser metal wire deposition through feedback from optical measurements. *Optics and Lasers in Engineering*, 48(4):478–485, 2010.

[92] Christian Herde, Andreas Eggers, Martin Fränzle, and Tino Teige. Analysis of hybrid systems using HySAT. In *Third International Conference on Systems (icons 2008)*, pages 196–201. IEEE, 2008.

[93] David J Hoelzle, Andrew G Alleyne, and Amy J Wagoner Johnson. Basis task approach to iterative learning control with applications to micro-robotic deposition. *IEEE Transactions on Control Systems Technology*, 19(5):1138–1148, 2010.

[94] David J Hoelzle and Kira L Barton. On spatial iterative learning control via 2-D convolution: Stability analysis and computational efficiency. *IEEE Transactions on Control Systems Technology*, 24(4):1504–1512, 2016.

[95] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[96] Ling Hou and Anthony N Michel. Asymptotic stability of systems with saturation constraints. *IEEE transactions on automatic control*, 43(8):1148–1154, 1998.

[97] Yong Huang, Ming C Leu, Jyoti Mazumder, and Alkan Donmez. Additive manufacturing: current state, future potential, gaps and needs, and recommendations. *Journal of Manufacturing Science and Engineering*, 137(1):014001, 2015.

[98] Uduak Inyang-Udoh, Yijie Guo, Joost Peters, Tom Oomen, and Sandipan Mishra. Layer-to-layer predictive control of inkjet 3-d printing. *IEEE/ASME Transactions on Mechatronics*, 25(4):1783–1793, 2020.

[99] Rolf Isermann. Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, 29(1):71–85, 2005.

[100] Wayne A Jansen, Theodore Winograd, and Karen Scarfone. Guidelines on active content and mobile code: Version 2. In *National Institute of Standards and Technology (US)*, number NIST Special Publication 800-28, Version 2. National Institute of Standards and Technology (US), 2008.

[101] Daniel Jung and Christofer Sundström. A combined data-driven and model-based residual selection algorithm for fault detection and isolation. *IEEE Transactions on Control Systems Technology*, 27(2):616–630, 2017.

[102] Hyoung Seok Kang, Ju Yeon Lee, SangSu Choi, Hyun Kim, Jun Hee Park, Ji Yeon Son, Bo Hyun Kim, and Sang Do Noh. Smart manufacturing: Past research, present findings, and future directions. *International journal of precision engineering and manufacturing-green technology*, 3(1):111–128, 2016.

[103] Nikola Kasabov and Elisa Capecci. Spiking neural network methodology for modelling, classification and understanding of eeg spatio-temporal data measuring cognitive processes. *Information Sciences*, 294:565–575, 2015.

[104] Hassan K Khalil. Nonlinear systems. *Upper Saddle River*, 2002.

[105] Aftab A Khan, James R Moyne, and Dawn M Tilbury. An approach for factory-wide control utilizing virtual metrology. *IEEE Transactions on Semiconductor Manufacturing*, 20(4):364–375, 2007.

[106] G. L. Knapp, T. Mukherjee, J. S. Zuback, H. L. Wei, T. A. Palmer, A. De, and T. DebRoy. Building blocks for a digital twin of additive manufacturing. *Acta Materialia*, 135:390–399, 2017.

[107] GL Knapp, T Mukherjee, JS Zuback, HL Wei, TA Palmer, A De, and T DebRoy. Building blocks for a digital twin of additive manufacturing. *Acta Materialia*, 135:390–399, 2017.

[108] Jaromír J Koliha and Ivan Straškraba. Power bounded and exponentially bounded matrices. *Applications of Mathematics*, 44(4):289–308, 1999.

[109] Ilya Kolmanovsky and Elmer G Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical problems in engineering*, 4(4):317–367, 1998.

[110] Ilya Kovalenko, Dawn Tilbury, and Kira Barton. The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, 86:105–117, 2019.

[111] Ilya Kovalenko, Dawn Tilbury, and Kira Barton. Priced Timed Automata Models for Control of Intelligent Product Agents in Manufacturing Systems. In *15th Workshop on Discrete Event Systems (WODES)*, 2020.

[112] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[113] Denise Lam, Chris Manzie, and Malcolm Good. Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6137–6142. IEEE, 2010.

[114] Wilbur Langson, Ioannis Chryssochoos, SV Raković, and David Q Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[115] Kim Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *International Conference on Computer Aided Verification*, volume 2102, pages 493–505, 2001.

[116] Dan Li, Nagi Gebraeel, and Kamran Paynabar. Detection and differentiation of replay attack and equipment faults in scada systems. *IEEE Transactions on Automation Science and Engineering*, 2020.

[117] Yuan Li, You Fu Li, Qing Lin Wang, De Xu, and Min Tan. Measurement and defect detection of the weld bead based on online vision inspection. *IEEE Transactions on Instrumentation and Measurement*, 59(7):1841–1849, 2009.

[118] Benjamin Lindemann, Fabian Fesenmayr, Nasser Jazdi, and Michael Weyrich. Anomaly detection in discrete manufacturing using self-learning approaches. *Procedia CIRP*, 79:313–318, 2019.

[119] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks. *IEEE control systems letters*, 2019.

[120] Finn Lindgren, Havard Rue, et al. Bayesian spatial modelling with r-inla. *Journal of Statistical Software*, 63(19):1–25, 2015.

[121] Chao Liu, Pingyu Jiang, and Wenlei Jiang. Web-based digital twin modeling and remote control of cyber-physical production systems. *Robotics and Computer-Integrated Manufacturing*, 64:101956, 2020.

[122] Derong Liu and Anthony N Michel. *Dynamical systems with saturation nonlinearities: analysis and design*. Springer-Verlag, 1994.

[123] Jia Liu, Yun Bai, Prahalad Rao, Christopher B Williams, Chenang Liu, Chris Williams, and Zhenyu Kong. Layer-Wise Spatial Modeling of Porosity in Additive Manufacturing Real-time process monitoring for advanced manufacturing View

project Sensing and Data Analytics for Additive Manufacturing View project Layer-wise Spatial Modeling of Porosity in Additive M. 10, 2018.

[124] Jia Liu, Chenang Liu, Yun Bai, Prahalada Rao, Christopher B Williams, and Zhenyu Kong. Layer-wise spatial modeling of porosity in additive manufacturing. *IISE Transactions*, 51(2):109–123, 2019.

[125] Felipe Lopez, Miguel Saez, Yuru Shao, Efe C Balta, James Moyne, Z Morley Mao, Kira Barton, and Dawn Tilbury. Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms. *IEEE Robotics and Automation Letters*, 2(4):1885–1892, 2017.

[126] Felipe Lopez, Yuru Shao, Z. Morley Mao, James Moyne, Kira Barton, and Dawn Tilbury. A software-defined framework for the integrated management of smart manufacturing systems. *Manufacturing Letters*, 15:18 – 21, 2018.

[127] Lu Lu, Jian Zheng, and Sandipan Mishra. A layer-to-layer model and feedback control of ink-jet 3-D printing. *IEEE/ASME Transactions on Mechatronics*, 20(3):1056–1068, 2015.

[128] Yan Lu, Paul Witherell, Felipe Lopez, and Ibrahim Assouroko. Digital solutions for integrated and collaborative additive manufacturing. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V01BT02A033–V01BT02A033. American Society of Mechanical Engineers, 2016.

[129] Zhibo Luo and Yaoyao Zhao. A survey of finite element analysis of temperature and thermal stress fields in powder bed fusion additive manufacturing. *Additive Manufacturing*, 21:318–332, 2018.

[130] Claudio Mandolla, Antonio Messeni Petruzzelli, Gianluca Percoco, and Andrea Urbinati. Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry. *Computers in Industry*, 109:134–152, 2019.

[131] Deborah Mies, Will Marsden, and Stephen Warde. Overview of additive manufacturing informatics:a digital thread. *Integrating Materials and Manufacturing Innovation*, 5(1):114–142, 2016.

[132] Sajed Miremadi, Zhennan Fei, Knut Åkesson, and Bengt Lennartson. Symbolic Supervisory Control of Timed Discrete Event Systems. *IEEE Transactions on Control Systems Technology*, 23(2):584–597, 2015.

[133] Sandipan Mishra, Ufuk Topcu, and Masayoshi Tomizuka. Optimization-based constrained iterative learning control. *IEEE Transactions on Control Systems Technology*, 19(6):1613–1621, 2010.

[134] Omar A Mohamed, Syed H Masood, and Jahar L Bhowmik. Optimization of fused deposition modeling process parameters: a review of current research and future prospects. *Advances in Manufacturing*, 3(1):42–53, 2015.

[135] Nader Motee and Ali Jadbabaie. Optimal control of spatially distributed systems. *IEEE Transactions on Automatic Control*, 53(7):1616–1629, 2008.

[136] Simon Mounsey, Bernard Hon, and Chris Sutcliffe. Performance modelling and simulation of metal powder bed fusion production system. *CIRP Annals*, 65(1):421–424, 2016.

[137] James Moyne, Enrique Del Castillo, and Arnon M Hurwitz. *Run-to-run control in semiconductor manufacturing*. CRC press, 2000.

[138] James Moyne, Yassine Qamsane, Efe C. Balta, Ilya Kovalenko, John Faris, Kira Barton, and Dawn M. Tilbury. A requirements driven digital twin framework: Specification and opportunities. *IEEE Access*, 8:107781–107801, 2020.

[139] Jordi Mũnoz-Marí, Francesca Bovolo, Luis Gómez-Chova, Lorenzo Bruzzone, and Gustavo Camp-Valls. Semisupervised one-class support vector machines for classification of remote sensing data. *IEEE transactions on geoscience and remote sensing*, 48(8):3188–3197, 2010.

[140] Chris J Myers and TH-Y Meng. Synthesis of timed asynchronous circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(2):106–119, 1993.

[141] Brian N. Turner, Robert Strong, and Scott A. Gold. A review of melt extrusion additive manufacturing processes: I. Process design and modeling. *Rapid Prototyping Journal*, 20(3):192–204, 2014.

[142] Elisa Negri, Luca Fumagalli, and Marco Macchi. A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, 11:939–948, 2017.

[143] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. Qualitative and quantitative monitoring of spatio-temporal properties. In *Runtime Verification*, pages 21–37. Springer, 2015.

[144] Petter Nilsson, Omar Hussien, Ayca Balkan, Yuxiao Chen, Aaron D Ames, Jessy W Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2015.

[145] Anudeep Padmanabhan and Jing Zhang. Cybersecurity risks and mitigation strategies in additive manufacturing. *Progress in Additive Manufacturing*, pages 1–7, 2018.

[146] Antonio Padovano, Francesco Longo, Letizia Nicoletti, and Giovanni Mirabelli. A digital twin based service oriented application for a 4.0 knowledge navigation in the smart factory. *IFAC-PapersOnLine*, 51(11):631–636, 2018.

[147] Jörn Pagel and Frank M Schurr. Forecasting species ranges by statistical estimation of ecological niches and spatial population dynamics. *Global Ecology and Biogeography*, 21(2):293–304, 2012.

[148] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[149] Christopher Pannier, Maxwell Wu, David Hoelzle, and Kira Barton. Lpv models for jet-printed heightmap control. In *2019 American Control Conference (ACC)*, pages 5402–5407. IEEE, 2019.

[150] Christopher P Pannier, Mamadou Diagne, Isaac A Spiegel, David J Hoelzle, and Kira Barton. A dynamical model of drop spreading in electrohydrodynamic jet printing. *Journal of Manufacturing Science and Engineering*, 139(11):111008, 2017.

[151] Gabriele Pannocchia and Alberto Bemporad. Combined design of disturbance model and observer for offset-free model predictive control. *IEEE Transactions on Automatic Control*, 52(6):1048–1053, 2007.

[152] Neal Parikh and Stephen Boyd. Block splitting for distributed optimization. *Mathematical Programming Computation*, 6(1):77–102, 2014.

[153] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[154] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. Attack detection and identification in cyber-physical systems. *IEEE transactions on automatic control*, 58(11):2715–2729, 2013.

[155] J Pellegrino, T Makila, S Mcqueen, and E Taylor. Measurement science roadmap for polymer-based additive manufacturing. *NIST Adv. Manuf. Ser*, 100, 2016.

[156] Jesús Pinto-Lopera, José ST Motta, and Sadek Absi Alfaro. Real-time measurement of width and height of weld beads in gmaw processes. *Sensors*, 16(9):1500, 2016.

[157] Matthew Porter, Vikram Raghavan, Yikai Lin, Z Morley Mao, Kira Barton, and Dawn Tilbury. Production as a service: Optimizing utilization in manufacturing systems. In *ASME 2016 Dynamic Systems and Control Conference*, pages V002T21A012–V002T21A012. American Society of Mechanical Engineers, 2016.

[158] Pramudita R Prasetyanti and Jan Paul Medema. Intra-tumor heterogeneity from a cancer stem cell perspective. *Molecular cancer*, 16(1):41, 2017.

[159] Yassine Qamsane, Chien-Ying Chen, Efe Balta, Bin-Chou Kao, Sibin Mohan, Sibin Moyne, Dawn Tilbury, and Kira Barton. A unified digital twin framework for real-time monitoring and evaluation of smart manufacturing systems. In *IEEE International Conference on Automation Science and Engineering (CASE)*, 2019.

[160] Prahalad K Rao, Jia Peter Liu, David Roberson, Zhenyu James Kong, and Christopher Williams. Online real-time quality monitoring in additive manufacturing processes using heterogeneous sensors. *Journal of Manufacturing Science and Engineering*, 137(6):061007, 2015.

[161] J I Rasmussen, K G Larsen, and K Subramani. On using priced timed automata to achieve optimal scheduling. *Formal methods in system design*, 29(1):97–114, 2006.

[162] Volker Renken, Stephan Albinger, Gert Goch, Arne Neef, and Claus Emmelmann. Development of an adaptive, self-learning control concept for an additive manufacturing process. *CIRP Journal of Manufacturing Science and Technology*, 19:57–61, 2017.

[163] Andrea Riebler, Sigrunn H Sørbye, Daniel Simpson, and Håvard Rue. An intuitive bayesian spatial model for disease mapping that accounts for scaling. *Statistical methods in medical research*, 25(4):1145–1165, 2016.

[164] Steven Riley, Ken Eames, Valerie Isham, Denis Mollison, and Pieter Trapman. Five challenges for spatial epidemic models. *Epidemics*, 10:68–71, 2015.

[165] Sabino Francesco Roselli, Kristofer Bengtsson, and Knut Åkesson. SMT solvers for flexible job-shop scheduling problems: A computational analysis. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 673–678. IEEE, 2019.

[166] Roland Rosen, Georg Von Wichert, George Lo, and Kurt D Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567–572, 2015.

[167] Gian-Carlo Rota and W Strang. A note on the joint spectral radius. 1960.

[168] Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[169] M. Saez, F. Maturana, K. Barton, and D. Tilbury. Anomaly detection and productivity analysis for cyber-physical systems in manufacturing. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 23–29, Aug 2017.

[170] Patrick M Sammons, Michelle L Gegel, Douglas A Bristow, and Robert G Landers. Repetitive process control of additive manufacturing with application to laser metal deposition. *IEEE Transactions on Control Systems Technology*, (99):1–10, 2018.

[171] Leonardo Santana, Jorge Lino Alves, and Aurélio da Costa Sabino Netto. A study of parametric calibration for low cost 3D printing: Seeking improvement in dimensional quality. *Materials & Design*, 135:159–172, 2017.

[172] Jonathan E Seppala, Seung Hoon Han, Kaitlyn E Hillgartner, Chelsea S Davis, and Kalman B Migler. Weld formation during material extrusion additive manufacturing. *Soft matter*, 13(38):6761–6769, 2017.

[173] Jeff S Shamma. *Cooperative control of distributed multi-agent systems*. Wiley Online Library, 2007.

[174] Deepak Sharma, Helen Armer, and James Moyne. A comparison of data mining methods for yield modeling, chamber matching and virtual metrology applications. In *2012 SEMI Advanced Semiconductor Manufacturing Conference*, pages 231–236. IEEE, 2012.

[175] Haiyu Song, Peng Shi, Cheng-Chew Lim, Wen-An Zhang, and Li Yu. Attack and estimator design for multi-sensor systems with undetectable adversary. *Automatica*, 109:108545, 2019.

[176] Zhen Song, Antun Skuric, and Kun Ji. A recursive watermark method for hard real-time industrial control system cyber-resilience enhancement. *IEEE Transactions on Automation Science and Engineering*, 17(2):1030–1043, 2020.

[177] Isaac Spiegel, Ilya Kovalenko, David Hoelzle, Patrick M Sammons, and Kira L Barton. Hybrid modeling and identification of jetting dynamics in electrohydrodynamic jet printing. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 695–701. IEEE, 2017.

[178] Isaac A Spiegel, Patrick Sammons, David J Hoelzle, and Kira Barton. Hybrid modeling of electrohydrodynamic jet printing. *IEEE Transactions on Control Systems Technology*, 2019.

[179] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, pages 1–36, 2020.

[180] KVSM Stouffer, V Pillitteri, S Lightman, M Abrams, and A Hahn. Guide to industrial control systems (ICS) security-NIST. SP. 800-82r2. *NIST, US Department of Commerce, Gaithersburg, Maryland*, pages 1–247, 2015.

[181] Zhanghan Tang, Margreta Kuijper, Michelle S Chong, Iven Mareels, and Christopher Leckie. Linear system securitydetection and correction of adversarial sensor attacks in the noise-free case. *Automatica*, 101:53–59, 2019.

[182] Fei Tao and Qinglin Qi. New IT driven service-oriented smart manufacturing: framework and characteristics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):81–91, 2017.

[183] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, 2018.

[184] Gustavo Tapia and Alaa Elwany. A Review on Process Monitoring and Control in Metal-Based Additive Manufacturing. *Journal of Manufacturing Science and Engineering*, 136(6):060801, 2014.

[185] Dawn M Tilbury. Cyber-physical manufacturing systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:427–443, 2019.

[186] David Tilman and Peter Kareiva. *Spatial ecology: the role of space in population dynamics and interspecific interactions (MPB-30)*, volume 30. Princeton University Press, 2018.

[187] Fabio Danilo Torrisi and Alberto Bemporad. HYSDEL – a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249, 2004.

[188] Brian N. Turner and Scott A Gold. A review of melt extrusion additive manufacturing processes: II. Materials, dimensional accuracy, and surface roughness. *Rapid Prototyping Journal*, 21(3):250–261, 2015.

[189] Jurgen van Zundert, Joost Bolder, and Tom Oomen. Optimality and flexibility in iterative learning control for varying tasks. *Automatica*, 67:295–302, 2016.

[190] Bing-Chuan Wang, Han-Xiong Li, and Hai-Dong Yang. Spatial correlation based incremental learning for spatiotemporal modeling of battery thermal process. *IEEE Transactions on Industrial Electronics*, 2019.

[191] Feifan Wang, Sepehr Fathizadan, Feng Ju, Kyle Rowe, and Nils Hofmann. Print surface thermal modeling and layer time control for large-scale additive manufacturing. *IEEE Transactions on Automation Science and Engineering*, 2020.

[192] Jinjiang Wang, Laibin Zhang, Lixiang Duan, and Robert X Gao. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. *Journal of Intelligent Manufacturing*, 28(5):1125–1137, 2017.

[193] Zheng Wang, Farshad Harirchi, Dhananjay Anand, Chee Yee Tang, James Moyne, and Dawn Tilbury. Conflict-driven hybrid observer-based anomaly detection. In *2018 Annual American Control Conference (ACC)*, pages 5793–5800. IEEE, 2018.

[194] Zhi Wang, Patrick M Sammons, Christopher P Pannier, Kira Barton, and David J Hoelzle. System identification of a discrete repetitive process model for electrohydrodynamic jet printing. In *2018 Annual American Control Conference (ACC)*, pages 4464–4471. IEEE, 2018.

[195] Stephen J Wright. *Primal-dual interior-point methods*. SIAM, 1997.

[196] Bo Wu, Michael D Lemmon, and Hai Lin. Formal Methods for Stability Analysis of Networked Control Systems with IEEE 802.15.4 Protocol. *IEEE Transactions on Control Systems Technology*, 26(5):1635–1645, 2018.

[197] Haixi Wu, Zhonghua Yu, and Yan Wang. Real-time FDM machine condition monitoring and diagnosis based on acoustic emission and hidden semi-markov model. *The International Journal of Advanced Manufacturing Technology*, 90(5-8):2027–2036, 2017.

[198] Mingtao Wu, Zhengyi Song, and Young B Moon. Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods. *Journal of Intelligent Manufacturing*, pages 1–13, 2017.

[199] Huanxiong Xia, Jiacai Lu, Sadegh Dabiri, and Gretar Tryggvason. Fully resolved numerical simulations of fused deposition modeling. part i: fluid flow. *Rapid Prototyping Journal*, 24(2):463–476, 2018.

[200] Jun Xiong, Ziqiu Yin, and Weihua Zhang. Closed-loop control of variable layer width for thin-walled parts in wire and arc additive manufacturing. *Journal of Materials Processing Technology*, 233:100–106, 2016.

[201] Jun Xiong and Guangjun Zhang. Adaptive control of deposited height in gmaw-based layer additive manufacturing. *Journal of Materials Processing Technology*, 214(4):962–968, 2014.

[202] Lin Ye, Yongning Zhao, Cheng Zeng, and Cihang Zhang. Short-term wind power prediction based on spatial model. *Renewable energy*, 101:1067–1074, 2017.

[203] Shen Yin, Xiangping Zhu, and Chen Jing. Fault detection based on a robust one class support vector machine. *Neurocomputing*, 145:263–268, 2014.

[204] Jae Yoon, David He, and Brandon Van Hecke. A PHM approach to additive manufacturing equipment health monitoring, fault diagnosis, and quality control. In *Proceedings of the Prognostics and Health Management Society Conference, Fort Worth, TX, USA*, volume 29, pages 1–9. Citeseer, 2014.

[205] Zhenhua Yu, Jie Ouyang, Sisi Li, and Xia Peng. Formal modeling and control of cyber-physical manufacturing systems. *Advances in Mechanical Engineering*, 9(10):1687814017725472, 2017.

[206] Tansel Yucelen, Wassim M Haddad, and Eric M Feron. Adaptive control architectures for mitigating sensor attacks in cyber-physical systems. *Cyber-Physical Systems*, 2(1-4):24–52, 2016.

[207] Jianguo Zhang, Kai-Kuang Ma, Meng-Hwa Er, and Vincent Chong. Tumor segmentation from magnetic resonance imaging by learning via one-class support vector machine. *International Workshop on Advanced Image Technology (IWAIT '04)*, pages 207–211, 2004.

[208] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, 2017.

[209] Hesam Zomorodi and Robert G Landers. Extrusion based additive manufacturing using explicit model predictive control. In *American Control Conference (ACC), 2016*, pages 1747–1752. IEEE, 2016.

[210] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.