# Infinite-Dimensional Optimization: Modeling Abstractions and Software

by

Joshua L. Pulsipher

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2022

Date of final oral examination: 1/12/2022

The dissertation is approved by the following members of the Final Oral Committee:

Victor M. Zavala, Professor, Chemical and Biological Engineering
Reid C. Van Lehn, Assistant Professor, Chemical and Biological Engineering
Ross Swaney, Associate Professor, Chemical and Biological Engineering
Line Roald, Assistant Professor, Electrical and Computer Engineering

To my amazing wife and best friend, McKenna.

# ACKNOWLEDGMENTS

It has been a long road to come to this point and I would like to acknowledge everyone that has helped me on this journey toward earning my doctoral degree. I do my best to concisely attribute the support many as kindly offered me along the way.

In my public education, my fifth grade mathematics teacher (Mrs. Glines), my high school chemistry teacher (Mrs. Teske), and others were very influential in kindling my passion for using mathematics and science to describe the world around me and instilling my confidence to learn and master complex mathematical phenomena. These and other exceptional instructors certainly had large influence over my decision to pursue higher education in chemical engineering at the young age of 15. From there many professors, notably including Dr. Tanya Knickerbocker, Dr. Sam Mazhari, Dr. Martin Meister, Dr. Bradley Bundy, Dr. Thomas Knotts, and Dr. John Hedengren, did a lot to further instill my passion for learning (particularly in applying mathematics and chemistry in tackling engineering challenges).

I have to thank a chance encounter with Dr. John Hedengren at BYU for ultimately inspiring me to seek out a career in process systems engineering (PSE) research. In an unlikely exchange, I brought up my being trained unmanned aerial vehicle (UAV) pilot to which John offered me a job on the spot to help him start a UAV research team. In the years that ensued, John's mentorship did a lot to develop my skills as a researcher in PSE, cultivate my ability to sell my work, and help me realize my potential for pursuing a doctoral degree. I am also grateful to my group members (in the PRISM group) that I worked alongside with that helped me in transforming from the drone pilot into one of the core researchers, namely Abe Martin, Colter Lund, Jose Mojica, Ivan Rojas, Trent Okeson, Reza Asgharzadeh, Joseph Clark, Ammon Eaton, Landen Blackburn, Spencer Christiansen, and Logan Beal.

During my time here at UW-Madison I certainly need to acknowledge my advisor Dr. Victor Zavala for his patient, uplifting mentorship over the past 4 and a half years. Victor's what can I do for you attitude coupled with his charismatic character greatly influenced my decision to choose UW-Madison for my graduate studies and to choose Victor as my academic advisor. He has taught me so much during my time here that has shaped me into the researcher I am today. I am truly grateful for his mentorship and I plan to apply

much of what he has taught me going forward in my academic career.

I also have to thank my group members in the Zavalab whose friendship and support I treasure. Jordan Jalving, Sungho Shin, Apoorva Sampat, Ranjeet Kumar, and Yicheng Hu were key in helping me survive my technical elective courses and learn the ropes of modeling/analyzing optimization problems (primarily in Julia). Alexander Smith and Yue Shao helped me survive the challenging courses that we took together. I have also enjoyed collaborating with Weiqi Zhang and Shengli Jiang as our research paths have crossed. Finally, I will miss all the good conversations, food, and fun times I have shared with everyone in the group, not failing to mention Shiyi Qin, Jiaze Ma, Leonardo Gonzalez, Dilara Goreke, Blake Lopez, David Cole, Philip Tominac, and Jaron Thompson.

I have also had the privilege to mentor some exceptional undergraduate researchers which include Daniel Rios, Tyler Hongisto, Benjamin Davidson, Luke Coutinho, and Baide Xue. They were integral in helping me prepare many of the case studies presented in this work.

I am also grateful for the service rendered by my graduate defense committee members: Dr. Victor Zavala, Dr. Ross Swaney, Dr. Reid Van Lehn, and Dr. Line Roald. Their feedback has been key in developing this thesis and their questions/insights were quite helpful in shaping my future research plans.

I must also acknowledge the financial support I have graciously received along my journey toward the fulfillment of my PhD. The Running Start program in Washington state enabled me to start my higher education when I was 15 which was key in invigorating my appetite for higher-level education. The academic scholarship support at BYU also was key in enabling me to pursue my B.S. in chemical engineering. Finally, the support of department of chemical and biological engineering here at UW-Madison and the generous support by the U.S. Department of Energy under grant DE-SC0014114 have made all my graduate studies possible.

Finally, I must express my immense gratitude and love for my family that have resolutely stood by my side throughout my journey. My parents, John and Jennifer, have done so much to inspire me to pursue my dreams and ambitions, and helped me find the tools I needed. My brothers, Brian and Jacob, have also been a huge support over the years. A lot of credit must be given to my exceptional wife, McKenna, who has lovingly encouraged me through the tough times and patiently supported me through the busy times. I find it hard to imagine making it through these last 4 and a half years without her at my side. Finally, I acknowledge my faithful fluffy feline pandemic office companion, Rapunzel.

Joshua L. Pulsipher
Madison, WI
January 2022

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

## ABSTRACT

This dissertation presents unifying modeling abstractions for characterizing, analyzing, and solving infinite-dimensional optimization (InfiniteOpt) problems that are motivated by emergent applications in engineering. InfiniteOpt problems involve modeling components (variables, objectives, and constraints) that are functions defined over infinite domains. Examples include continuous-time dynamic optimization (time is an infinite domain and components are a function of time), PDE optimization problems (space and time are infinite domains and components are a function of space-time), as well as stochastic and semi-infinite optimization (random space is an infinite domain and components are a function of such random space). InfiniteOpt problems also arise from combinations of these problem classes (e.g., stochastic PDE optimization). Applications include model predictive control, process design, parameter estimation, reliability analysis, flexibility analysis, design of dynamic experiments, and more.

InfiniteOpt problems are often solved via discretization (e.g., finite differences), this is so predominant that application classes often forego InfiniteOpt representations and the use of other transformation techniques (e.g., projection unto orthogonal basis functions). Moreover, there exists a conceptual gap in abstracting these diverse optimization disciplines rigorously through a common lens. Coherent abstractions play a key role in facilitating innovative advancements and enabling general modeling languages.

To address this conceptual gap, we present a unifying abstraction for characterizing and modeling InfiniteOpt problems. This abstraction allows us to transfer techniques across disciplines and enable new modeling paradigms that include generalized risk measures, random field optimization, event-constrained optimization, and lifting-function based parameter estimation. Our abstraction serves as the backbone of an intuitive `Julia`-based modeling package called `InfiniteOpt.jl` which enables cutting-edge research and makes these advanced modeling techniques accessible.

Finally, we apply these principles to develop advanced abstractions for measuring and designing the flexibility and reliability of complex process systems. Flexibility denotes the ability of a system to maintain feasible operation in response to random fluctuations (i.e., continuous disturbances), and reliability assesses the ability of a system to continue feasible operation when subjected to random component failure (i.e., discrete events). Our proposed approaches facilitate the measuring and design of system flexibility/reliability that are more tractable and accurate relative to existing techniques.

# Chapter 1

## INTRODUCTION

In this chapter, we present the objectives and motivation that are at the very center of this dissertation. We introduce infinite-dimensional optimization to the reader and discuss the inherent complexities and conceptual gaps that are manifest in this challenging discipline. We also summarize the prevalent existing modeling approaches in the literature and establish our research objectives in building upon these approaches in an effort toward establishing a unifying abstraction to facilitate transfer between the existing paradigms and engender new ones. Finally, we summarize the structure and content of this dissertation.

## 1.1 Infinite-Dimensional Optimization

Infinite-dimensional optimization (InfiniteOpt) problems contain parameters that live in infinite domains (e.g., time, space, random) [2]; the components of these problems (variables, objectives, and constraints) are parameterized over these domains and thus are functions with infinite-dimensional domains (they form manifolds and surfaces). A classical example of an InfiniteOpt problem is continuous-time dynamic optimization [3]; here, the control trajectory is a function of time and time is a parameter that lives in an infinite-dimensional (continuous) domain. This formulation contrasts with that of a discrete-time dynamic optimization problem, in which the control trajectory is a collection of values

defined over a finite set of times (domains are finite). Given the infinite-dimensional nature of variables, objectives, and constraints, one requires specialized techniques to define an InfiniteOpt problem properly. This is done by using *measures*, which are operators that summarize/collapse an infinite-dimensional object into a scalar quantity. For instance, in dynamic optimization, one often minimizes the integral of the cost over the time domain and, in stochastic optimization, one often minimizes the expected value or variance of the cost. Measures are thus a key modeling element of InfiniteOpt problems that help manipulate the *shape* of infinite-dimensional objects to achieve desired outcomes (e.g., minimize peak/extreme costs or satisfy constraints with high probability). InfiniteOpt problems also often contain differential operators that dictate how components evolve over their corresponding domains; these operators often appear in problems that include differential and algebraic equations (DAEs) and partial differential equations (PDEs).

InfiniteOpt problems encompass a wide range of classical fields such as dynamic optimization [4], PDE-constrained optimization [5], stochastic optimization [6], and semi-infinite optimization [7]. We also often encounter InfiniteOpt problems that are obtained by combining infinite-dimensional domains (e.g., space, time, and random domains). This situation arises, for instance, in stochastic dynamic optimization (e.g., stochastic optimal control) problems and in optimization problems with stochastic PDEs. In these problems, one needs to define measures that summarize modeling objects that are defined over the multiple domains (e.g., minimize the space-time integral of the cost or minimize the expected value of the time integral of the cost). InfiniteOpt problems appear in applications such as continuous-time model predictive control and moving horizon estimation [8, 9], design under uncertainty [10, 11], portfolio planning [12, 13], parameter estimation for differential equations [14, 15], reliability analysis [16, 17], optimal power flow [18, 19], and dynamic design of experiments [20, 21].

The infinite-dimensional nature of modeling objects make InfiniteOpt problems challenging to solve [22, 23, 24]. Specifically, these problems often need to be *transcribed/transformed* into a finite dimensional representation via discretization. For instance, differen-

tial equations and associated domains are often discretized using finite difference and quadrature schemes [4], while expectation operators and associated random domains are often discretized using Monte Carlo (MC) sampling and quadrature schemes [25, 26]. The finite-dimensional representation of the problem can be handled using standard optimization solvers (e.g., `Ipopt` and `Gurobi`). Sophisticated transformation techniques are used in different scientific disciplines; for example, projection onto orthogonal basis functions is a technique that is commonly used in PDE and stochastic optimization [2, 27].

## 1.2 Current Modeling Approaches

InfiniteOpt problems are typically modeled uniquely according to the nomenclature and practices that are specific to their particular optimization discipline. In dynamic optimization, problems defined on continuous time $t \in \mathcal{D}_t$ are typically modeled with an integrated dynamic cost function $f(\cdot)$ that is subject to DAEs $g(\cdot) = 0$ that model the system physics (typically establishing the behavior of the state variables) and path/point constraints $h(\cdot) \leq 0$ that put restrictions on state and/or control variable values:

$$\begin{aligned} \min_{y(t)} \quad & \int_{t \in \mathcal{D}_t} f(y'(t), y(t), t) dt \\ \text{s.t.} \quad & g(y'(t), y(t), t) = 0, \qquad t \in \mathcal{D}_t \\ & h(y(t), t) \leq 0, \qquad t \in \mathcal{D}_t \end{aligned} \tag{1.1}$$

where $y(t) \in \mathbb{R}^{n_y}$ are state/control variables and $y'(t) \in \mathbb{R}^{n_y}$ are their first-order time derivatives [3]. These encapsulate multiple application areas in chemical engineering such as optimal control [28] and dynamic parameter estimation [14]. Moreover, (1.1) is often solved via discretization over finite time points. This approach is so prominent in the literature that many models are solely characterized in discrete-time (implicitly being discretized approximations of the underlying continuous-time model) [29].

PDE-constrained optimization problems typically exhibit an integral measured cost

function $f(\cdot)$ that is subject to PDEs $g(\cdot) = 0$ that characterize the system behavior (e.g., Fick's second law) and path/point constraints $h(\cdot) \leq 0$ that place restrictions on the decision space. In chemical engineering, PDE-constrained problems typically correspond to space-time problems (defined over space-time $(t, x) \in \mathcal{D}_{t,x}$) that are modeled:

$$\begin{aligned} \min_{y(t,x)} \quad & \int_{(t,x)\in\mathcal{D}_{t,x}} f(Dy(t,x), y(t,x), t, x)dtdx \\ \text{s.t.} \quad & g(Dy(t,x), y(t,x), t, x) = 0, && (t,x) \in \mathcal{D}_{t,x} \\ & h(y(t,x), t, x) \leq 0, && (t,x) \in \mathcal{D}_{t,x} \end{aligned} \tag{1.2}$$

where $Dy(t, x)$ denote the collection of partial derivatives used in the problem. In this form, (1.2) clearly closely resembles the form of its dynamic counterpart in (1.1). PDE-constrained problems do not however exhibit the same affinity toward discretized forms due to the combinatorial nature of constructing space-time support grids that readily can incur tractability concerns [30]. Hence, so-called order reduction techniques are often employed to transform the InfiniteOpt problem into a tractable finite form via projecting it onto a collection of orthogonal basis functions as is discussed further in Section 2.3.2.

Two-stage stochastic optimization problems defined with a random parameter $\xi \in \mathcal{D}_\xi$ (where $\mathcal{D}_\xi$ is the co-domain of its distribution) are characterized by a cost function $f(\cdot)$ measured by a risk measure $R_\xi[\cdot]$ and features first/second stage constraints $g(\cdot) \leq 0$:

$$\begin{aligned} \min_{z,y(\xi)} \quad & R_\xi[f(z, y(\xi), \xi)] \\ \text{s.t.} \quad & g(z, y(\xi), \xi) \leq 0, \quad \xi \in \mathcal{D}_\xi \end{aligned} \tag{1.3}$$

where $z \in \mathbb{R}^{n_z}$ are first stage variables and $y(\xi) \in \mathbb{R}^{n_y}$ are second stage variables. These are typically solved via direct transcription over Monte Carlo (MC) samples of $\xi$ [26]. Thus, we observe that (1.3) also shares a number of similar modeling elements and solution approaches with its dynamic and PDE-constrained counterparts.

Although common mathematical elements of InfiniteOpt problems are found across

disciplines, there are limited approaches/tools available to model and solve these problems in a unified manner. Powerful, domain-specific software implementations are currently available for tackling dynamic and PDE optimization problems; examples include `Gekko`, `ACADO`, and `gPROMS` [31, 32, 33]. A key limitation of these tools is that the modeling abstraction used is specialized to specific problem classes and are not that easy to extend. On the other hand, there are powerful algebraic modeling languages such as `JuMP`, `CasADi`, `PETSc`, and `Pyomo` that offer high modeling flexibility to tackle different problem classes; however, these tools require the user to transform InfiniteOpt problems manually (which is tedious and prone to error). These limitations have recently motivated the development of modeling frameworks such as `Pyomo.dae` [22]; this framework unifies the notion of variables, objectives, and constraints defined over continuous, infinite-dimensional domains (sets). This abstraction facilitates the modeling and transformation (via automatic discretization techniques) of optimization problems with embedded DAEs and PDEs. A limitation of this abstraction is that the notion of continuous sets is limited to space and time and this hinders generalizability (e.g., domains defined by random parameters need to be treated separately). Moreover, this framework provides limited capabilities to define measures (e.g., multi-dimensional integrals and risk functionals).

A unifying abstraction for InfiniteOpt problems can facilitate the development of software tools and the development of new analysis and solution techniques. For instance, it has been recently shown that a graph abstraction unifies a wide range of optimization problems such as discrete-time dynamic optimization (graph is a line), network optimization (graph is the network itself), and multi-stage stochastic optimization (graph is a tree) [34]. This unifying abstraction has helped identify new theoretical properties and decomposition algorithms [35, 36]; this has been enabled, in part, via transferring techniques and concepts across disciplines. The limited availability of unifying modeling tools ultimately limits our ability to experiment with techniques that appear in different disciplines and limits our ability to identify new modeling abstractions to tackle emerging applications.

## 1.3 Flexibility and Reliability Analysis

An important group of stochastic InfiniteOpt problems pertain to those that arise in flexibility and reliability analysis. These are applied to physical systems that are subjected to uncertainties and feature variables that are parameterized over a random domain. Such uncertainty may stem from a wide variety of sources including the system's environment (e.g., ambient temperature, product demand) and the system itself (e.g., kinetic constants, battery life, equipment failure). Flexibility denotes the ability of a system to maintain feasible operation over a range of continuous uncertain/random conditions [37]. Whereas, reliability denotes the ability of a system to feasibly operate in face of discrete random failures [38]. This becomes prevalent in many applications such as chemical processes [39, 40], power systems [41, 42], and autonomous vehicles [43, 44] where it is critical that sufficient flexibility/reliability is engineered into system design to promote profitability and safety. For instance, chemical plants encounter numerous uncertainties (e.g., feed flowrates, heat transfer coefficients, equipment fatigue) that can induce costly failures for designs that are not sufficiently flexible and reliable [45]. Accounting for these quantities can also help reduce design costs, since accurate analysis of a system's flexibility/reliability helps to prevent the over-engineering that has prevailed traditional chemical process design practices [37].

Flexibility analysis is mature field in which a number of approaches to measure system flexibility have been proposed, and an extensive review is provided in [46]. Two prominent measures have emerged from this effort: the stochastic flexibility ($SF$) index and the flexibility ($F$) index [47, 37]. The $SF$-index adopts stochastic programming approaches to define joint-probability of satisfying all the system constraints when subjected to uncertainty with a known distribution. Typically, this is evaluated with MC sampling or quadrature techniques, but these have historically suffered from scalability issues [47, 48, 49, 50]. The $F$-index characterizes the uncertainty via an uncertainty set (similar to what is used in robust optimization) and uses a semi-infinite InfiniteOpt prob-

lem to determine the largest uniformly scaled set of uncertainty values the system can withstand while satisfying its constraints; the resulting scaling factor is then taken to be the $F$-index [51]. This has seen fairly wide adoption in a variety of application areas. However, traditionally this approach has only utilized hyperbox uncertainty sets which do not capture parameter correlation and might be overly conservative in some cases [52].

Reliability has been traditionally defined as the probability that a system remains functional under component failures [38]. The most prominent model used in industry to quantify reliability is based on so-called reliability block diagrams (RBDs). Here, the system is modeled as a network (a directed graph) of series/parallel paths in which each path has a single source and sink node. The system is said to function under a given failure if there exists at least one path between the source and the sink node. The RBD approach exploits the simple topology of series/parallel systems to analytically compute the reliability of the overall system from the reliability of its individual components [53]. Here, it is also implicitly assumed that the probability of failure for every component can be characterized using the same probability distribution. The availability of an analytical measure facilities the design of systems of maximum reliability [54]. Unfortunately, the RBD approach is difficult to apply to more complex settings that involve, for instance, topologies with multiple source and sink nodes and loops and components with different probability distributions. As a result, analytical reliability measures cannot be easily derived under such settings.

The recursive decomposition algorithm (DFA) is a technique that aims to quantify reliability of more complex network topologies by systematically exploring paths between source and sink nodes [55]. This approach is more general but is not amenable for design tasks. Simulation-based approaches such as Monte Carlo (MC) sampling provide a general approach to quantify reliability. These approaches estimate reliability by “probing" the system against failure scenarios and from this determine the probability that the system remains functional [56]. This approach is computationally more expensive than the analytical RBD approach because it requires repetitive simulations but can also enable

the use of a wide range of stochastic programming formulations and solution techniques [57]. Such a connection to stochastic programming techniques could enable general optimal design strategies, but such an extension has not yet been investigated.

## 1.4 Research Objectives

The overarching vision of our work is to establish new modeling abstractions and approaches that generally capture InfiniteOpt problems in a unified manner to set a foundation for enhanced information transfer and collaboration. In other words, we hope to bridge the divide that exists between these historically isolated optimization disciplines through the power of abstraction. From this we also hope to inspire and create unified software tools to facilitate this research effort and make these advanced modeling approaches more accessible to engineers in emergent application fields. Our work will have a particular focus on problems that entail random effects and/or space-time domains since these are key considerations in many biological and chemical engineering application areas such as molecular dynamics, micro-kinetics, process intensification, process control, fluid dynamics, and microbial communities.

With this vision in mind, the first part of this dissertation seeks to establish a unifying modeling abstraction for InfiniteOpt problems and explore new modeling paradigms that arise from theoretical transfer between different disciplines. In this effort, we seek to accomplish the following objectives:

- to formalize a modeling abstraction with unified nomenclature and notation for general InfiniteOpt problems,
- to enable advanced solution techniques through generalized formulation transformations,
- to implement this new modeling/transformation abstraction in software,
- to develop new modeling techniques in dynamic, PDE-constrained, and stochastic

optimization using novel theory transfer, and

- to create a unified approach for modeling uncertainty over general infinite domains (e.g., space and/or time) in an optimization context.

In the second and final part of this dissertation, we focus on applying these principles to develop generalized approaches for measuring and designing the flexibility and reliability of complex process systems. For this, we set the following objectives:

- to incorporate more accurate characterizations of system uncertainty,
- to develop systematic approaches for identifying system vulnerabilities, and
- to enhance the scalability of optimal reliability/flexibility design problems.

## 1.5 Thesis Overview

This body of this dissertation is broken up into two parts. Part I comprises Chapters 2 - 5 and addresses the research objectives for developing and exploring a unifying modeling abstraction for InfiniteOpt problems. Part II comprises Chapters 6 - 8 and tackles the research objectives for flexibility and reliability analysis. We summarize of each chapter below.

Chapter 2 details the unifying modeling abstraction we have developed for general treatment of InfiniteOpt problems. In this, we establish unified nomenclature and notation to express these problems. We also characterize generalized transformation methods to solve these problems, and we detail how this all is implemented in a `Julia` package called `InfiniteOpt.jl`. We further motivate this modeling approach through some illustrative case studies.

Chapter 3 features a discovery of how risk measures from stochastic optimization (which are used to summarize random cost functions) can be transferred to functions over general domains. This innovation is particularly pertinent to dynamic optimization

formulations where these measures enable new ways to shape optimal policy trajectories. We formalize the theory and properties of this new class of measures and demonstrate their utility in the context of optimal control.

Chapter 4 presents a new area of optimization that we call random field optimization which incorporates random field theory to characterize uncertainty over general problem domains. This general paradigm captures existing modeling approaches such as multi-stage stochastic optimization, but is able to generalize over continuous time and spatial position. This forms the foundation for a new class of optimization/modeling strategies to enable new applications which we demonstrate via case studies.

Chapter 5 introduces other innovations that are enabled through our proposed unifying abstraction for InfiniteOpt problems. We propose a new constraint class called event constraints which enforce constraints on the probability of invoking certain events. These capture chance constraints from stochastic optimization as a special case and can be applied to general problem domains. We also present a new approach for parameter estimation over infinite domains with discrete data. Lifting-functions are used to cast these as InfiniteOpt problems which enhance the problem tractability and the parameter accuracy relative to traditional methods.

Chapter 6 presents our abstraction for measuring system flexibility using generalized uncertainty sets. It also details new analysis approaches such as limiting constraint ranking which helps to identify system vulnerabilities. This modeling abstraction is implemented in a `Julia` package called `FlexibilityAnalysis.jl`. We exemplify these advances via case studies.

Chapter 7 builds upon the work of Chapter 6 to establish a stochastic programming approach for designing flexible systems. Moreover, we propose a novel relaxation technique that significantly increases the scalability of this approach relative to classical methods. We use case studies to demonstrate the quality of the relaxation approach empirically.

Chapter 8 establishes a new graph-based abstraction for assessing and promoting

the reliability of complex systems. This provides a straightforward modeling approach to assessing the reliability of general process systems enables a new class of reliability design problems. The tractability of these design approaches is enhanced by extending the relaxation approach developed in Chapter 7.

Finally, Chapter 9 concludes this dissertation by summarizing our key findings. It also identifies future avenues of research to build upon the abstractions and approaches presented in this work.

# Part I

ABSTRACTING GENERAL INFINITEOPT PROBLEMS

# Chapter 2

## UNIFYING MODELING ABSTRACTION

The content of this chapter is published in [30].

## 2.1 Introduction

Following from the discussion in Section 1.2, we seek to develop a unifying abstraction for InfiniteOpt problems can facilitate the development of software tools and the development of new analysis and solution techniques.

In this chapter, we propose such a unifying abstraction that facilitates the analysis, modeling, and solution of InfiniteOpt problems (see Figure 2.1). Central to our abstraction is the notion of infinite parameters, which are parameters defined over infinite-dimensional domains (e.g., time, space, and random parameters). The proposed abstraction allows us to construct these domains systematically by using Cartesian product operations and to define variables, objectives, and constraints over such domains and subdomains (restricted domains). The ability to handle restricted subdomains allows us to capture infinite-dimensional and standard (finite-dimensional) variables in a unified setting. Another key notion of the proposed abstraction are measure operators; these operators allow us to summarize infinite-dimensional objects over specific domains or subdomains and with this formulate problems with different types of objectives and con-

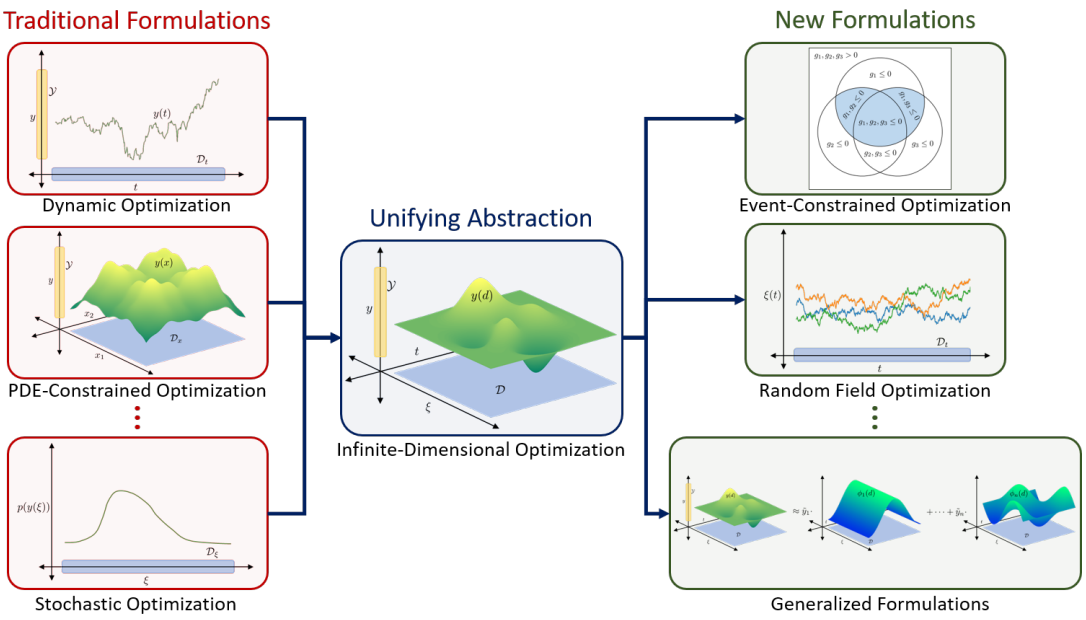



Figure 2.1: Summary of the proposed InfiniteOpt abstraction; the abstraction seeks to unify existing problem classes and use this to develop new classes.

straints. The proposed abstraction also incorporates differential operators, which are used to model how variables evolve other their corresponding domains. These modeling elements provide a bridge between different disciplines and enables cross-fertilization. For instance, we show that the proposed abstraction allows us to leverage the use of risk measures (used in stochastic optimization) to shape time-dependent trajectories (arising in dynamic optimization). The proposed abstraction also facilitates the development of new abstractions such as event-constrained optimization problems and optimization problems with space-time random fields. The proposed abstraction forms the basis of a `Julia`-based modeling package that we call `InfiniteOpt.jl`. In this context, we show that the abstraction facilitates software development and enables a compact and intuitive modeling syntax and the implementation of transformation techniques (e.g., quadrature and sampling).

The chapter is structured as follows. Section 2.2 details the proposed unifying abstraction. Section 2.3 reviews problem transformations into finite-dimensional representations

through the lens of the abstraction. Section 2.4 discusses the implementation of this modeling paradigm in `InfiniteOpt.jl`. Section 2.5 presents case studies to demonstrate how this abstraction captures existing InfiniteOpt problem classes. Later chapters will highlight the innovations that are enabled by this modeling abstraction.

## 2.2 InfiniteOpt Abstraction

In this section, we outline the proposed unifying abstraction for InfiniteOpt problems. Specifically, we discuss the different elements of the abstraction, which include infinite domains and parameters, decision variables, measure operators, and differential operators.

### 2.2.1 *Infinite Domains and Parameters*

An InfiniteOpt problem includes a collection of infinite domains $\mathcal{D}_\ell$ with index $\ell \in \mathcal{L}$. An individual infinite domain is defined as $\mathcal{D}_\ell \subseteq \mathbb{R}^{n_\ell}$. The term *infinite* refers to the fact that an infinite domain has infinite cardinality (i.e., $|\mathcal{D}_\ell| = \infty$) and is thus continuous. We also note that each infinite domain $\mathcal{D}_\ell$ is a subdomain of an $n_\ell$-dimensional Euclidean space $\mathbb{R}^{n_\ell}$.

An infinite domain encompasses different domains encountered in applications; for instance, this can represent a time domain of the form $\mathcal{D}_t = [t_0, t_f] \subset \mathbb{R}$ (with $n_t = 1$), a 3D spatial domain $\mathcal{D}_x = [-1, 1]^3 \in \mathbb{R}^3$ (with $n_x = 3$), or the co-domain of a multivariate random variable $\mathcal{D}_\xi = (-\infty, \infty)^m \in \mathbb{R}^m$ (with $n_\xi = m$).

Infinite parameters are parameters that live in the associated infinite domains; specifically, we define the general parameter $d \in \mathcal{D}_\ell$. In our previous examples, the parameters are time $t \in \mathcal{D}_t$, space $x \in \mathcal{D}_x$, and random parameters $\xi \in \mathcal{D}_\xi$. We will see in Section 2.2.2 how infinite parameters are used to index infinite variables (i.e., they are the independent variables of the decision functions).

The domain of the InfiniteOpt problem is the Cartesian product of the infinite do-

mains:

$$\mathcal{D} := \prod_{\ell \in \mathcal{L}} \mathcal{D}_\ell. \tag{2.1}$$

The construction of the problem domain is exemplified in Figure 2.2; here, we see that the domain obtained from the Cartesian product of a 1D random domain $\mathcal{D}_\xi$ and a 1D temporal domain $\mathcal{D}_t$.

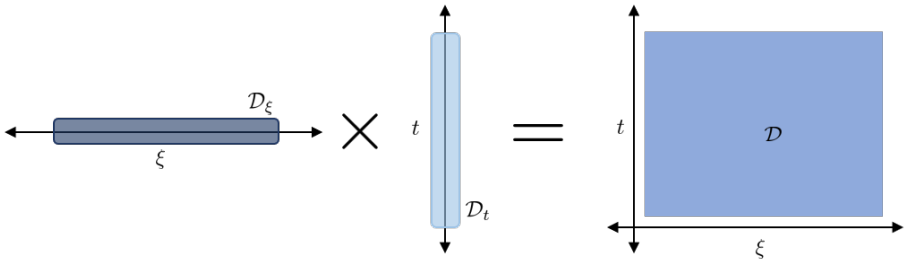



Figure 2.2: Cartesian product of random domain $\mathcal{D}_\xi$ and time domain $\mathcal{D}_t$ to produce $\mathcal{D}$.

### 2.2.2 *Decision Variables*

A key feature of an InfiniteOpt problem is that it contains decision variables that are *functions* of infinite-dimensional parameters; as such, decision variables are also infinite-dimensional. In the proposed abstraction, we also consider InfiniteOpt problems that contain finite-dimensional variables (that do not depend on any parameters) and finite-dimensional variables that originate from reductions of infinite variables (e.g., integration over a domain or evaluation of an infinite variable at a point in the domain). To account for these situations, we define different types of variables: infinite, semi-infinite, point, and finite.

Infinite variables $y : \mathcal{D} \mapsto \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ are functions that map an infinite domain $\mathcal{D}$ to the domain $\mathcal{Y}$. These variables are expressed as:

$$y(d) \in \mathcal{Y},\ d \in \mathcal{D}. \tag{2.2}$$

Figure 2.3 shows that infinite variables are functions that can be interpreted as man-

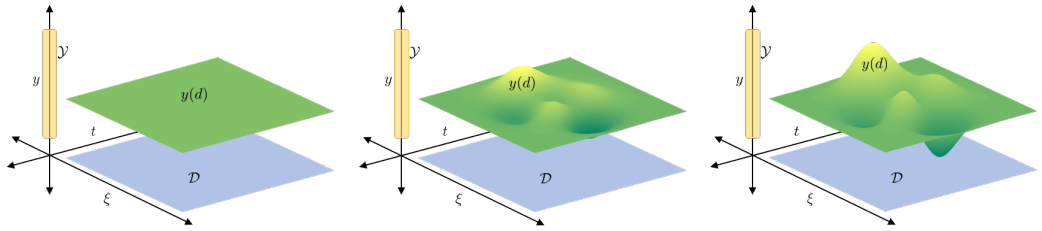
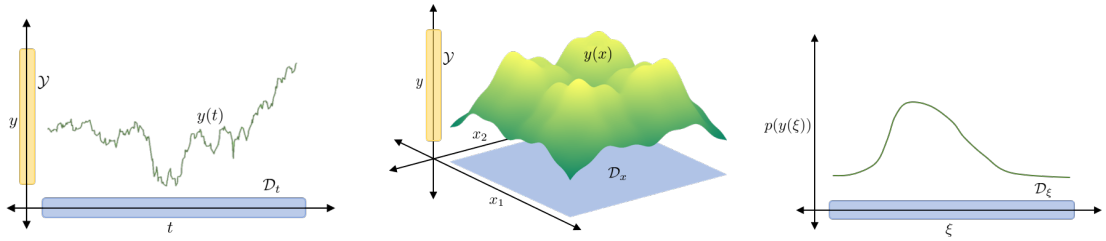



Figure 2.3: Depiction of realizations of an infinite variable $y(d)$ with $\mathcal{Y} \subseteq \mathbb{R}$ and $\mathcal{D} \subseteq \mathbb{R}^2$. The horizontal axes define the domain $\mathcal{D}$ and the vertical axis denotes the domain of feasible decisions $\mathcal{Y}$.

ifolds (also known as surfaces and fields). The goal of the InfiniteOpt problem is to shape these manifolds to achieve pre-determined goals (e.g., minimize their mean value or their peaks). Example classes of infinite variables include uncertainty-dependent decision policies arising in stochastic optimization (i.e., recourse variables), time-dependent control/state policies arising in dynamic optimization, or space-time fields arising in PDE optimization. For instance, in a stochastic PDE optimization problem, one might have an infinite variable of the form $y(t, x, \xi)$, which is simultaneously parameterized over time $t$, space $x$, and uncertainty $\xi$. In other words, an infinite variable is equivalent to a collection of finite decision variables indexed over an infinite domain (producing an infinite collection of variables). Figure 2.4 illustrates some infinite variables commonly encountered in different disciplines.




(a) Dynamic Optimization (b) PDE Optimization (c) Stochastic Optimization

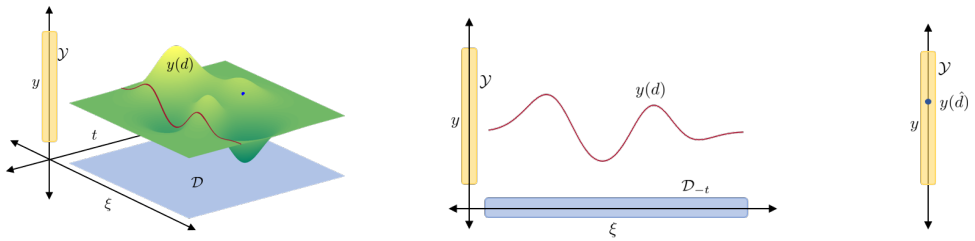
Figure 2.4: Example of infinite variables arising in traditional formulations. The recourse variable $y(\xi)$ is visualized in terms of its probability density function (as is customary in stochastic optimization).

Semi-infinite variables $y : \mathcal{D}_{-\ell} \mapsto \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ correspond to infinite variables in which the subdomain $\mathcal{D}_\ell$ has been restricted/projected to a single point $\hat{d}_\ell$; the restricted domain

is denoted as $\mathcal{D}_{-\ell}$. These variables are also functions that map from the infinite domain $\mathcal{D}_{-\ell}$ to the domain $\mathcal{Y}$:

$$y(d) \in \mathcal{Y},\ d \in \mathcal{D}_{-\ell}. \tag{2.3}$$

We refer to $\hat{d}_\ell \in \mathcal{D}_\ell$ as a support point of the domain. A depiction of how semi-infinite variables are derived from infinite variables via projection is provided in Figure 2.5. Example classes mirror those of infinite variables with multiple parameter dependencies; for example, in a stochastic PDE problem, we might want to evaluate the variable $y(t, x, \xi)$ at the support point $t = 0$ (initial time); this gives the semi-infinite variable $y(0, x, \xi)$ and domain $\mathcal{D}_{-t} = \mathcal{D}_x \times \mathcal{D}_\xi$.




(a) Infinite (b) Semi-Infinite (c) Point

Figure 2.5: Illustration of how a semi-infinite variable $y(d),\ d \in \mathcal{D}_{-\ell}$ and a point variable $y(\hat{d})$ are obtained from an infinite variable $y(d),\ d \in \mathcal{D}$ via restriction/projection. Semi-infinite and point variables are realizations of an infinite variable and live in the domain $\mathcal{Y}$.

Point variables denote infinite variables in which the entire $\mathcal{D}$ is restricted to a single point $\hat{d}$. These are finite-dimensional variables that can be expressed as:

$$y(\hat{d}) \in \mathcal{Y}. \tag{2.4}$$

Figure 2.5 illustrates how point variables relate to other variable classes. Examples of point variables include random variables evaluated at a specific scenario/sample/realization of the uncertain parameter or a space-time variable evaluated at a specific point in the domain (e.g., boundary conditions). Point variables are also used in dynamic optimization to specify so-called point constraints (these ensure that a time trajectory satisfies

a set of constraints at specific time points).

The proposed abstraction also considers finite variables:

$$z \in \mathcal{Z} \subseteq \mathbb{R}^{n_z} \tag{2.5}$$

where $\mathcal{Z}$ denotes the feasible set. These are variables that are not parameterized over an infinite domain. Examples of finite variables arising in applications are first-stage decision variables and design variables arising in stochastic optimization or model parameters estimated in a dynamic optimization problem. Figure 2.6 shows that this variable is analogous to the point variable depicted in Figure 2.5c. In fact, this highlights that a point variable is a finite variable; the difference is that a point variable is derived from a restriction of an infinite variable. However, technically speaking, a finite variable can be seen as a special case of an infinite variable in which the domain is a point. As such, infinite variables provide a unifying framework to capture different types of variables.


Figure 2.6: A depiction of a finite variable $z \in \mathcal{Z} \subseteq \mathbb{R}$.

### 2.2.3 *Differential Operators*

Differential operators are a typical modeling element of InfiniteOpt problems. These operators capture how a given decision variable (an infinite-dimensional function) changes over its associated domain; for example, in a dynamic optimization problem, we want to know how quickly does a state/control variable change in time.

The proposed abstraction defines a differential operator of the general form:

$$D : \mathscr{Y} \mapsto \mathscr{D} \tag{2.6}$$

where $\mathscr{Y}$ is a function space for infinite variables $y(d)$ and $\mathscr{D}$ is a function space that

describes the differential output functions. Differential operators are applied to infinite variables $y(d),\ d \in \mathcal{D}$. These operators capture partial derivatives over individual parameters and more sophisticated operators that simultaneously operate on multiple parameters such as the Laplacian operator (typically encountered in PDE optimization). Figure 2.7 illustrates a differential operator acting on an infinite variable.

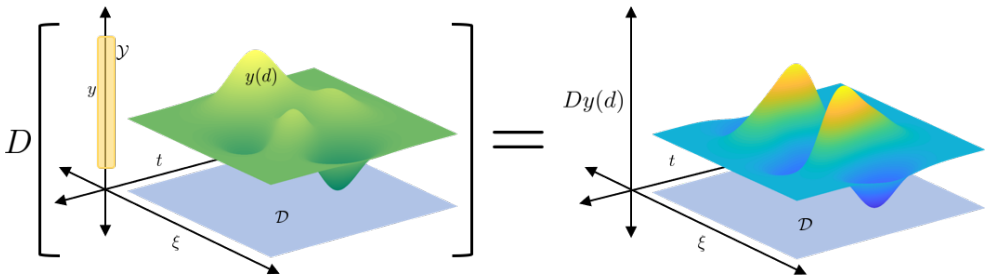



Figure 2.7: Depiction of a differential operator $D$ acting on the infinite variable $y(d)$.

Differential operators map the infinite domain $\mathcal{D}$ to the scalar domain $\mathbb{R}$. The output of a differential operator is an infinite variable $Dy(d),\ d \in \mathcal{D}$ that inherits the domain of the argument $y(d)$. For example, consider the infinite variable $y(t),\ t \in \mathcal{D}_t$; the partial derivative operator is an infinite variable $y'(t) := \partial y(t)/\partial t,\ t \in \mathcal{D}_t$. Some other specific examples include the partial derivative $\frac{\partial y(t,x,\xi)}{\partial t},\ (t,x,\xi) \in \mathcal{D}$ and the Laplacian $\Delta y(x),\ x \in \mathcal{D}_x$. Note that derivatives of the form $\frac{\partial}{\partial \xi}$ are not typically used in stochastic optimization problems; however, the proposed abstraction allows for this operator to be defined. This modeling feature can be used, for instance, to control how a random variable changes in the uncertainty space (this can be used to manipulate the shape of its probability density function).

### 2.2.4 *Measure Operators*

Measure operators are key modeling constructs that are used to *summarize* functions by collapsing them to a single quantity. For example, in a dynamic optimization problem, one typically minimizes the time-integral of the cost (a scalar quantity). The proposed

abstraction defines a measure operator of the general form:

$$M_\ell : \mathscr{Y} \mapsto \mathscr{M} \tag{2.7}$$

where $\mathscr{Y}$ is the function space of infinite variable inputs and $\mathscr{M}$ denotes the function space of output measured functions. Here, the index $\ell$ indicates that the operator is applied on the subdomain $\mathcal{D}_\ell$ and thus has the effect of restricting the domain. As such, the output of a measure operator is a semi-infinite variable that lives in the restricted domain $\mathcal{D}_{-\ell}$. Figure 2.8 illustrates such a measure operator.

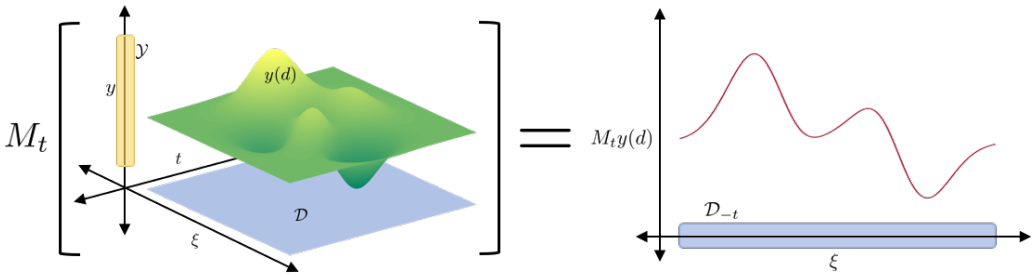



Figure 2.8: Measure operator $M_t$ that acts on domain $\mathcal{D}_t$ of the infinite variable $y(d)$, $d \in \mathcal{D}$. This operation returns the semi-infinite variable $m(\xi) := M_t\, y,\ \xi \in \mathcal{D}_\xi$.

Measure operators are a key feature of InfiniteOpt problems; specifically, objective functions and constraints are often expressed in terms of measure operators. For instance, consider a field $y(t, x, \xi)$ arising in a stochastic PDE problem; one can define measure operator that computes the time-integral $m(x, \xi) := \int_{t \in \mathcal{D}_t} y(t, x, \xi) dt,\ (x, \xi) \in \mathcal{D}_x \times \mathcal{D}_\xi$ and note that the output of this operation is a semi-infinite variable $m(x, \xi)$ that lives in $\mathcal{D}_{-t} = \mathcal{D}_x \times \mathcal{D}_\xi$. One can also define an operator that computes the expectation $m(t, x) := \int_{\xi \in \mathcal{D}_\xi} y(t, x, \xi) p(\xi) d\xi,\ (t, x) \in \mathcal{D}_t \times \mathcal{D}_x$ (where $p(\cdot)$ is the probability density function of $\xi$); this operation gives a semi-infinite variable $m(t, x)$ that lives in $\mathcal{D}_{-\xi} = \mathcal{D}_t \times \mathcal{D}_x$.

The expectation is a measure operator that is of particular interest in stochastic optimization because this can be used to compute different types of risk measures and probabilities; for instance, in the previous example, one might want to compute a probability

of the form:

$$\mathbb{P}_{\xi}(y(t,x,\xi) \in \mathcal{Y}), \quad (t,x) \in \mathcal{D}_t \times \mathcal{D}_x. \tag{2.8}$$

This is the probability that $y(t,x,\xi)$ is in the domain $\mathcal{Y}$ and can be computed by using an expectation operator:

$$\begin{aligned} \mathbb{P}_{\xi}(y(t,x,\xi) \in \mathcal{Y}) &= \mathbb{E}_{\xi}\left[\mathbb{1}[y(t,x,\xi) \in \mathcal{Y}]\right] \\ &= \int_{\xi \in \mathcal{D}_\xi} \mathbb{1}[y(t,x,\xi) \in \mathcal{Y}] p(\xi) d\xi. \end{aligned} \tag{2.9}$$

where $\mathbb{1}[\cdot]$ is the indicator function and the argument of this function is the *event* of interest. We recall that the indicator function returns a value of 0 if the event is not satisfied or a value of 1 if the event is satisfied. An important observation is that the indicator function can be used to define a wide range of measures and over different types of domains; for instance, the measure $\int_{t \in \mathcal{D}_t} \mathbb{1}[y(t) > \bar{y}] dt$ denotes the amount of time that the function $y(t),\ t \in \mathcal{D}_t$ crosses the threshold $\bar{y}$.

Measure operators can also be used to summarize infinite variables over multiple subdomains; for example, one can consider the following measures:

$$M_{t,x}\, y = \int_{t \in \mathcal{D}_t} \int_{x \in \mathcal{D}_x} y(t,x,\xi) dx dt, \quad \xi \in \mathcal{D}_\xi \tag{2.10a}$$

$$M_{t,x,\xi}\, y = \mathbb{E}_{\xi}\left[\int_{t \in \mathcal{D}_t} \int_{x \in \mathcal{D}_x} y(t,x,\xi) dx dt\right] \tag{2.10b}$$

$$M_{t,x,\xi}\, y = \mathbb{E}_{\xi}\left[\int_{t \in \mathcal{D}_t} \int_{x \in \mathcal{D}_x} \mathbb{1}[y(t,x,\xi) \in \mathcal{Y}] dx dt\right]. \tag{2.10c}$$

One can thus see that a wide range of measures can be envisioned.

### 2.2.5 *Objectives*

In InfiniteOpt problems, objective functions are functions of infinite variables; as such, objectives are infinite variables. Minimizing or maximizing an infinite-dimensional function does not yield a well-posed optimization problem. This situation is similar in spirit

to that appearing in multi-objective optimization problem, in which we seek to simultaneously minimize/maximize a finite collection of objectives (in an InfiniteOpt problem, the collection is infinite). To pose well-defined formulations, one often resorts to scalarization techniques; the idea is to reduce/summarize the infinite-dimensional function into a single scalar quantity. The goal of this scalarization procedure is to manipulate the shape of the infinite-dimensional objective (e.g., minimize its mean value or its extreme value). Scalarization is performed by using measure operators; for instance, in the context of multi-objective optimization, one scalarizes the objectives by computing a weighted summation of the objectives. In an InfiniteOpt setting, this weighting is done by computing a weighted integral of the objective. For instance, in dynamic optimization, we often have a time-dependent objective function $f(t) := f(y(t), t),\ t \in \mathcal{D}_t$; here, we can notice that the objective depends on an infinite variable and is thus also an infinite variable. We can scalarize this variable by using the measure $M_t f := \int_{t\in\mathcal{D}_t} f(t)w(t)dt$ with a weighting function satisfying $w : \mathcal{D}_t \to [0, 1]$ and $\int_{t\in\mathcal{D}_t} w(t)dt = 1$ (note that this measure is a time-average of the objective trajectory).

In space-time PDE optimization, the objective is defined over an infinite domain $\mathcal{D} = \mathcal{D}_t \times \mathcal{D}_x$ that depends on decision variables $y(t, x) \in \mathcal{Y}$; as such, the objective is given by the field $f(t, x) := f(y(t, x), t, x),\ t \in \mathcal{D}_t, x \in \mathcal{D}_x$. One can scalarize this field by using a measure:

$$M_{t,x} f = \int_{(t,x)\in\mathcal{D}_{t,x}} f(t, x)w(t, x)dtdx, \tag{2.11}$$

with weighting function satisfying $w : \mathcal{D}_{t,x} \to [0, 1]$ and $\int_{(t,x)\in\mathcal{D}_{t,x}} w(t, x)dtdx = 1$. One can think of this measure as a space-time average of the objective.

In stochastic optimization, we have infinite-dimensional objectives of the form $f(\xi) := f(z, y(\xi)),\ \xi \in \mathcal{D}_\xi$, where $y(\xi)$ is a recourse variable (an infinite variable). Scalarization can be achieved by using the expectation operator:

$$M_\xi f = \mathbb{E}_\xi[f(\xi)] = \int_{\xi\in\mathcal{D}_\xi} f(\xi)p(\xi)d\xi \tag{2.12}$$

where $p(\xi)$ is the probability density function satisfying $p(\xi) \geq 0$ and $\int_{\xi \in \mathcal{D}_\xi} p(\xi) = 1$. This measure is illustrated in Figure 2.9.

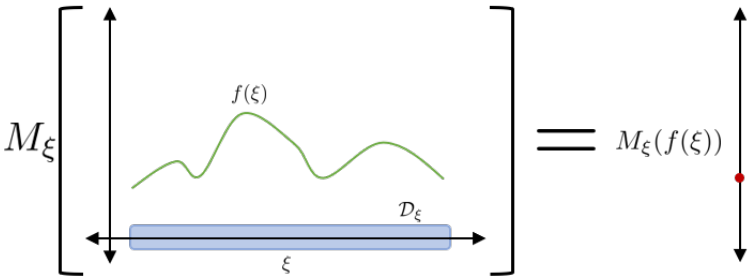



Figure 2.9: Depiction of measure operator $M_\xi$ acting on infinite variable $f(\xi) = f(z, y(\xi))$.

Average measures as those described previously are intuitive and widely used in practice; however, in Chapter 3 we will see that one can envision using a huge number of measures to perform scalarization. The huge number of choices arises from the fact that one can manipulate the shape of an infinite-dimensional function in many different ways (by focusing on different features of the function); for instance, one can minimize the peak of the function or minimize its variability. In the field of stochastic optimization, for instance, one aims to manipulate the shape of the infinite-dimensional objective by selecting different risk measures (summarizing statistics) such as the variance, median, quantile, worst/best -case value, or probabilities. We will see that one can borrow risk measures used in stochastic optimization to summarize infinite variables in other domains (e.g., space-time); this leads to interesting approaches to shape complex manifolds/fields arising in complex InfiniteOpt problems.

### 2.2.6 *Constraints*

As in the case of objectives, constraints in InfiniteOpt problems depend on infinite variables and are thus infinite variables themselves. One thus need to use specialized techniques to handle constraints and with this ensure that the problem is well-posed. A key observation that arises in this context is that constraints are treated differently than objectives; specifically, one typically seeks to impose bounds on constraint values and

one can handle collections of constraints simultaneously. For instance, in semi-infinite optimization problems, one enforces constraints of the form:

$$g_j(y(d), d) \leq 0,\ j \in \mathcal{J}, d \in \mathcal{D}. \tag{2.13}$$

In vector form, this collection of constraints can be expressed as:

$$g(y(d), d) \leq 0,\ d \in \mathcal{D}. \tag{2.14}$$

where $g(\cdot)$ is a vector function that contains the constraint collection $g_j(\cdot)\, j \in \mathcal{J}$. We can see that the constraint functions $g(\cdot)$ are required to a take value below zero *for all* values of the parameter $d \in \mathcal{D}$. Moreover, we can see that the constraints $j \in \mathcal{J}$ are all enforced at once. Figure 2.10 illustrates this constraint system. This particular approach to enforcing constraints is also widely used in dynamic optimization and stochastic optimization. For instance, in the context of dynamic optimization, one may seek to keep time trajectories for controls/states below a certain threshold value for all times in a time horizon. In the context of stochastic optimization, one may seek to satisfy the demand of a product for all possible realizations of uncertainty (in this context the constraints are said to be enforced *almost surely* or with probability of one).

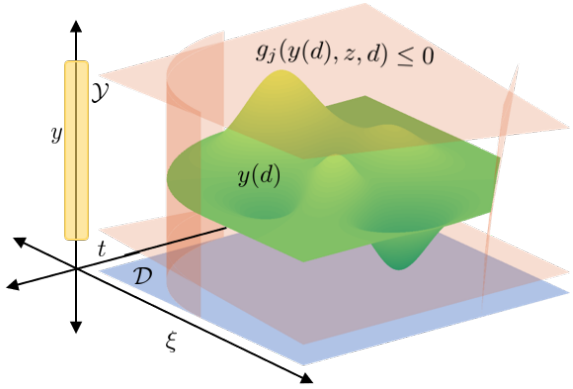



Figure 2.10: Depiction of infinite-dimensional constraints $g_j(y(d), z, d) \leq 0$ defined over an infinite domain $\mathcal{D}$.

These types of constraints are defined in our abstraction using the general form:

$$g(Dy, y(d), z, d) \leq 0,\ d \in \mathcal{D}. \tag{2.15}$$

These encapsulate the above use cases and are exemplified by the following PDE optimization constraints that include differential operators, path constraints, and point constraints (e.g., boundary conditions):

$$\begin{gathered} g\left(Dy(t,x), y(t,x), t, x\right) = 0,\ (t,x) \in \mathcal{D}_t \times \mathcal{D}_x \\ g\left(y(t,x), t, x\right) \leq 0,\ (t,x) \in \mathcal{D}_t \times \mathcal{D}_x \\ g\left(y(\hat{t},\hat{x}), \hat{t}, \hat{x}\right) \leq 0. \end{gathered} \tag{2.16}$$

Constraints that follow the form of (2.15) can be quite restrictive for certain applications, since they need to hold for every value parameter $d \in \mathcal{D}$. One can relax this requirement by instead enforcing the constraint on a selected set of points in the domain $\mathcal{D}$ or by enforcing constraints on a measure of the constraint functions. For instance, consider the set of constraint functions $h_k(Dy(d), y(d), z, d),\ k \in \mathcal{K}$; we can aim to enforce constraints on expected values of such functions as:

$$\int_{\xi \in \mathcal{D}_\xi} h_k(y(\xi), \xi) p(\xi) d\xi \geq 0,\ k \in \mathcal{K}. \tag{2.17}$$

Given that there are a wide range of measures that can help shape functions over infinite-dimensional domains, one can also envision different approaches to enforce constraints. For instance, in stochastic optimization, one typically uses scalar chance (probabilistic) constraints of the form:

$$\mathbb{P}_\xi \left(h_k(y(\xi), \xi) \leq 0\right) \geq \alpha,\ k \in \mathcal{K}. \tag{2.18}$$

This set of constraints require that each constraint function $h_k(\cdot)$ is kept below zero to a certain probability level $\alpha$. In stochastic optimization, one also often enforces joint chance

constraints:

$$\mathbb{P}_{\xi}\left(h_k(y(\xi),\xi) \leq 0,\ k \in \mathcal{K}\right) \geq \alpha. \tag{2.19}$$

The joint chance constraint can also be expressed in vector form as:

$$\mathbb{P}_{\xi}\left(h(y(\xi),\xi) \leq 0\right) \geq \alpha. \tag{2.20}$$

Joint chance constraints require that the constraint functions $h(\cdot)$ are kept (jointly) below zero with a certain probability level $\alpha$. We will see that joint chance constraints allow us to enforce constraints on probability of *events* and we will see that this provides a flexible modeling construct to capture complex decision-making logic. For instance, we might want to ensure that the temperature of a system is higher than a certain value *and* that the concentration of the system is lower than a certain value with a given probability. Joint chance constraints can also be interpreted as a generalization of other constraint types; for instance, if we set $\alpha = 1$, the constraint (2.19) is equivalent to (2.13).

The above measure constraints can be expressed in the following general form:

$$Mh(Dy, y(d), z, d) \geq 0. \tag{2.21}$$

For instance, the chance constraint (2.20) can be expressed as:

$$M_{\xi}h \geq 0 \tag{2.22}$$

with

$$M_{\xi}h = \mathbb{E}_{\xi}\left[\mathbb{1}[h(y(\xi),\xi) \leq 0]\right] - \alpha. \tag{2.23}$$

### 2.2.7 *Formulation*

We summarize the previous elements to express the InfiniteOpt problem in the following abstract form:

$$\begin{aligned} \min_{y(d)\in\mathcal{Y}, z\in\mathcal{Z}} \quad & Mf\left(Dy, y(d), z, d\right) \\ \text{s.t.} \quad & g\left(Dy, y(d), z, d\right) \leq 0,\ d \in \mathcal{D} \\ & Mh\left(Dy, y(d), z, d\right) \geq 0. \end{aligned} \tag{2.24}$$

This abstract form seeks to highlight the different elements of the proposed abstraction (e.g., infinite domains and variables, finite variables, measure operators, differential operators).

## 2.3 Transformations

We now discuss how InfiniteOpt problems are solved through the lens of the proposed unifying abstraction. Solution approaches typically rely on transforming the InfiniteOpt problem (2.24) into a finite-dimensional formulation that can be solved using conventional optimization solvers. There are numerous possible methods to transform InfiniteOpt problems that are used in different domains such as dynamic, PDE, and stochastic optimization. Our goal here is not to provide an extensive discussion and implementation of all these approaches; instead, we highlight common elements of different approaches and motivate how these can be facilitated by using a unifying abstraction.

### 2.3.1 *Direct Transcription*

Our first goal is to obtain a finite representation of an infinite domain $\mathcal{D}_\ell$; direct transcription accomplishes this via a finite set of support points that we represent as:

$$\hat{\mathcal{D}}_\ell = \{\hat{d}_{\ell,i} : \hat{d}_{\ell,i} \in \mathcal{D}_\ell, i \in \mathcal{I}_\ell\}. \tag{2.25}$$

The concept of the support set $\hat{\mathcal{D}}_\ell$ used here is general and a variety of methods can be employed to generate it. In stochastic optimization, for instance, a set of MC samples is typically drawn from a probability density function of the infinite parameters [6], while PDE problems commonly use quadrature schemes [58]. The proposed abstraction seeks to enable porting techniques across fields; for instance, one might envision generating support points for a space-time domain by sampling or one might envision generating support points for a random domain by using quadrature points (as done in sparse grids and Latin hypercube sampling).

The support set for the InfiniteOpt problem $\hat{\mathcal{D}}$ is the Cartesian product of the individual supports sets:

$$\hat{\mathcal{D}} := \prod_{\ell \in \mathcal{L}} \hat{\mathcal{D}}_\ell. \tag{2.26}$$

Figure 2.11 illustrates how the support set (a finite domain) approximates the infinite domain $\mathcal{D}$. Note that this definition of $\hat{\mathcal{D}}$ assumes that the individual domains $\mathcal{D}_\ell$ are independent of one another. This assumption does not hold in some complex applications; for instance, in stochastic dynamic optimization problems, we might have random parameters that are functions of time (this is discussed further in Chapter 4).

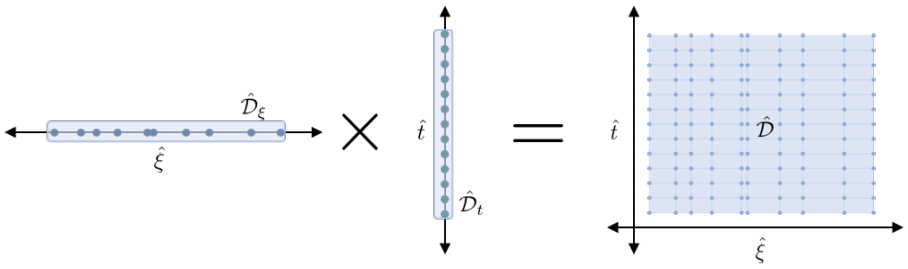



Figure 2.11: Finite support set $\hat{\mathcal{D}}$ that approximates infinite domain $\mathcal{D}$.

The infinite-dimensional formulation (2.24) is projected onto the finite support set to yield a finite-dimensional approximation that can be modeled using conventional optimization solvers. We now proceed to discuss how this projection is achieved. Measures are approximated with an appropriate numerical scheme; this can take on a range of forms and may require the incorporation of additional supports and variables. For in-

stance, a common set of measures (e.g., expectations and integrals over space-time domains) are of the form:

$$M_\ell y = \int_{d' \in \mathcal{D}_\ell} y(d') w(d') dd' \tag{2.27}$$

where $w(\cdot)$ is a weighting function. Such measures can be approximated using support points as:

$$M_\ell y \approx \sum_{i \in \mathcal{I}_\ell} \beta_i y(\hat{d}_{\ell,i}) w(\hat{d}_{\ell,i}). \tag{2.28}$$

This general form is used in quadrature and sampling schemes; the only difference between these schemes arises in how the supports $\hat{d}_{\ell,i}$ and the coefficients $\beta_i$ are selected. Figure 2.12 depicts a measure approximated via quadrature.

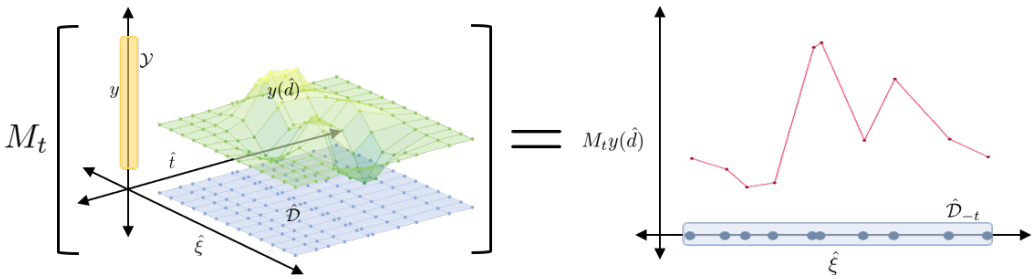



Figure 2.12: Depiction of a measure operator $M_t$ approximated via a numerical scheme (quadrature in this case).

Differential operators appearing in formulation (2.24) also need to be approximated. Sometimes these operators can be reformulated in integral form; in such a case, one can use the measure approximations previously discussed. However, in some cases, this reformulation is not possible; for instance, a differential operator might be implicitly defined within expression functions (e.g., boundary conditions) and/or within measures. In our framework, we treat differential operators as infinite variables and handle them via *lifting*. To illustrate how this is done, suppose that we have an expression of the form:

$$g\left(\frac{\partial y(d)}{\partial d_\ell}, y(d), z\right) = 0, \; d \in \mathcal{D}_\ell. \tag{2.29}$$

Here, we introduce an auxiliary variable $y'(d)$ and reformulate the above constraint as:

$$g(y'(d), y(d), z) = 0,\ d \in \mathcal{D}_\ell \tag{2.30a}$$

$$\frac{\partial y(d)}{\partial d_\ell} = y'(d),\ d \in \mathcal{D}_\ell. \tag{2.30b}$$

The second expression can now be approximated using traditional schemes using support points; for instance, when $d$ denotes time (e.g., in a dynamic optimization problem), one typically uses a backward finite difference:

$$y(\hat{d}_{\ell,i}) = y(\hat{d}_{\ell,i-1}) + (\hat{d}_{\ell,i} - \hat{d}_{\ell,i-1}) y'(d_{\ell,i}). \tag{2.31}$$

Figure 2.13 illustrates how these techniques approximate differential operators. A lifting approach can be used to handle higher-order and multi-dimensional operators (e.g., Laplacian) via nested recursions. These basic constructs can be used to enable the implementation of direct transcription schemes such as MC sampling, quadrature, finite difference, and orthogonal collocation over finite elements.

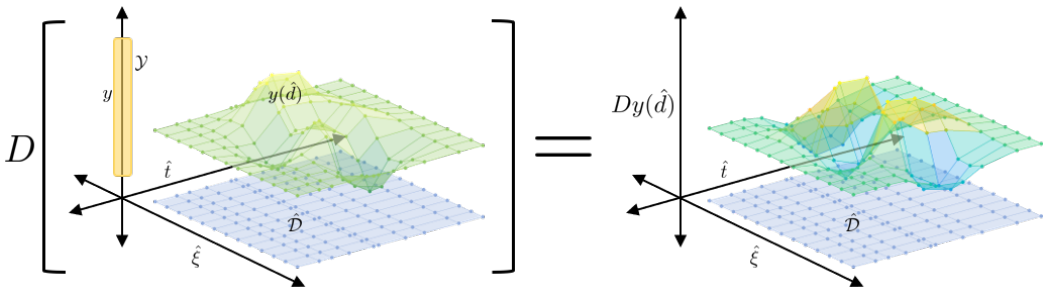



Figure 2.13: Depiction of a differential operator $D$ approximated via a numerical scheme (central finite differences in this case) relative to a realization of infinite variable $y(d)$.

Once the measures and derivatives are approximated, the direct transcription procedure is finalized by projecting the remaining constraints with infinite domain dependencies over the finite support set $\hat{\mathcal{D}}$. The transformation incurred by direct transcription is often linear since the typical measure and differential operator approximations are linear transformations of the respective modeling objects (e.g., MC sampling and finite differ-

ence). For instance, this means that if the InfiniteOpt problem of interest is an infinite quadratic program (QP), then its transcribed variant will typically be a finite QP.

We note that direct transcription of problems with multiple infinite domains (e.g., PDE-constrained problems) can incur tractability concerns due to the support combinatorics. For such problems, decomposition approaches such as the one proposed in [59] can be used address these limitations. Moreover, alternative transformations may enhance tractability for certain problem classes as is discussed in Section 2.3.2.

### 2.3.2 *Alternative Transformations*

Direct transcription is a common class of methods for transforming an InfiniteOpt problem into a finite-dimensional representation by using a finite set of support points. A limitation of this approach is that it does not provide a solution in functional form (it only provides a solution defined at the support points). Alternative transformation methods can be envisioned to deliver solutions in functional form. The method of weighted residuals (MWR) is a general class of methods that conducts the transformation by approximating the problem elements using basis expansions. Popular MWR techniques include polynomial chaos expansion (PCE) used in stochastic optimization [60] and orthogonal collocation used in dynamic optimization [61, 27]. For instance, Gnegel et. al. recently demonstrated how such basis expansion techniques can enhance the tractability of mixed-integer PDE problems relative to using traditional transcription methods [62]. Such techniques are often behind what are typically referred to as order reduction methods in the PDE community [61].

In MWR, a set of trial/basis functions $\Phi = \{\phi_i(d) : i \in \mathcal{I}\}$ is defined over an infinite domain $\mathcal{D}$ and linear combinations of these functions are used to approximate the infinite variables:

$$y(d) \approx \sum_{i \in \mathcal{I}} \tilde{y}_i \phi_i(d) \tag{2.32}$$

where $\tilde{y}_i \in \mathbb{R}$ are the basis function coefficients. An illustration of this approximation

is given in Figure 2.14; here, we require that the basis functions $\phi_i(d)$ and the infinite variables $y(d)$ both reside in a common space such that this approximation becomes exact when the set $\Phi$ is an orthogonal set of basis functions and $|\Phi| \to \infty$ [63]. Since the basis functions are known, this representation allows us to represent the infinite variables $y(d)$ in terms of the coefficients $\tilde{y}_i$ (which are finite variables). As such, this approach effectively transforms infinite variables into finite variables. The goal is now to project the formulation (2.24) onto a set of basis functions so as to obtain a finite formulation that solely depends on the finite variables $\tilde{y}_i$ and $z$. This is done by expressing differential and measure operators by using the basis expansion of the infinite variables (i.e., with operators applied to the basis functions). In certain cases, the expansion coefficients can be useful in evaluating certain measure types; for example, the coefficients will correspond to the statistical moments of the infinite variables when PCE is applied to a stochastic formulation with a basis that is orthogonal to the probability density function and these moments are often used to express expectations and risk measures [64].

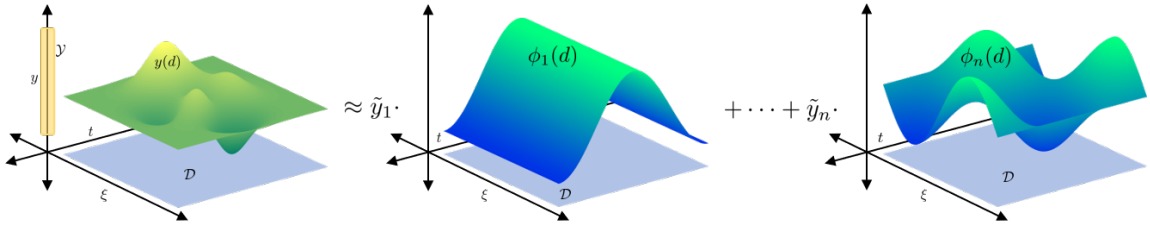



Figure 2.14: Depiction of how an infinite variable $y(d)$ can be approximated as a linear combination of basis functions $\phi_i(\cdot)$.

After basis expansion representations are incorporated, the problem is fully defined in terms of the finite-dimensional variables $\tilde{y}_i$ and $z$. However, this representation is not yet tractable, since it still contains infinite-dimensional objects (e.g., basis functions and associated operators). To deal with this, we consider the residual (i.e., finite error) $R(d)$ associated with performing this projection on each constraint and on the objective. Each resulting residual will be made as small as possible by exacting that they be orthogonal

to a set of weight functions $\psi_k(d),\ k \in \mathcal{K}, d \in \mathcal{D}$:

$$\langle R, \psi_k \rangle_w = 0,\ \forall k \in \mathcal{K} \tag{2.33}$$

where $\langle \cdot, \cdot \rangle_w$ denotes the inner product between functions using the appropriate weighting function $w(d)$ for the given space:

$$\langle R, \psi_k \rangle_w = \int_{d' \in \mathcal{D}} R(d')\psi_k(d')w(d')dd'. \tag{2.34}$$

The weight functions are typically chosen such that $|\mathcal{K}| = |\mathcal{I}|$. The projection results in a tractable finite-dimensional formulation; what remains, is our choice of the weight functions $\psi_k(\cdot)$. This choice gives rise to a range of techniques; if the Galerkin method is applied then we choose $\phi_k(d) = \psi_k(d)$ and have that $\mathcal{I} = \mathcal{K}$. This induces the first $|\mathcal{I}|$ terms of the residuals in the trial functions to vanish if the functions are orthogonal [63]. Another popular choice is that of orthogonal collocation, where we choose $\psi_k(d) = \delta(d - \hat{d}_k)$; here, the set $\hat{d}_k,\ k \in \mathcal{K}$ denote collocation points (i.e., particular infinite parameter supports) and $\delta(\cdot)$ is the Dirac delta function. This approach seeks to enforce that the residual is zero at the collocation points. When orthogonal basis functions are chosen and this is applied over a set of transcription points (i.e., finite elements), we obtain a method known as orthogonal collocation over finite elements. A variety of other methods such as least squares and the method of moments can also be employed to weight the residuals (these are discussed in detail in [65]).

The transformation of (2.24) to a finite-dimensional form via MWR is, in general, a nonlinear transformation (depending on the choices of the trial functions $\phi_i(\cdot)$, weight functions $\psi_k(\cdot)$, and their corresponding space). However, there exist special cases where the transformation is linear, as is often the case with PCE transformations [66]. Advantages of employing MWR instead of direct transcription is that one obtains functional representations for the infinite variables (as opposed to values at the support points), one can achieve better stability for boundary-valued problems, and one can obtain bet-

ter accuracy for certain formulations [2]. On the other hand, the main disadvantage of MWR is that evaluating differential and measure operators and inner products tends to be cumbersome (especially for nonlinear formulations). Also, basis functions can be difficult to derive for formulations with multivariate infinite domains. In our abstraction, we provide the modeling elements that facilitate the implementation of these transformation techniques.

## 2.4 Software Implementation in InfiniteOpt.jl

In this section, we describe how the proposed abstraction can facilitate the development of modeling tools. Specifically, the proposed abstraction is used as the backbone of a modeling package that we call `InfiniteOpt.jl` (https://github.com/zavalab/InfiniteOpt.jl). `InfiniteOpt.jl` is written in the `Julia` programming language [67].

### 2.4.1 *Modeling*

`InfiniteOpt.jl` builds upon the capabilities of `JuMP.jl` [68] to intuitively and compactly express InfiniteOpt problems. Some modeling features of `InfiniteOpt.jl` are illustrated by using the example problem:

$$
\begin{aligned}
\min_{y_a(t), y_b(t,\xi), y_c(\xi), z} \quad & \int_{t\in\mathcal{D}_t} y_a(t)^2 + 2\mathbb{E}_{\xi}[y_b(t,\xi)]dt && && \text{(2.35a)}\\
\text{s.t.} \quad & \frac{\partial y_b(t,\xi)}{\partial t} = y_b(t,\xi)^2 + y_a(t) - z_1, && \forall t\in\mathcal{D}_t,\ \xi\in\mathcal{D}_{\xi} && \text{(2.35b)}\\
& y_b(t,\xi) \le y_c(\xi)U, && \forall t\in\mathcal{D}_t && \text{(2.35c)}\\
& \mathbb{E}_{\xi}[y_c(\xi)] \ge \alpha && && \text{(2.35d)}\\
& y_a(0) + z_2 = \beta && && \text{(2.35e)}\\
& y_a(t), y_b(t,\xi) \in \mathbb{R}_+, y_c(\xi)\in\{0,1\}, z\in\mathbb{Z}^2, && \forall t\in\mathcal{D}_t,\ \xi\in\mathcal{D}_{\xi} && \text{(2.35f)}
\end{aligned}
$$

Here, $y_a(t)$, $y_b(t,\xi)$, and $y_c(\xi)$ are infinite variables, $z$ are finite variables, $U, \alpha, \beta \in \mathbb{R}$ are constants, $\mathcal{D}_t = [t_0, t_f]$ is the time domain, and $\mathcal{D}_\xi$ is the co-domain of the random parameter $\mathcal{N}(\mu, \Sigma)$.

**Code Snippet 2.1: Modeling problem (2.35) using `InfiniteOpt.jl`.**

```
1  using InfiniteOpt, Distributions, KNITRO
2
3  # Initialize the model
4  model = InfiniteModel(KNITRO.Optimizer)
5
6  # Add the infinite parameters corresponding to the infinite domains
7  @infinite_parameter(model, t ∈ [t0, tf], num_supports = 100)
8  @infinite_parameter(model, ξ[1:10] ~ MvNormal(μ, Σ), num_supports = 100)
9
10 # Add the variables and their domain constraints
11 @variable(model, 0 ≤ ya, Infinite(t))
12 @variable(model, 0 ≤ yb, Infinite(t, ξ))
13 @variable(model, yc, Infinite(ξ), Bin)
14 @variable(model, z[1:2], Int)
15
16 # Define the objective
17 @objective(model, Min, ∫(ya ^ 2 + 2 * 𝔼(yb, ξ), t))
18
19 # Add the constraints
20 @constraint(model, ∂(yb, t) == yb ^ 2 + ya - z[1])
21 @constraint(model, yb ≤ yc * U)
22 @constraint(model, 𝔼(yc, ξ) ≥ α)
23 @constraint(model, ya(0) + z[2] == β)
24
25 # Solve and retrieve the results
26 optimize!(model)
27 opt_objective = objective_value(model)
```

The corresponding `InfiniteOpt.jl` syntax for expressing this problem is shown in Code Snippet 2.1. An InfiniteOpt problem is stored in an `InfiniteModel` object; line 4 shows the initialization of the model object `model`. The model is automatically transcribed into a finite dimensional representation and solved using the `KNITRO` solver [69]. More information on how the `InfiniteModel` is transcribed by `InfiniteOpt.jl` is provided in Section 2.4.2. Lines 7 and 8 use `@infinite_parameter` to define the infinite parameters with their respective infinite domains and indicate that each domain should use 100 finite supports in the transcription. The random parameters $\xi$ can be associated with any probably density function supported by the Julia package `Distributions.jl` [70]. Lines 11-14 define the decision variables and their associated properties in accordance with Equation (2.35f) following a symbolic `JuMP.jl`-like syntax by means of `@variable`. Line 17 defines the complex objective depicted in Equation (2.35a) via `@objective`. Lines 20-23 define constraints (2.35b)-(2.35e) using `@constraint`. Notice how the differential operator

and measure operators (in this case an expectation and an integral) are easily incorporated using `Julia` syntax. Lines 26-27 illustrate how the model `model` is solved using `optimize!` and then how the solution information can be extracted from the model.

### 2.4.2 *Transformations*

We have discussed how our unifying abstraction is implemented in `InfiniteOpt.jl` (one creates a model as an `InfiniteModel` object), here we describe the general transformation framework incorporated into `InfiniteOpt.jl` that facilitates the implementation of different transformation approaches (e.g., direct transcription and MWRs). We also outline the efficient direct transcription capabilities that are currently implemented.

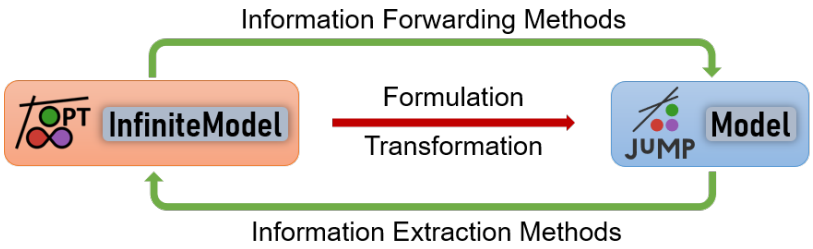



Figure 2.15: Transformation framework employed by `InfiniteOpt.jl` for converting an `InfiniteModel` into a `JuMP.jl` `Model`.

The framework centers around applying a transformation to the `InfiniteModel` that converts it to a standard `JuMP.jl` `Model` object (referred to as an optimizer model in this context). The optimizer model can then be solved using the optimizers implemented in `MathOptInterface.jl` [71]. Moreover, this framework features a collection of methods to enable a seamless interface between the `InfiniteModel` and its corresponding optimizer model to facilitate capabilities such as information extraction (e.g., solution queries) that do not require direct interrogation of the optimizer model. This framework is summarized in Figure 2.15. This software structure distinguishes `InfiniteOpt.jl` from other software tools (e.g., `Pyomo.dae` and `Gekko`) whose implementations center around (and are limited to) direct transcription. Thus, `InfiniteOpt.jl` is unique in providing a flexi-

ble API for solving InfiniteOpt problems.

Following this framework, a desired solution scheme is incorporated by defining a few prescribed methods (principally `build_optimizer_model!`) to implement the associated transformation. This methodology is implicitly invoked on line 26 of Code Snippet 2.1 where `optimize!` creates an optimizer model using the prescribed transformation and then solves it with the desired optimizer. The full technical detail of this API is beyond the scope of this work and is available via the `InfiniteOpt.jl` documentation.

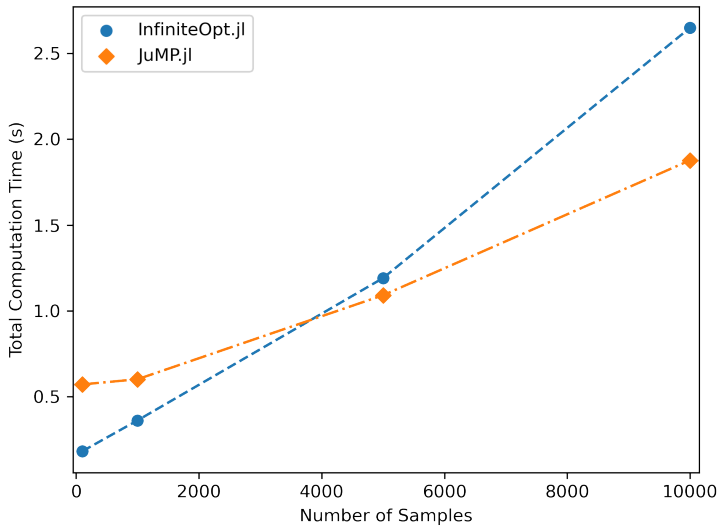



Figure 2.16: Juxtaposition of the total computation time used to formulate and solve a stochastic optimization problem [1] using MC sampling implemented in `InfiniteOpt.jl v0.4.1` and manual transcription in `JuMP.jl v0.21.8`.

`InfiniteOpt.jl` provides an efficient implementation of direct transcription following the procedures described in Section 2.3; this serves as the default transformation technique for InfiniteOpt models. These techniques are implemented in a sub-module called `TranscriptionOpt` that follows the optimizer model framework shown in Figure 2.15. The `TranscriptionOpt` module features a sophisticated finite support generation and management system that enables tackling a wide variety of infinite-dimensional optimization formulations using diverse evaluation techniques for measure and derivative

operators. Moreover, its automatic transcription is efficient and compares competitively to manually transcribing a problem and implementing it via `JuMP.jl`. This incredible behavior is demonstrated in Figure 2.16 where a two-stage stochastic optimization problem (the 3-node distribution network example featured in [1]) is solved for a range of MC samples using automatic transcription in `InfiniteOpt.jl` and manual transcription in `JuMP.jl`. We note that, contrary to other software implementations, automatic transcription in `InfiniteOpt.jl` denotes a minor computational expense relative to manual transcription with the benefit of avoiding the errors commonly incurred by transcribing InfiniteOpt formulations manually.

## 2.5 Case Studies

In this section, we provide illustrative case studies to demonstrate the concepts discussed. These case studies seek to exemplify how the unifying abstraction captures a wide range of formulation classes. These cases are implemented via `InfiniteOpt.jl v0.5.1` using `Ipopt v3.13.2` for continuous problems and `Gurobi v9.1.1` for integer-valued problems.

### 2.5.1 *Spatial-Temporal Control of Atomic Layer Deposition*

Our unifying abstraction readily incorporates PDE-constrained optimization problems. We demonstrate this using a problem that seeks optimal conditions for an atomic layer deposition process. In particular, we examine the atomic layer deposition (ALD) process on a porous substrate surface that is presented in [72]. This amounts to a 1D reactive diffusion system where a substrate is exposed to a gaseous precursor that diffuses and reacts with it (this process is taken to be isotropic across the surface). The ALD model considers the precursor density $y_p(t, x) \in \mathbb{R}_+$ in relation to the fraction of available sites in the substrate surface layer $y_\theta(t, x) \in [0, 1]$. Hence, the surface coverage is given by $1 - y_\theta(t, x)$. Following the derivation presented in [72], we obtain the PDE modeling

equations:

$$\begin{aligned} \frac{\partial y_p(t,x)}{\partial t} &= \kappa \frac{\partial^2 y_p(t,x)}{\partial x^2} - \gamma \xi y_p(t,x) y_\theta(t,x), \quad (t,x) \in \mathcal{D}_{t,x} \\ \frac{\partial y_\theta(t,x)}{\partial t} &= -\eta \xi y_p(t,x) y_\theta(t,x), \qquad\qquad (t,x) \in \mathcal{D}_{t,x} \end{aligned} \tag{2.36}$$

where $\mathcal{D}_{t,x} = \mathcal{D}_t \times \mathcal{D}_x = [0, 10] \times [0, 500]$, $\kappa = 2.81 \times 10^6\ \mu m^2 s^{-1}$ is the diffusivity constant, $\gamma = 6.912 \times 10^8\ s^{-1}$ is the precursor species reaction constant, $\eta = 1.538 \times 10^7\ \mu m^3 s^{-1}$ is the site reaction constant, and $\xi = 2 \times 10^{-4}$ is the reaction probability. For boundary conditions we enforce:

$$\begin{aligned} y_p(0,x) &= 0, && x \in \mathcal{D}_x \\ y_p(t,0) &= z_p, && t \in \mathcal{D}_{t>0} \\ \frac{\partial y_p(t,x)}{\partial x}\Big|_{x=500} &= 0, && t \in \mathcal{D}_t \\ y_\theta(0,x) &= 1, && x \in \mathcal{D}_x \end{aligned} \tag{2.37}$$

which enforce that the substrate not have any precursor initially, the ambient gaseous density be constant at $z_p \in \mathbb{R}$, no diffusion occurs beyond a 500 $\mu m$ depth, and all the sites are initially available, respectively.

We seek to optimally choose $z_p$ such that the final coverage profile $1 - y_\theta(10, x)$ follows a desired setpoint $1 - \bar{y}_\theta(x)$. Here the setpoint function is defined via a modified inverse sigmoid function that resembles the desired profile presented in [72].

$$\bar{y}_\theta(x) = 1 - \frac{1}{1 + \exp\left(x - \frac{\rho_1}{2}\right)^{\rho_2}} \tag{2.38}$$

where $\rho_1 = 500$ and $\rho_2 = 0.15$. The objective seeks to minimize the expected setpoint tracking error at the final time:

$$\min \int_{x \in \mathcal{D}_x} (y_\theta(10, x) - \bar{y}_\theta(x))^2 dx. \tag{2.39}$$

Putting together (2.36), (2.37), and (2.39) we obtain:

$$
\begin{aligned}
\min \quad & \int_{x\in\mathcal{D}_x} (y_\theta(10,x) - \bar{y}_\theta(x))^2 dx \\
\text{s.t.} \quad & \frac{\partial y_p(t,x)}{\partial t} = \kappa \frac{\partial^2 y_p(t,x)}{\partial x^2} - \gamma\xi y_p(t,x) y_\theta(t,x), && (t,x) \in \mathcal{D}_{t,x} \\
& \frac{\partial y_\theta(t,x)}{\partial t} = -\eta\xi y_p(t,x) y_\theta(t,x), && (t,x) \in \mathcal{D}_{t,x} \\
& y_p(0,x) = 0, && x \in \mathcal{D}_x \\
& y_p(t,0) = z_p, && t \in \mathcal{D}_{t>0} \\
& \frac{\partial y_p(t,x)}{\partial x}\Big|_{x=500} = 0, && t \in \mathcal{D}_t \\
& y_\theta(0,x) = 1, && x \in \mathcal{D}_x.
\end{aligned}
\tag{2.40}
$$

This is implemented in `InfiniteOpt.jl` using direct transcription over 100 spatial support points and 10 temporal support points; Code Snippet 2.2 shows the implementation.

**Code Snippet 2.2: Formulation (2.40) via `InfiniteOpt.jl`.**

```julia
using InfiniteOpt, Ipopt

# Setup the model
model = InfiniteModel(Ipopt.Optimizer)

# Define the infinite parameters
@infinite_parameter(model, t ∈ [0, 10], num_supports = 10)
@infinite_parameter(model, x ∈ [0, L], num_supports = 100)

# Define the variables
@variable(model, 0 <= yθ <= 1, Infinite(t, x), start = 1)
@variable(model, 0 <= yp, Infinite(t, x))
@variable(model, 0 <= zp <= 0.1, start = 0.01)

# Set the objective
@objective(model, Min, ∫((yθ(10, x) + (1 / (1 + exp(x - 250) ^ 0.15)) - 1) ^ 2, x))

# Define the equations
@constraint(model, ∂(yp, t) == κ * @∂(yp, x ^ 2) - γ * ξ * yp * yθ)
@constraint(model, ∂(yθ, t) == -η * ξ * yp * yθ)
@constraint(model, yp(0, x) == 0)
@constraint(model, yp(t, 0) == zp, DomainRestrictions(t => [supports(t)[2], 10]))
@constraint(model, ∂(yp, t)(t, L) == 0)
@constraint(model, yθ(0, x) == 1)

# Solve and extract the results
optimize!(model)
yθs = value(yθ, ndarray = true)
zp = value(zp)
xs = value(x)
```

Figure 2.17 shows the optimal coverage profile $1 - y^*_\theta(0,x)$. This is obtained with an optimal choice of $z_p = 0.048$. We observe that the setpoint concentration profile is closely

matched at this precursor density. It is not a perfect fit however, since we cannot readily change the slope of the curve by only controlling the precursor density. In Chapter 4, we will revisit this example while considering a random reaction probability $\xi(x)$ that propagates over position $x$ using random field theory.

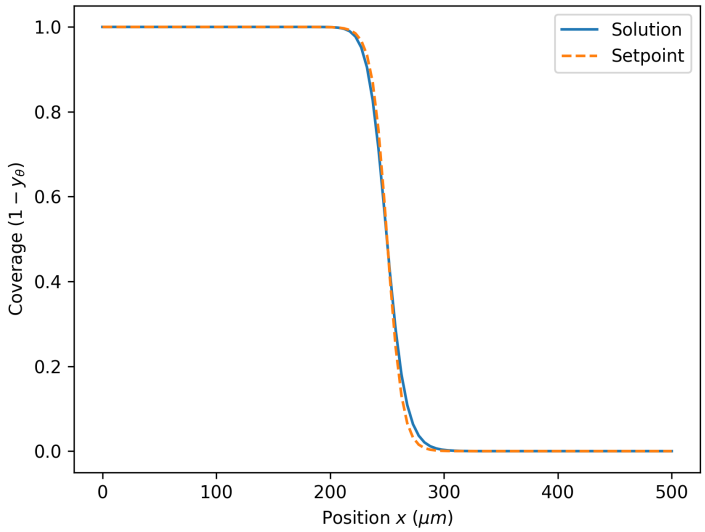



Figure 2.17: The optimal coverage profile across the substrate relative to the desired setpoint. This is set at $t = 10$.

### 2.5.2 *Optimal Flexibility Design*

Here we showcase how stochastic joint-chance constrained problems are a natural fit for our measure-centric unifying abstraction. Specifically, we consider an optimal stochastic flexibility index design problem (which is the focus of Chapter 7). This gives us a preview of how the flexibility and reliability analysis approaches discussed in Part II of this dissertation can readily be cast as general InfiniteOpt problems in our modeling abstraction.

An optimal $SF$-index design problem can be posed as a joint-chance constrained formulation the seeks to minimize the design cost while a minimally allowable $SF$-index is

enforced:

$$\begin{aligned} \min \quad & f(z) \\ \text{s.t.} \quad & \mathbb{P}_{\xi}\left(g(z, y(\xi), \xi) \leq 0\right) \geq \alpha \\ & z \in \mathcal{Z} \end{aligned} \tag{2.41}$$

where $z$ constitute the design variables, $y(\xi)$ are the system (recourse) variables, and $\alpha \in [0,1]$ is the desired probability level threshold [1]. This problem features two competing priorities: minimizing the design cost and maximizing the $SF$-index; this inherent tradeoff behavior is controlled by our choice of $\alpha$. We can reformulate the joint-chance constraint via a big-$M$ constraint reformulation that introduces binary variables $y_r(\xi) \in \{0,1\}$ [57]:

$$\begin{aligned} \min \quad & f(z) \\ \text{s.t.} \quad & g(z, y(\xi), \xi) \leq y_r(\xi) M, \quad \xi \in \mathcal{D}_{\xi} \\ & \mathbb{E}_{\xi}[1 - y_r(\xi)] \geq \alpha \\ & z \in \mathcal{Z} \end{aligned} \tag{2.42}$$

where $M \in \mathbb{R}_+$ is a sufficiently large upper-bounding constant.

We consider a distribution network that is modeled by performing balances at each node $n \in \mathcal{C}$ and enforcing capacity constraints on the arcs $y_{a,l}, l \in \mathcal{A}$, and on the suppliers $y_{s,b}, b \in \mathcal{S}$. The demands $d_m, m \in \mathcal{R}$, are assumed to be the uncertain parameters which are given by the MC samples. The deterministic network model is given by:

$$\sum_{l \in \mathcal{A}_n^{rec}} y_{a,l} - \sum_{l \in \mathcal{A}_n^{snd}} y_{a,l} + \sum_{b \in \mathcal{S}_n} y_{s,b} - \sum_{m \in \mathcal{R}_n} d_m = 0, \quad n \in \mathcal{C} \tag{2.43a}$$

$$-a_l^C - z_{a,l} \leq y_{a,l} \leq a_l^C + z_{a,l}, \quad l \in \mathcal{A} \tag{2.43b}$$

$$0 \leq y_{s,b} \leq s_b^C + z_{s,b}, \quad b \in \mathcal{S} \tag{2.43c}$$

where $\mathcal{A}_n^{rec}$ denotes the set of receiving arcs at node $n$, $\mathcal{A}_n^{snd}$ denotes the set of sending arcs at $n$, $\mathcal{S}_n$ denotes the set of suppliers at $n$, $\mathcal{R}_n$ denotes the set of demands at $n$, $a_l^C$

are the arc capacities, $z_{a,l} \geq 0$ are design variables that increase arc capacity, $s_b^C$ are the supplier capacities, and $z_{s,b} \geq 0$ are design variables that increase supplier capacity.

To pose our problem we let the demand $d$ be random $\xi$ system conditions. We obtain our formulation by inserting Equations (2.43) into Problem (2.42) to obtain:

$$
\begin{aligned}
\min \quad & \sum_{l \in \mathcal{A}} z_{a,l} + \sum_{b \in \mathcal{S}} z_{s,b} \\
\text{s.t.} \quad & \sum_{l \in \mathcal{A}_n^{rec}} y_{a,l}(\xi) - \sum_{l \in \mathcal{A}_n^{snd}} y_{a,l}(\xi) + \sum_{b \in \mathcal{S}_n} y_{s,b}(\xi) - \sum_{m \in \mathcal{R}_n} \xi_m = 0, && n \in \mathcal{C}, \xi \in \mathcal{D}_\xi \\
& -a_l^C - z_{a,l} - y_{a,l}(\xi) \leq y_r(\xi) M, && l \in \mathcal{A}, \xi \in \mathcal{D}_\xi \\
& y_{a,l}(\xi) - a_l^C - z_{a,l} \leq y_r(\xi) M, && l \in \mathcal{A}, \xi \in \mathcal{D}_\xi \\
& y_{s,b}(\xi) - s_b^C - z_{s,b} \leq y_r(\xi) M, && b \in \mathcal{S}, \xi \in \mathcal{D}_\xi \\
& y_{s,b}(\xi) \geq 0, && \xi \in \mathcal{D}_\xi \\
& \mathbb{E}_\xi[1 - y_r(\xi)] \geq \alpha \\
& z_{s,b}, z_{a,l} \geq 0, && l \in \mathcal{A}, b \in \mathcal{S}.
\end{aligned}
\tag{2.44}
$$

We apply a simple three-node distribution network that features a centralized supplier configuration. Figure 2.18 details this network and provides the arc and supplier capacities. The network is subjected to multivariate Gaussian demands $\xi = (d_1, d_2, d_3) \sim \mathcal{N}(\mu, \Sigma)$. The mean $\mu$ is taken to be $\mu = (0.0, 60.0, 10.0)$, and the covariance matrix $\Sigma$ is:

$$
\Sigma = \begin{bmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 120 \end{bmatrix}. \tag{2.45}
$$

We also set the parameter $M = 10000$.

Under these conditions, we implement Problem (2.44) in `InfiniteOpt.jl` using 1,000 MC samples. Moreover, we obtain Pareto pairs by varying the value of $\alpha$ from 0.97 to 1 in increments of 0.001. Code Snippet 2.3 summarizes the implementation, omitting the constants and storage arrays for succinctness.

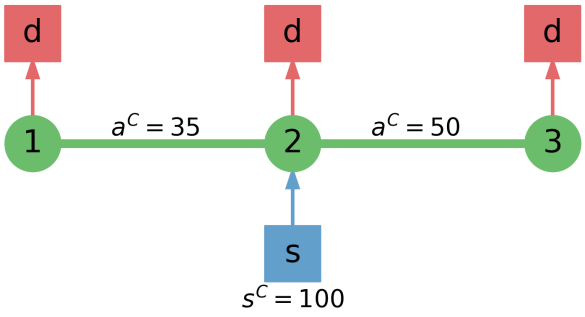



Figure 2.18: The three-node distribution network

**Code Snippet 2.3: Formulation (2.44) via `InfiniteOpt.jl`.**

```julia
using InfiniteOpt, Distributions, Gurobi

# Initialize the model
model = InfiniteModel(Gurobi.Optimizer); set_silent(model)

# Set the parameters
@infinite_parameter(model, ξ[1:3] ~ MvNormal(μ, Σ), num_supports = 1000)
@finite_parameter(model, α == 0.9)

# Initialize the variables
@variable(model, y[1:3], Infinite(ξ))
@variable(model, z[1:3] >= 0)
@variable(model, yr, Infinite(ξ), Bin)

# Set objective function
@objective(model, Min, sum(z))

# Set the node balance constraints
@constraint(model, y[1] - ξ[1] == 0)
@constraint(model, -y[1] -y[2] + y[3] - ξ[2] == 0)
@constraint(model, y[2] - ξ[3] == 0)

# Set the line capacity constraints
@constraint(model, -y[1] - 35 - z[1] <= yr * M)
@constraint(model, y[1] - 35 - z[1] <= yr * M)
@constraint(model, -y[2] - 50 - z[2] <= yr * M)
@constraint(model, y[1] - 50 - z[2] <= yr * M)

# Set the generator capacity constraints
@constraint(model, -y[3] <= yr * M)
@constraint(model, y[3] - 100 - z[3] <= yr * M)

# Enforce the minimum SF
@constraint(model, expect(1 - yr, ξ) >= α)

# Get the Pareto solutions
for i in eachindex(αs)
    set_value(α, αs[i])
    optimize!(model)
    if has_values(model)
        zs[:, i] = value.(z)
        costs[i] = objective_value(model)
        SFs[i] = sum(1 .- value(yr)) / length(value(yr))
    end
end
```

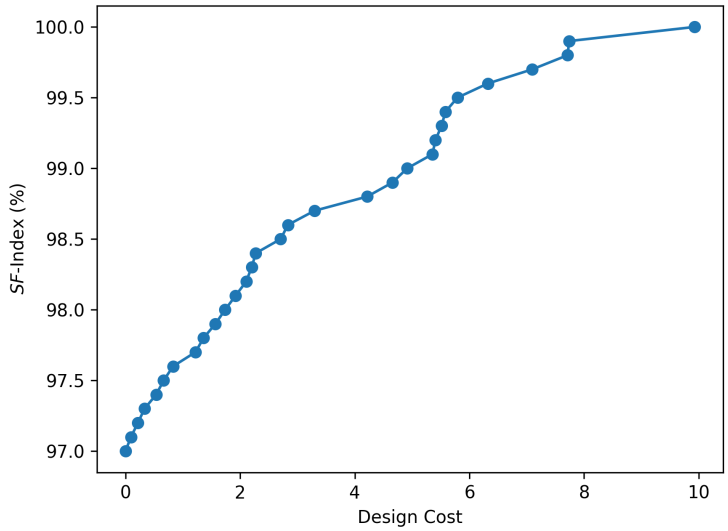



Figure 2.19: The optimal Pareto frontier yielded from solving Problem (2.44) with varied $\alpha$. In total, 31 unique solution pairs are obtained.

Figure 2.19 shows the Pareto frontier we obtain from our analysis. We observe that the base design (i.e., zero cost) has a relatively high $SF$-index. From there we are able to achieve complete flexibility (relative to the MC samples) with a cost of 10.1. This demonstrates how traditional stochastic optimization formulations are readily captured by our unifying abstraction.

### 2.5.3 *Optimal Pandemic Policy Design*

Here we exemplify how stochastic optimal control problem can be readily modeled following our abstraction. We do so by considering a pandemic optimal control problem; here, we seek to combat the spread of a contagion while minimizing the enforced isolation policy $y_u(t) \in [0,1]$ (i.e., social distancing policy). The majority of other pandemic control studies in the literature [73, 74, 75] use integral objectives that uniformly penalize the shape of optimal trajectories. We represent these traditional formulations using the objective:

$$\min_{y_u(\cdot)} \frac{1}{S} \int_{t \in \mathcal{D}_t} y_u(t) dt \tag{2.46}$$

with $S = \int_{t \in \mathcal{D}_t} dt$. This objective minimizes the time-average isolation policy. In Chapter 3 we will demonstrate how our abstraction can be used to propose new measures to shape these trajectories.

We model the spread of the contagion through a given population using the SEIR model [76], which considers four population categories:

$$\text{Susceptible} \rightarrow \text{Exposed} \rightarrow \text{Infectious} \rightarrow \text{Recovered}.$$

We define the fractional populations of individuals susceptible to infection $y_s : \mathcal{D}_t \rightarrow [0,1]$, exposed individuals that are not yet infectious $y_e : \mathcal{D}_t \rightarrow [0,1]$, infectious individuals $y_i : \mathcal{D}_t \rightarrow [0,1]$, and recovered individuals $y_r : \mathcal{D}_t \rightarrow [0,1]$ (considered immune to future infection). The variables are normalized such that $y_s(t) + y_e(t) + y_i(t) + y_r(t) = 1$. The deterministic SEIR model is formalized as:

$$\begin{aligned} \frac{dy_s(t)}{dt} &= (y_u(t) - 1)\beta y_s(t) y_i(t),\ t \in \mathcal{D}_t \\ \frac{dy_e(t)}{dt} &= (1 - y_u(t))\beta y_s(t) y_i(t) - \xi y_e(t),\ t \in \mathcal{D}_t \\ \frac{dy_i(t)}{dt} &= \xi y_e(t) - \gamma y_i(t),\ t \in \mathcal{D}_t \\ \frac{dy_r(t)}{dt} &= \gamma y_i(t),\ t \in \mathcal{D}_t, \end{aligned} \tag{2.47}$$

where $\beta, \gamma, \xi \in \mathbb{R}$ are the rates of infection, recovery, and incubation, respectively. For our case study, we consider $\xi$ to be an uncertain parameter $\xi \sim \mathcal{U}(\underline{\xi}, \overline{\xi})$. This introduces the random domain $\mathcal{D}_\xi$ (i.e., the co-domain of $\mathcal{U}(\underline{\xi}, \overline{\xi})$) and gives a stochastic dynamic

optimization problem of the form:

$$
\begin{aligned}
\min \quad & \frac{1}{S}\int_{t\in\mathcal{D}_t} y_u(t)dt \\
\text{s.t.} \quad & \frac{\partial y_s(t,\xi)}{\partial t} = (y_u(t)-1)\beta y_s(t,\xi)y_i(t,\xi), && t\in\mathcal{D}_t, \xi\in\mathcal{D}_\xi \\
& \frac{\partial y_e(t,\xi)}{\partial t} = (1-y_u(t))\beta y_s(t,\xi)y_i(t,\xi) - \xi y_e(t,\xi), && t\in\mathcal{D}_t, \xi\in\mathcal{D}_\xi \\
& \frac{\partial y_i(t,\xi)}{\partial t} = \xi y_e(t,\xi) - \gamma y_i(t,\xi), && t\in\mathcal{D}_t, \xi\in\mathcal{D}_\xi \\
& \frac{\partial y_r(t,\xi)}{\partial t} = \gamma y_i(t,\xi), && t\in\mathcal{D}_t, \xi\in\mathcal{D}_\xi \\
& y_s(0,\xi) = s_0, y_e(0,\xi) = e_0, y_i(0,\xi) = i_0, y_r(0,\xi) = r_0, && \xi\in\mathcal{D}_\xi \\
& y_i(t,\xi) \le i_{max}, && t\in\mathcal{D}_t, \xi\in\mathcal{D}_\xi \\
& y_u(t) \in [0,\overline{y_u}], && t\in\mathcal{D}_t
\end{aligned}
\tag{2.48}
$$

where $s_0, e_0, i_0, r_0 \in \mathbb{R}$ denote the initial population fractions, $i_{max}$ denotes the maximum allowable fraction of infected individuals $y_i(t)$, and $\overline{y_u}$ denotes the maximum realizable population isolation. The state variables $y_s(\cdot), y_i(\cdot), y_e(\cdot), y_r(\cdot)$ are now infinite variables that are parameterized in the time and random domains, while the control variable $y_u$ is an infinite variable that is only parameterized in the time domain (since we need to decide our control policy before knowing the realizations of $\xi$).

Table 2.1: Parameter values used in the InfiniteOpt formulation (2.48).

| $\beta$ | $\gamma$ | $\underline{\xi}$ | $\overline{\xi}$ | $i_{max}$ | $\overline{y_u}$ | $s_0$ | $e_0$ | $i_0$ | $r_0$ |
|---|---|---|---|---|---|---|---|---|---|
| 0.727 | 0.303 | 0.1 | 0.6 | 0.02 | 0.8 | $1-10^{-5}$ | $10^{-5}$ | 0 | 0 |

We solve the InfiniteOpt problem (2.48) using the parameters defined in Table 2.1 with $\mathcal{D}_t = [0, 200]$. We transcribe it via `InfiniteOpt.jl` using 111 supports for $\mathcal{D}_t$ and 20 MC samples for $\mathcal{D}_\xi$. Code Snippet 2.4 shows an excerpt of this implementation in `InfiniteOpt.jl`. The optimal policies are shown in Figure 2.20. Here a fluctuating policy is able to effectively keep the spread of the contagion within acceptable limits. Moreover, we observe that this is easily implemented in `InfiniteOpt.jl` in a modular manner such that different solution techniques can be quickly tested. In Chapter 3 we use other

measure operators to shape this optimal policy in different ways.

**Code Snippet 2.4: Formulation (2.48) via `InfiniteOpt.jl`.**

```
using InfiniteOpt, Distributions, Ipopt

# Initialize the model
model = InfiniteModel(Ipopt.Optimizer)

# Set the infinite parameters
@infinite_parameter(model, t ∈ [t0, tf], num_supports = 101)
add_supports(t, extra_ts)
@infinite_parameter(model, ξ ~ Uniform(ξ_min, ξ_max), num_supports = 20)

# Set the infinite variables
var_inds = [:s, :e, :i, :r]
@variable(model, 0 ≤ y[var_inds], Infinite(t, ξ))
@variable(model, 0 ≤ yu ≤ 0.8, Infinite(t), start = 0.2)

# Set the integral objective
@objective(model, Min, 1 / (tf - t0) * ∫(yu, t))

# Define the initial conditions
@constraint(model, [v ∈ var_inds], y[v](0, ξ) == y0[v])

# Define the SEIR equations
@constraints(model, begin
    ∂(y[:s], t) == -(1 - yu) * β * y[:s] * y[:i]
    ∂(y[:e], t) == (1 - yu) * β * y[:s] * y[:i] - ξ * y[:e]
    ∂(y[:i], t) == ξ * y[:e] - γ * y[:i]
    ∂(y[:r], t) == γ * y[:i]
end)

# Define the infection limit
@constraint(model, y[:i] ≤ i_max)

# Optimize and get the results
optimize!(model)
state_opt = value.(y, ndarray = true)
control_opt = value(yu)
obj_opt = objective_value(model)
ts = value(t)
ξs = value(ξ)
```

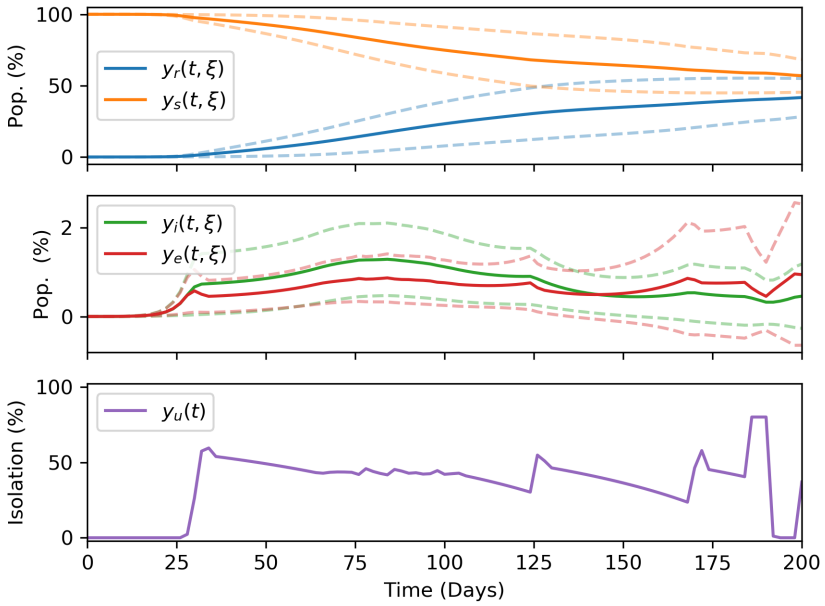



Figure 2.20: Optimal trajectories for formulation (2.48). For the state variables $y_s(t,\xi)$, $y_e(t,\xi)$, $y_r(t,\xi)$, and $y_r(t,\xi)$ the solid lines denote the trajectories averaged over $\xi$ and the dashed lines denote the trajectories that are one standard deviation away from the mean.

# Chapter 3

GENERALIZED RISK MEASURES

The content of this chapter is published in [30, 77].

## 3.1 Introduction

In this chapter, we discuss how risk measures from stochastic optimization (SO) can be transferred to other optimization disciplines as general measure operators following the unifying abstraction for InfiniteOpt problems we presented in Chapter 2. In particular, we will focus on how these generalized measures can be applied to continuous-time dynamic optimization (DO) problems of the form:

$$\begin{aligned} \min_{y(t)\in\mathcal{Y}} \quad & M_t f(y'(t), y(t), t) \\ \text{s.t.} \quad & g(y'(t), y(t), t) \leq 0, \quad t \in \mathcal{D}_t. \end{aligned} \tag{3.1}$$

However, as we will later show, the choice of infinite parameter is arbitrary, and these measures can be applied in other disciplines (e.g., PDE-constrained optimization) [30].

In the context of DO, the infinite domain $\mathcal{D}_t$ is the time domain $[t_0, t_f]$, $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ are time-valued decision functions (e.g., state/control variables), $y'(t) \in \mathbb{R}^{n_y}$ are derivative variables $dy(t)/dt$, $f(\cdot)$ is an infinite-dimensional cost function, and $g(\cdot)$ is a

vector-valued infinite-dimensional constraint functions $g_j(\cdot),\ j \in \mathcal{J} \subseteq \mathbb{R}^{n_g}$.

The measure operator $M_t : \mathcal{D}_t \mapsto \mathbb{R}$ is the focus of this chapter; this operator seeks to scalarize infinite-dimensional cost functions (summarizing them over the time domain $\mathcal{D}_t$) to form a well-posed objective function. Such measures can also be used to handle constraints, but here we focus on objectives to simplify the presentation. Problem (3.1) is general and captures a wide range of DO problems (e.g., model predictive control and state/parameter estimation) where $y(\cdot)$ is comprised of state/control/parameter variables and $g(\cdot) \leq 0$ can include DAE/path/point constraints.

Classical DO formulations minimize the integral of the cost trajectory (known as the Bolza objective). In other words, these formulations use the measure operator:

$$M_t f(t) = \int_{t \in \mathcal{D}_t} f(t) dt \tag{3.2}$$

where we write $f(t) := f(y'(t), y(t), t)$ for convenience. Minimizing (3.2) has the effect of *uniformly* shaping the cost trajectory $f(t)$ over the domain $\mathcal{D}_t$. This also amounts to minimizing the total cost (which is equivalent to the average cost with a normalization factor of $(t_f - t_0)^{-1}$). However, we can envision more advanced DO formulations that aim to shape the cost trajectory $f(\cdot)$ with other measures (e.g., peak or excursion costs). For example, Risbeck and Rawlings recently proposed an MPC objective that penalizes the total and peak of the cost [78]. Minimizing the peak of a cost-trajectory is often a desirable feature in systems, as these can be associated with unsafe behavior and economic penalties.

Under the InfiniteOpt representation, Problem (3.1) is analogous to the SO problem:

$$\begin{aligned} \min_{z \in \mathcal{Z}, y(\cdot) \in \mathcal{Y}} \quad & M_\xi f(z, y(\xi), \xi) \\ \text{s.t.} \quad & g(z, y(\xi), \xi) \leq 0, \quad \xi \in \mathcal{D}_\xi. \end{aligned} \tag{3.3}$$

where $z \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$ are here-and-now decision variables, $\xi$ is a random parameter, $\mathcal{D}_\xi \subseteq \mathbb{R}^{n_\xi}$ is the co-domain of its distribution, and $y(\xi) \in \mathcal{Y} \in \mathbb{R}^{n_y}$ are recourse decision

variables. We observe that (3.1) is a special case of (3.3) if we define $n_z = 0$, $n_\xi = 1$, $\mathcal{D}_\xi = \mathcal{D}_t$, $M_\xi = M_t$, and set $y(\xi)$ to contain both $y'(t)$ and $y(t)$.

In SO, the measure $M_\xi$ can draw from a wide collection of risk measures (summarizing statistics) that aim to shape the probability density function of $f(\xi)$; some examples include the expected value, mean-variance, absolute deviation, and conditional-value-at-risk (CVaR) [79]. Risk measures enable greater flexibility in shaping the cost function $f(\xi)$ and are commonly used to penalize extreme events (i.e., high costs). By leveraging the connection between (3.1) and (3.3), we will see that risk measures can readily be transferred to DO.

In this chapter, we formalize the use of risk measures to DO problems that follow (3.1). This new class of dynamic measures $M_t$ enable us to shape trajectories and policies in new ways. Moreover, we highlight how the Julia-based modeling package `InfiniteOpt.jl` provides a useful interface to compactly express such measures.

## 3.2 Time-Valued Density Functions

We begin by establishing time analogues of probability density functions (pdf) and of cumulative density functions (cdf) typically used in SO (i.e., pdfs and cdfs defined over the time domain $\mathcal{D}_t$). These provide key constructs to facilitate the use of risk measures in DO.

In SO, $\xi$ is described by the pdf $p_\xi : \mathcal{D}_\xi \mapsto \mathbb{R}_{\geq 0}$, which satisfies $\int_{\xi \in \mathcal{D}_\xi} p_\xi(\xi) d\xi = 1$. The pdf is used to compute statistics such as the expectation:

$$\mathbb{E}_\xi[f(\xi)] = \int_{\xi \in \mathcal{D}_\xi} f(\xi) p_\xi(\xi) d\xi. \tag{3.4}$$

This measure $M_\xi = \mathbb{E}_\xi$ is commonly used in SO; here, we can see that $p_\xi(\cdot)$ acts as a weighting function that places varied emphasis over the domain $\mathcal{D}_\xi$. The expectation summarizes the cost function in a single scalar value.

Following the analogy of SO and DO, we define a weighting function $p_t : \mathcal{D}_t \mapsto \mathbb{R}_{\geq 0}$ (i.e., a time-valued pdf) to yield the time average (expectation):

$$\mathbb{E}_t[f(t)] := \int_{t \in \mathcal{D}_t} f(t) p_t(t) dt. \tag{3.5}$$

This provides us flexibility in prioritizing different regimes in the domain $\mathcal{D}_t$. Note that the selection of the notation $t$ to denote the infinite domain is arbitrary; one can simply define a general infinite parameter $d$. We obtain (3.2) as a special case by setting $p(t) = 1$; however, we can envision choosing from a wide range of candidate pdfs, such as:

$$p_t(t) = \frac{1}{t_f - t_0} \tag{3.6}$$

Note that this pdf of the uniform random parameter and places equal emphasis on different parts of the time domain. Figure 3.1 provides a geometric interpretation of the time-expectation using the pdf in (3.6); here, the area of the rectangle with height $\mathbb{E}_t[f(t)]$ and width $t_f - t_0$ is equivalent to the area under $f(t)$ [80].

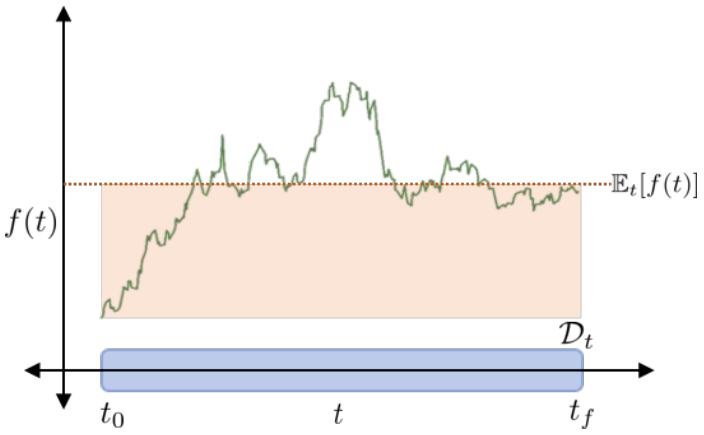



Figure 3.1: Visualization of the expectation measure $\mathbb{E}_t[f(t)] = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} f(t) dt$ where the rectangle formed has an area equal to that of the region under $f(t)$.

This inspires the consideration of defining other weighting functions associated with probability densities of different random parameters (e.g., Gaussian, exponential, Weibull);

this would have the effect of inducing interesting prioritization strategies that can be used to shape the cost surface in desirable ways. For instance, a Gaussian weighting function places emphasis on the middle of the time domain (and emphasis decays rapidly as one moves away from the center of the domain), while an exponential weighting function places emphasis at the beginning of the domain (and decays rapidly as one marches in time). This modeling feature can be useful in dynamic optimization and optimal control problems in which it is often desirable to place more/less emphasis on initial or final conditions. For instance, in infinite-horizon problems, the exponential pdf with decay rate parameter $\gamma \in \mathbb{R}_{>0}$:

$$p_t(t) = \gamma e^{-\gamma t} \tag{3.7}$$

behaves as a discount factor [36, 58, 81].

The cdf $P : \mathcal{D}_\xi \mapsto [0, 1]$ of a random variable $\xi$ is:

$$P(\xi; \hat{\xi}) := \mathbb{P}_\xi(\xi \leq \hat{\xi}) = \int_{\xi \in \{\xi \in \mathcal{D}_\xi : \xi \leq \hat{\xi}\}} p_\xi(\xi) d\xi \tag{3.8}$$

This denotes the cumulative probability of finding $\xi$ below the threshold $\hat{\xi} \in \mathcal{D}_\xi$. In the context of the DO, we can write the cdf of the time trajectory $f(t)$ by using the *excursion sets*:

$$\begin{aligned} \mathcal{D}_t^+(f(t); \hat{f}) &:= \{t \in \mathcal{D}_t : f(t) \geq \hat{f}\} \\ \mathcal{D}_t^-(f(t); \hat{f}) &:= \{t \in \mathcal{D}_t : f(t) \leq \hat{f}\} \end{aligned} \tag{3.9}$$

where $\mathcal{D}_t^+(f(t); \hat{f}) \subseteq \mathcal{D}_t$ and $\mathcal{D}_t^-(f(t); \hat{f}) \subseteq \mathcal{D}_t$ are the positive and negative function excursion sets, respectively. We can use the negative excursion set to establish:

$$P(f(t); \hat{f}) = \int_{t \in \mathcal{D}_t^-(f(t); \hat{f})} p_t(t) dt. \tag{3.10}$$

In a DO context, the cdf measures the fraction of time that the trajectory $f(t)$ is below the threshold $\hat{f}$. We will see that expressing the cdf in terms of excursion set allows us to interpret measures as mechanisms to bound trajectories.

## 3.3 Dynamic Measures

Here we illustrate the interpretation of risk measures in a DO context. Numerous risk measures have been proposed in the SO community and analyzing them is beyond the scope of this work (we refer the reader to [82] for a review). This section establishes key constructs and exemplifies the steps needed to interpret risk measures in a time setting. We discuss properties of these proposed measures in Section 3.4.

### 3.3.1 *Expectation*

The time expectation measure $\mathbb{E}_t$ shown in Equation (3.5) allows us to assess a weighted average of our cost trajectory $f(t)$ in accordance with the weighting function $p_t(t)$. In the context of DO, minimizing the expected cost provides an easily interpretable objective and the key modeling choice lies in the selection of $p_t(t)$. Reasonable candidates for many DO applications are (3.6) and (3.7), as demonstrated in Section 3.5; however, other weighting functions are possible (e.g., Gaussian and Gamma). The selection of the weighting function dictates how much emphasis is placed on different parts of the time domain. The expectation serves as a core construct in defining the more sophisticated measures.

**Proposition 3.1.** *The time expectation* $\mathbb{E}_t$ *in* (3.5) *is a special case of* $\mathbb{E}_\xi$ *in* (3.4) *if* $n_z = 0$, $n_\xi = 1$, $\mathcal{D}_\xi = \mathcal{D}_t$.

### 3.3.2 *Mean-Variance*

The mean-variance $\mathbb{E}$-$\mathbb{V}_\xi$ is a classical measure used in portfolio optimization [83]. For SO problems (3.3), this minimizes the variance (i.e., spread) of the cost outcomes in an attempt to mitigate high-cost events:

$$\mathbb{E}\text{-}\mathbb{V}_\xi[f(\xi)] := \mathbb{E}_\xi[f(\xi)] + \lambda \mathbb{V}_\xi[f(\xi)] \tag{3.11}$$

where $\mathbb{V}_\xi = \mathbb{E}_\xi[(f(\xi) - \mathbb{E}_\xi[f(\xi)])^2]$ is the variance and $\lambda \in \mathbb{R}_{\geq 0}$ is a tradeoff parameter. Transferring this to a DO setting we obtain the time-valued measure:

$$\mathbb{E}\text{-}\mathbb{V}_t[f(t)] := \mathbb{E}_t[f(t)] + \lambda \mathbb{V}_t[f(t)]. \tag{3.12}$$

Minimizing $\mathbb{E}\text{-}\mathbb{V}_t[f(t)]$ in a DO problem provides a tradeoff problem that seeks to minimize the magnitude of the cost trajectory (the expectation) and the variability/fluctuations (variance) of the cost trajectory. A disadvantage of this measure is that it penalizes cost variability equally for low and high costs. However, this property can be advantageous for cost functions that seek to enforce smooth control trajectories.

**Proposition 3.2.** *The measure operator* $\mathbb{E}\text{-}\mathbb{V}_t$ *from* (3.12) *is a special case of* $\mathbb{E}\text{-}\mathbb{V}_\xi$ *from* (3.11) *under the same conditions of Proposition 3.1.*

### 3.3.3 *Quantile*

The quantile $Q_\xi(f(\xi); \alpha)$ (also referred to as the value-at-risk) denotes the threshold value $\hat{f}$ for $f(\xi)$ such that the cumulative probability of incurring costs below the threshold is at least $\alpha \in [0, 1]$:

$$Q_\xi(f(\xi); \alpha) := \inf_{\hat{f} \in \mathbb{R}} \left\{ P(f(\xi); \hat{f}) \geq \alpha \right\}. \tag{3.13}$$

Constraining (3.13) is equivalent to enforcing a probabilistic constraint [84]:

$$Q_\xi(f(\xi); \alpha) \leq 0 \iff \mathbb{P}_\xi(f(\xi) \leq 0) \geq \alpha. \tag{3.14}$$

We can use the cdf (3.10) in combination with (3.13) to define the time-valued quantile:

$$Q_t(f(t); \alpha) := \inf_{\hat{f} \in \mathbb{R}} \left\{ \int_{t \in \mathcal{D}_t^-(f(t); \hat{f})} p_t(t) dt \geq \alpha \right\}. \tag{3.15}$$

Using this measure in (3.1) minimizes the excursion threshold of the cost function trajectory such that the fraction of time that exceed it is no more than $1 - \alpha$. Unlike the $\mathbb{E}\text{-}\mathbb{V}_t$

measure, the quantile measure only penalizes high cost values, making it an attractive alternative in certain cases. However, the potential disadvantages of this measure are that it does not strongly discourage high cost peaks in the positive function excursion $\mathcal{D}_t^+(f(t); Q_t(f(t); \alpha))$ and it is nonconvex and difficult to compute in general.

**Proposition 3.3.** *The quantile measure $Q_t$ (3.15) is a special case of its analogue $Q_\xi$ in (3.13) under the same conditions of Proposition 3.1.*

### 3.3.4 *Conditional-Value-at-Risk*

The conditional-value-at-risk (CVaR) measure seeks to address the limitations of the quantile measure $\mathbb{E}$-$\mathbb{V}_\xi$ by penalizing the expected value of the $1-\alpha$ largest cost values:

$$\text{CVaR}_\xi(f(\xi); \alpha) := \min_{\hat{f} \in \mathbb{R}} \left\{ \hat{f} + \frac{1}{1-\alpha} \mathbb{E}_\xi[f(\xi) - \hat{f}]_+ \right\} \tag{3.16}$$

where $\mathbb{E}_\xi[f(\xi) - \hat{f}]_+ := \mathbb{E}_\xi[\max(0, f(\xi) - \hat{f})]$ and $\alpha \in [0, 1)$. CVaR is also known as the superquantile; under mild assumptions, CVaR can be represented as:

$$\text{CVaR}_\xi(f(\xi); \alpha) = \mathbb{E}_\xi[f(\xi) : f(\xi) \geq Q_t(f(t); \alpha)] \tag{3.17}$$

since the minimizer $\hat{f}^*$ is $Q_t(f(t); \alpha)$ [85]. Hence, minimizing CVaR has the effect of minimizing the conditional expectation over the $1-\alpha$ probability region with the highest cost, thus hedging against extreme events. Moreover, the calculation of CVaR at a probability level $\alpha$ implicitly defines the quantile of $f(\xi)$.

A key property of CVaR is that it is a general measure that captures the expectation $\text{CVaR}_\xi(f(\xi); \alpha) = \mathbb{E}_\xi[f(\xi)]$ as $\alpha \to 0$ and the worst-case $\text{CVaR}_\xi(f(\xi); \alpha) = \sup_\xi(f(\xi))$ as $\alpha \to 1$. Figure 3.2 shows how these measures are typically interpreted in terms of the probability density function of the cost $f(\xi)$, motivating the $\alpha$ and $1-\alpha$ probability regions.

Following the excursion-based definition of the cdf, one can interpret CVaR as a mea-

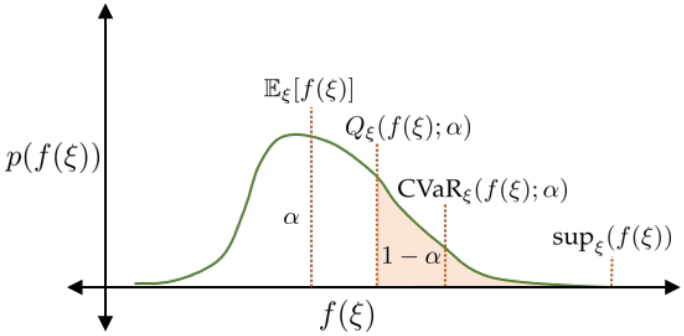



Figure 3.2: An illustration of $\mathrm{CVaR}_\xi(f(\xi);\alpha)$ in terms of the density function $p(f(\xi))$.

sure that captures the *excursion* of a function (field) from a given threshold. This reveals that CVaR considers the expectation of $f(\xi)$ over the restricted domain $\mathcal{D}_\xi^+(q_\alpha)$, which indexes the $1-\alpha$ probabilistic region shown in Figure 3.2:

$$\mathrm{CVaR}_\xi(f(\xi);\alpha) = \frac{1}{1-\alpha}\int_{\xi\in\mathcal{D}_\xi^+(f(\xi);Q_\xi(f(\xi);\alpha))} f(\xi)p(\xi)d\xi. \tag{3.18}$$

Figure 3.3 illustrates that $\mathrm{CVaR}_\xi(f(\xi);\alpha)$ using this functional interpretation for a realization of $f(\xi)$ and compares it to the other measures shown in Figure 3.2.

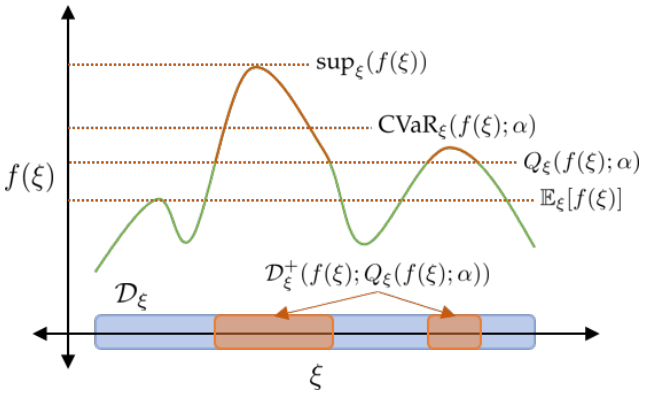



Figure 3.3: Illustration of $\mathrm{CVaR}_\xi(f(\xi);\alpha)$ following the representation given in (3.18). This provides an alternative view of the probabilistic representation shown in Figure 3.2.

With the above observations, the time-valued CVaR measure can be expressed as:

$$\mathrm{CVaR}_t(f(t);\alpha) := \min_{\hat{f}\in\mathbb{R}}\left\{\hat{f} + \frac{1}{1-\alpha}\mathbb{E}_t[f(t)-\hat{f}]_+\right\}. \tag{3.19}$$

This provides a convex measure that penalizes the high (peak) costs incurred in the positive function excursion set $\mathcal{D}_t^+(f(t); Q_t(f(t);\alpha))$. Note that this penalizes *multiple* peak costs (and not just the peak cost, as done in typical DO formulation). Moreover, one can show that:

$$\begin{aligned} \lim_{\alpha\to 0} \text{CVaR}_t(f(t);\alpha) &= \mathbb{E}_t[f(t)] \\ \lim_{\alpha\to 1} \text{CVaR}_t(f(t);\alpha) &= \max_{t\in\mathcal{D}_t} f(t) \end{aligned} \tag{3.20}$$

which both follow from Equation (3.17). As such, CVaR is highly versatile measure for use in DO that can capture both average and extreme features of a time trajectory. Figure 3.4 illustrates the application of this measure over the time domain and shows that this is analogous to the application over a random domain shown in Figure 3.3.

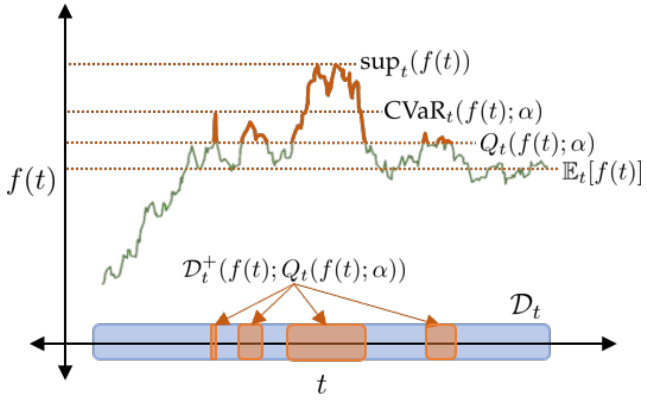



Figure 3.4: Illustration of $\text{CVaR}_t(f(t);\alpha)$, as represented in (3.19).

**Proposition 3.4.** *The CVaR measure in Equation* (3.19) *is a special case of CVaR$_\xi$ in Equation* (3.16) *under the same conditions of Proposition* 3.1.

### 3.3.5 *Disutility*

Disutility risk measures are another prevalent measure class used in SO; this employs an expectation over a disutility function $g : \mathbb{R} \mapsto \mathbb{R}$ (typically a convex increasing function)

that penalizes unfavorable values of $f(\xi)$:

$$D_{\tilde{\xi}}(f(\xi)) := \mathbb{E}_{\tilde{\xi}}[g(f(\xi))]. \tag{3.21}$$

In an effort to make a translation invariant measure, the measure is often expressed as:

$$\tilde{D}_{\tilde{\xi}}(f(\xi)) := \inf_{\hat{f}\in\mathbb{R}} \mathbb{E}_{\tilde{\xi}}[f(\xi) + g(f(\xi) - \hat{f})]. \tag{3.22}$$

One can show that $\text{CVaR}_{\tilde{\xi}}(f(\xi);\alpha)$ is a special case of (3.22). This follows by letting $g(x;\alpha) = (1-\alpha)^{-1}\max(0,x) - x$ where $x \in \mathbb{R}$. Then by substituting $g(x;\alpha)$ in (3.22) we obtain (3.16). We transfer (3.22) to DO by using the time-valued expectation $\mathbb{E}_t$:

$$\tilde{D}_t(f(t)) := \inf_{\hat{f}\in\mathbb{R}} \mathbb{E}_t[f(t) + g(f(t) - \hat{f})]. \tag{3.23}$$

This measure class provides great flexibility in shaping dynamic trajectories as there are diverse choices of $g(\cdot)$ (in addition to the flexibility provided via selecting the weighting function $p_t(\cdot)$). A useful survey on the properties of disutility functions in the context of SO is provided in [86].

**Proposition 3.5.** *The time-valued disutility measure $\tilde{D}_t(f(t))$ is a special case of $\tilde{D}_{\tilde{\xi}}(f(\xi))$ under the same conditions of Proposition 3.1.*

## 3.4 Measure Properties

Here we formalize some key mathematical properties of the dynamic measures presented in Section 3.3. This provides some interesting and useful insights on the behavior that these measures induce. These properties have been studied in the SO community, and we will show that this rich theory can be readily applied to DO.

In the context of SO, four main properties are typically considered for risk measures: convexity, monotonicity, translation invariance, and positive homogeneity. Moreover, a

measure operator is said to be coherent if it satisfies all these properties [79, 87].

Convexity asserts that a measure operator $M_{\tilde{\xi}}$ satisfy:

$$M_{\tilde{\xi}}(\beta f + (1-\beta)h) \leq \beta M_{\tilde{\xi}}(f) + (1-\beta)M_{\tilde{\xi}}(h) \tag{3.24}$$

for all measurable functions $f(\xi), h(\xi) : \mathcal{D}_{\tilde{\xi}} \mapsto \mathbb{R}$ in the linear function space $\mathscr{F}$ and all $\beta \in [0,1]$. This property is key for creating optimization objectives that are well-posed and guarantees that the measure of a convex cost is also convex.

Under monotonicity, we have that if $f_1(\xi) \succeq f_2(\xi)$ ($f_1(\xi)$ dominates $f_2(\xi)$), then the measure $M_{\tilde{\xi}}$ satisfies:

$$M_{\tilde{\xi}}(f_1(\xi)) \geq M_{\tilde{\xi}}(f_2(\xi)). \tag{3.25}$$

This ensures that, if a cost function dominates another cost function, then the measure former will also be greater than that of the latter. The concept of dominance (comparing whether a random variable is better than another random variable) is an interesting and important concept that has not been explored in DO. In a DO context, dominance of first-order ($f_1(t) \succeq f_2(t)$) requires that $P(f_1(t) > \hat{f}) \geq P(f_2(t) > \hat{f})$ for any threshold value $\hat{f}$. In other words, the fraction of time that the trajectory $f_1(t)$ remains above the threshold is greater or equal than the fraction of time that the trajectory $f_2(t)$ remains above the same threshold. A monotonic measure is such that, if $f_1(t) \succeq f_2(t)$, then $M_t(f_1(t)) \geq M_t(f_2(t))$ holds. Note that dominance holds trivially if $f_1(t) \geq f_2(t)$ for all $t \in \mathcal{D}_t$. These concepts are important because comparisons (benchmarks) of time trajectories are not as straightforward [88], as the trajectories are functions (not scalar values) and thus a trajectory might be better in some parts of the time domain but not in others

A translation invariant measure satisfies:

$$M_{\tilde{\xi}}(f(\xi) + a) = M_{\tilde{\xi}}(f(\xi)) + a \tag{3.26}$$

if $a \in \mathbb{R}$ and $f(\xi) \in \mathscr{F}$. In a DO context, this property ensures that offsetting the cost function will not change the shape of the optimal cost trajectory.

The positive homogeneity property is given by:

$$M_{\xi}(\tau f(\xi)) = \tau M_{\xi}(f(\xi)) \tag{3.27}$$

if $\tau > 0$ and $f(\xi) \in \mathscr{F}$. In the context of DO, a positive homogeneous measure provides the property that uniformly scaling the cost by $\tau$ will not affect the shape of the optimal cost trajectory.

The analysis of the stochastic risk measures featured in Section 3.3 is well-established in the SO literature and Table 3.1 provides a summary of these [87, 79].

| $M_{\xi}$ | (3.24) | (3.25) | (3.26) | (3.27) |
|---|---|---|---|---|
| $\mathbb{E}_{\xi}$ | Yes | Yes | Yes | Yes |
| $\mathbb{E}$-$\mathbb{V}_{\xi}$ | Yes | No | Yes | No |
| $Q_{\xi}$ | No | Yes | Yes | Yes |
| $\text{CVaR}_{\xi}$ | Yes | Yes | Yes | Yes |
| $\tilde{D}_{\xi}$ | Yes | Yes | Yes | Yes |

Table 3.1: A summary of the properties satisfied by certain measures $M_{\xi}$. Note that $\tilde{D}_{\xi}$ only satisfies (3.27) if $g(\cdot)$ is positive homogeneous.

Since the time-valued measures are special cases of the SO counterparts, they inherit the properties of Table 3.1. This illustrates how one can transfer rich theory from SO (with respect to these measure operators) to a DO context. The time-valued expectation measure is a coherent risk measure; this might explain why this has been the classical measure used in DO. It is particularly important to observe that convexity ensures that the use of this measure yields a convex objective if the cost function is convex. Interestingly, the convexity of the objective has key implications for establishing stability conditions (e.g., closed-loop stability of MPC) [89]. From this, we observe that other non-convex measures such as $\mathbb{E}$-$\mathbb{V}_t$ and $Q_t$ may not yield stability. On the other hand, $\text{CVaR}_t$ and $\tilde{D}_t$ are convex and thus might inherit stability properties (this is an interesting topic of future work).

Another key observation is to recall that the choice of infinite parameter $t$ was arbitrary for this analysis. Hence, transferring risk-measures to other domains (e.g., spatial position) will inherit the properties in Table 3.1 in like manner.

## 3.5 Case Study

In this section, we proceed to compare the time-valued measures proposed in Section 3.3 through illustrative optimal control studies. These are implemented in `InfiniteOpt.jl v0.5.1` and use `Ipopt v3.14.4` to solve the transcribed forms. Moreover, these highlight how the measure-centric modeling abstraction behind `InfiniteOpt.jl` makes it easy to conduct these comparisons.

### 3.5.1 *Optimal Pandemic Policy Design*

We compare the time-valued measures proposed in Section 3.3 in the context of optimal control. We adapt the pandemic control problem from Section 2.5.3 that seeks to choose an isolation policy to control the spread of a contagion that minimally impacts the economic impact (imposed by mandated isolation). The simplifying assumption is that $\xi$ is known and not uncertain. Thus, our state variables are comprised of $y_s(t)$, $y_e(t)$, $y_i(t)$, and $y_r(t)$. Moreover, we exhibit control by imposing an isolation policy $y_u(t) \in [0, \overline{y_u}] \subseteq [0, 1]$ that entails the separation of susceptible and exposed individuals ($y_u(t) = 0$ denotes no separation and $y_u(t) = 1$ denotes complete separation).

The majority of other pandemic control studies in the literature [73, 74, 75] use integral objectives that uniformly penalize the shape of optimal trajectories (as shown in Section 2.5.3). We represent traditional formulations using the objective:

$$\min_{y_u(\cdot)} \frac{1}{S} \int_{t \in \mathcal{D}_t} y_u(t) dt \tag{3.28}$$

with $S = \int_{t \in \mathcal{D}_t} dt$. From this we introduce the time-value expectation objective:

$$\min_{y_u(\cdot)} \mathbb{E}_t[y_u(t)] \tag{3.29}$$

which is equivalent to (3.28) when we use the uniform pdf shown in (3.6). We also consider an alternative objective by incorporating a peak penalty to also control the maximum isolation policy:

$$\min_{y_u(\cdot)} \max_{t \in \mathcal{D}_t} \; y_u(t). \tag{3.30}$$

This new objective can be formulated as:

$$\begin{aligned} \min_{y_u(\cdot), z} \quad & z \\ \text{s.t.} \quad & z \geq y_u(t), \quad t \in \mathcal{D}_t \end{aligned} \tag{3.31}$$

where $z \in \mathbb{R}$ is an auxiliary variable that captures the peak [23]. We will now show that objectives (3.29) and (3.30) are special cases of the CVaR objective:

$$\min_{y_u(\cdot)} \text{CVaR}_t(y_u(t); \alpha) \tag{3.32}$$

where $\alpha \in [0, 1)$. This problem can be reformulated as:

$$\begin{aligned} \min_{y_u(\cdot), y_m(\cdot), z} \quad & z - \frac{1}{1-\alpha} \mathbb{E}_t[y_m(t)] \\ \text{s.t.} \quad & y_m(t) \geq y_u(t) - z, \quad && t \in \mathcal{D}_t \\ & y_m(t) \geq 0, && t \in \mathcal{D}_t \end{aligned} \tag{3.33}$$

where $y_m : \mathcal{D}_t \to \mathbb{R}$ and $z \in \mathbb{R}$ are appropriate infinite and finite auxiliary variables [90]. Finally, we consider the mean-variance objective:

$$\min_{y_u(\cdot)} \mathbb{E}_t[y_u(t)] + \lambda \mathbb{V}_t[y_u(t)] \tag{3.34}$$

which is equivalent to:

$$\min_{y_u(\cdot)} \mathbb{E}_t[y_u(t)] + \lambda \mathbb{E}_t\left[(y_u(t) - \mathbb{E}_t[y_u(t)])^2\right]. \tag{3.35}$$

We apply these group of measured objectives in a formulation that seeks to minimize the measured isolation policy function $M_t y_u(t)$ while enforcing that the amount of infectious individuals $y_i(t)$ remains below $\overline{y_i} \in (0, 1]$:

$$\begin{aligned}
\min \quad & M_t y_u(t) \\
\text{s.t.} \quad & \frac{dy_s(t)}{dt} = (y_u(t) - 1)\,\beta y_s(t) y_i(t), && t \in \mathcal{D}_t \\
& \frac{dy_e(t)}{dt} = (1 - y_u(t))\,\beta y_s(t) y_i(t) - \xi y_e(t), && t \in \mathcal{D}_t \\
& \frac{dy_i(t)}{dt} = \xi y_e(t) - \gamma y_i(t), && t \in \mathcal{D}_t \\
& \frac{dy_r(t)}{dt} = \gamma y_i(t), && t \in \mathcal{D}_t \\
& y_s(0) = s_0, y_e(0) = e_0, y_i(0) = i_0, y_r(0) = r_0 \\
& y_i(t) \leq \overline{y_i}, && t \in \mathcal{D}_t \\
& y_u(t) \in [0, \overline{y_u}], && t \in \mathcal{D}_t
\end{aligned} \tag{3.36}$$

where $s_0, e_0, i_0, r_0 \in [0, 1]$ are initial conditions and $\beta, \gamma, \xi \in \mathbb{R}$ are the rates of infection, recovery, and incubation, respectively, which are specific to the disease in question. Here we specify initial conditions at $s_0 = .9999$, $e_0 = 10^{-5}$, and $i_0 = r_0 = 0$. The disease parameters are taken to be $\beta = 0.727$, $\gamma = 0.303$, and $\xi = 0.3$. We choose limits $\overline{y_u} = 0.8$ and $\overline{y_i} = 0.02$. Finally, we set $\mathcal{D}_t = [0, 200]$. We model Formulation (3.36) in `InfiniteOpt.jl` and use backward finite-difference to evaluate the derivatives using 101 discretization points. Code Snippet 3.1 highlights the compact syntax required to model Formulation (3.36) in `InfiniteOpt.jl` under these conditions. We explore the behavior of the varied objectives mentioned above, and we investigate the implications of using the uniform time-valued pdf in (3.6) against the exponential pdf of (3.7).

**Code Snippet 3.1: Formulation (3.36) implemented in `InfiniteOpt.jl`.**

```
using InfiniteOpt, Ipopt

# Set the parameters
γ, β, ξ = 0.303, 0.727, 0.3
s0, e0, i0, r0 = 1 - 1e-5, 1e-5, 0, 0

# Define the model
model = InfiniteModel(Ipopt.Optimizer)

# Define the time parameter
@infinite_parameter(model, t ∈ [0, 200], num_supports = 101)

# Add the variables
@variable(model, ys, Infinite(t))
@variable(model, ye, Infinite(t))
@variable(model, yi ≤ 0.02, Infinite(t))
@variable(model, yr, Infinite(t))
@variable(model, 0 ≤ yu ≤ 0.8, Infinite(t))

# Set the time expectation objective
@objective(model, Min, 𝔼(yu, t))

# Define the SEIR equations
@constraint(model, ∂(ys, t) == -(1 - yu) * β * ys * yi)
@constraint(model, ∂(ye, t) == (1 - yu) * β * ys * yi - ξ * ye)
@constraint(model, ∂(yi, t) == ξ * ye - γ * yi)
@constraint(model, ∂(yr, t) == γ * yi)
@constraint(model, ys(0) == ys0)
@constraint(model, ye(0) == ye0)
@constraint(model, yi(0) == yi0
@constraint(model, yr(0) == yr0)

# Solve the model and retrieve results
optimize!(model)
u_opt = value(yu)
ts = value(t)
```

Figure 3.5 summarizes the results; the expectation $\mathbb{E}_t[y_u(t)]$ with the uniform pdf shapes $y_u(t)$ identically to the classical integral measure. However, we are able to place increased emphasis on the early time regime when we use the exponential pdf defined in (3.7) in combination with $\mathbb{E}_t[y_u(t)]$. In comparison to the other expectation measure, the exponentially weighted counterpart exhibits a policy trajectory that is significantly reduced at early times while later times lead to increased isolation requirements. This highlights how the choice of pdf $p_t(t)$ enhances the flexibility of our proposed measures in accordance with the requirements of the problem. The optimal policy we obtain with the mean-variance measure $\mathbb{E}\text{-}\mathbb{V}_t(y_u(t); 8)$ demonstrates how placing increased priority on minimizing the variance of the cost function induces the trajectory to be increasingly smoothed (i.e., cost fluctuations are damped). This comes at the trade-off (controlled via specification of $\lambda$) of increasing the mean isolation policy, but helps to derive a more

consistent policy. For this application, a smoother policy would likely be preferred since rapid policy changes can be highly disruptive and can lead to public dissatisfaction. Finally, in contrast to the mean-variance (which equally penalizes positive and negative cost deviations from the mean), the CVaR measure only penalizes the high-cost deviations that surpass the threshold determined by the $\alpha$-quantile. In Figure 3.5 we see that $\text{CVaR}_t(y_u(t); 0.9)$ flattens the peak isolation policy values observed with the standard integral/expectation policy. This hedging against high costs also induces a more substantial response at later times which results in a larger cumulative cost. Thus, we observe a trade-off (controlled via $\alpha$) between penalizing cost peaks and minimizing the total cumulative cost.

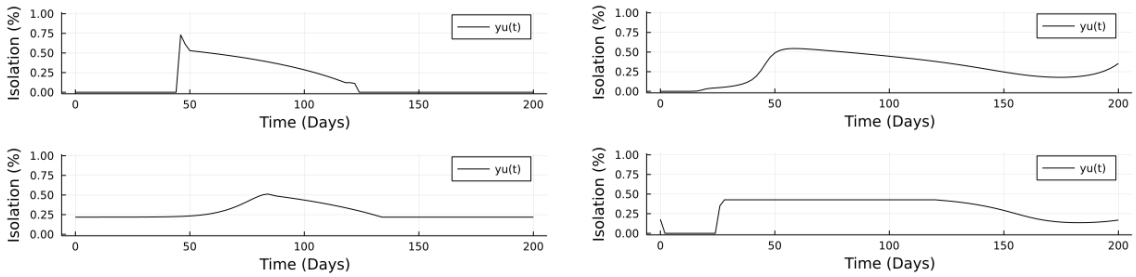



Figure 3.5: The optimal policy trajectories $y_u(t)$. Top Left: $\mathbb{E}_t$ with uniform pdf. Top Right: $\mathbb{E}_t$ with exponential pdf ($\gamma = 1$). Bottom Left: $\mathbb{E}$-$\mathbb{V}_t$ with $\lambda = 8$ and uniform pdf. Bottom Right: $\text{CVaR}_t$ with $\alpha = 0.9$ and uniform pdf.

### 3.5.2 *Double Level Control*

We adopt a benchmark optimal control problem involving the level control of a two tank system. Figure 3.6 summarizes the tank system. We model the pump rate $y_p(t)$ as the control variable with states $y_{h1}(t)$ and $y_{h2}(t)$. The model seeks to minimize the deviation

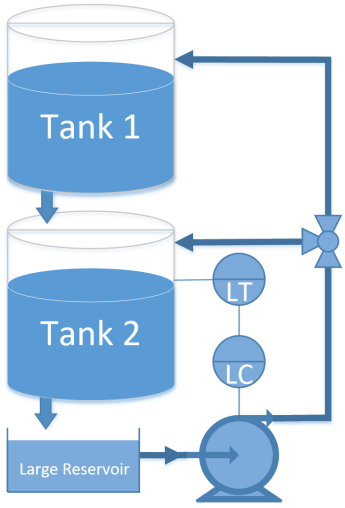



Figure 3.6: Representation of the tank system featured in Section 3.5.2.

from the setpoint $\overline{y_{h2}}(t)$ of the height of the second tank:

$$\begin{aligned}
\min_{y_{h1}(t),y_{h2}(t),y_p(t)} \quad & M_t \left(y_{h2}(t) - \overline{y_{h2}}(t)\right)^2, && t \in \mathcal{D}_t \\
\text{s.t.} \quad & \frac{dy_{h1}(t)}{dt} = \omega_1 y_p(t) - \omega_2 \sqrt{y_{h2}(t)}, && t \in \mathcal{D}_t \\
& \frac{dy_{h2}(t)}{dt} = \omega_2 \sqrt{y_{h1}(t)} - \omega_2 \sqrt{y_{h2}(t)}, && t \in \mathcal{D}_t \\
& y_{h2}(0) = y_{h1}(0) = 0.5 \\
& 0 \leq y_p(t) \leq 1, && t \in \mathcal{D}_t \\
& 0 \leq y_{h1}(t), y_{h2}(t) \leq 1, && t \in \mathcal{D}_t
\end{aligned} \tag{3.37}$$

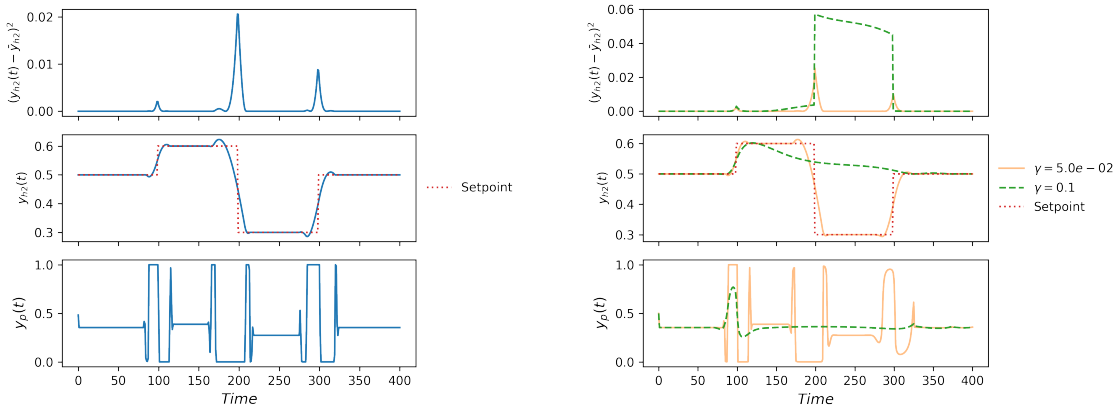
where $\omega_1$, $\omega_2$ = 0.08, 0.04 and are the inlet valve and tank outlet valve coefficients, respectively. Additionally, we set $\mathcal{D}_t = [0, 400]$ seconds. We model Formulation (3.37) in `InfiniteOpt.jl` and use backward finite difference to evaluate the derivatives in accordance with 401 discretization points. For the objective function, we consider the following time-valued measures: $\mathbb{E}_t$ using uniform and exponential pdfs, $\mathbb{V}$-$\mathbb{E}_t$, and $\text{CVaR}_t$. Code Snippet 3.2 shows how this is compactly implemented.

**Code Snippet 3.2: Formulation (3.37) implemented in `InfiniteOpt.jl`.**

```julia
using InfiniteOpt, Ipopt

# Set the parameters
w1, w2 = 0.08, 0.04

# Define the model
model = InfiniteModel(Ipopt.Optimizer)

# Define the time parameter
@infinite_parameter(model, t ∈ [0, 400], num_supports = 401)

# Add the doublet setpoint
yh2_bar = parameter_function(t) do t
            if 100 <= t < 200
                return 0.6
            elseif 200 <= t < 300
                return 0.3
            else
                return 0.5
            end
        end

# Add the variables with their bounds
@variable(model, 0 ≤ yh1 ≤ 1, Infinite(t), start = 0.5)
@variable(model, 0 ≤ yh2 ≤ 1, Infinite(t), start = 0.5)
@variable(model, 0 ≤ yp ≤ 1, Infinite(t))

# Set the time expectation objective
@objective(model, Min, 𝔼((yh2 - yh2_bar) ^ 2, t))

# Define the equations
@constraint(model, ∂(yh1, t) == w1 * yp - w2 * sqrt(yh2))
@constraint(model, ∂(yh2, t) == w2 * sqrt(yh1) - w2 * sqrt(yh2))
@constraint(model, yh1(0) == 0.5)
@constraint(model, yh2(0) == 0.5)

# Solve the model and retrieve results
optimize!(model)
yh2_opt = value(yh2)
ts = value(t)
```

We summarize the results obtained using the expectation operators in Figure 3.7. The uniformly weighted results featured in Figure 3.7a correspond to traditional techniques and establish a baseline for comparison. We observe how Figure 3.7b shows how that we place increased emphasis on the early time regime of the sum-squared-tracking error. We use three different values of $\gamma$ to demonstrate how varying $\gamma$ impacts the shape of the tracking error trajectory (larger $\gamma$ values induce more weighting of the initial time regime). Hence, the results obtained with $\gamma = 0.1$ achieve a low tracking error over the first 120 seconds, but then incur large deviations afterward since the later time regime has effectively no impact on the objective.

Finally, the results obtained using the $\mathbb{V}$-$\mathbb{E}_t$ and $\text{CVaR}_t$ measure operators are sum-

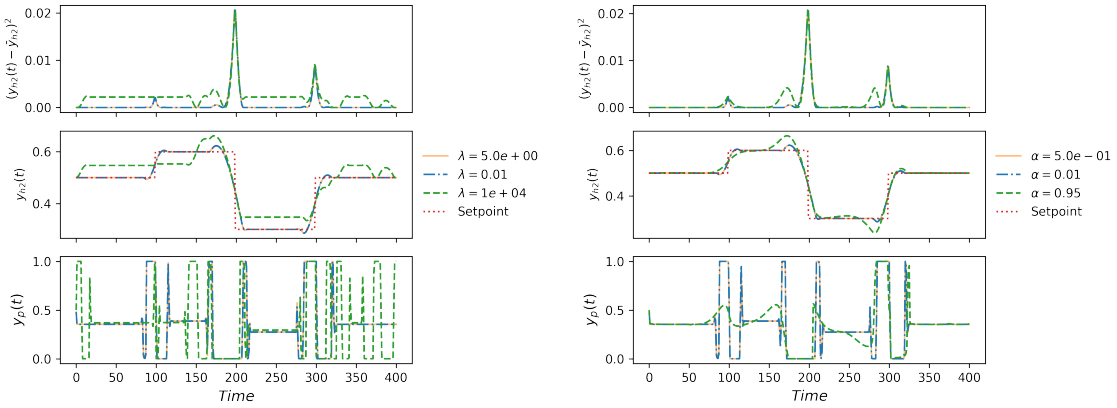



(a) Uniform

(b) Exponential

Figure 3.7: The optimal trajectories using the $\mathbb{E}_t$ measure operator. The exponential pdf places more emphasis in reducing the tracking error at the early times as the value of $\gamma$ is increased.

marized in Figure 3.8. In Figure 3.8a, we see how increasing $\lambda$ is able to dampen the fluctuation in the tracking error to a minor extent. However, this comes at the cost of increasing the tracking error at certain times; thus, this highlights how the uniform fluctuation dampening exhibited by mean-variance measures is not well-suited for certain applications. In comparison, varying the value of $\alpha$ in the CVaR measure has very little effect. This likely occurs because this problem entails very little flexibility in how the tracking error trajectory can be shaped. Hence, using alternative measures like CVaR will not always make a significant difference in shaping optimal trajectories of particular problems. Note the small degradations in tracking error performance observed with $\alpha = 0.95$ can likely be attributed to the large error peak at $t = 200$ dominating the objective.




(a) $\mathbb{V}$-$\mathbb{E}_t$ (b) $\text{CVaR}_t$

Figure 3.8: The optimal trajectories using the $\mathbb{V}$-$\mathbb{E}_t$ and $\text{CVaR}_t$ measure operators. They have a less dramatic impact on the results that what we observe in Section 3.5.1 since this problem exhibits less flexibility in how the trajectories can be shaped.

# Chapter 4

## RANDOM FIELD OPTIMIZATION

The content of this chapter is originally presented in [30, 91].

## 4.1 Introduction

Many complex systems involve variables that are defined over continuous domains (e.g., space and/or time), making such variables infinite-dimensional (such variables are functions). These systems are difficult to model and optimize since they involve sophisticated mathematical objects such as partial differential equations (PDEs), derivatives, and integrals. Dynamic optimization and PDE-constrained optimization denote common optimization modeling paradigms that involve such objects and these paradigms can be more broadly classified as classes of infinite-dimensional optimization (InfiniteOpt) problems [30]. These problem classes arise in general research domains such as dynamic system identification [14], model predictive control [8], process intensification [92], and path planning [93]. Specific applications of such research domains include autonomous vehicles [94], process systems [95], diffusion processes [96], microbial communities [14], structural design [97], molecular dynamics [98], and reaction systems [63].

The complexity of infinite-dimensional systems is significantly exacerbated when these are impacted by uncertainty. Here, uncertainty can arise from a wide variety of sources

that might involve random variables that live in infinite-dimensional spaces such as environmental disturbances (e.g., wind and temperature), uncertain fields (e.g., porosity), forecasts (e.g., time series), and molecular-level effects (e.g., random fluctuations) [99]. Incorporating these elements is key in engineering and scientific applications, but this practice has been limited in the context of optimization. One can attribute this to the fact that infinite-dimensional random variables involve complex mathematical objects such as stochastic differential equations and space-time kernel functions. Specifically, such uncertainties typically represent stochastic processes (sometimes referred to as random processes) which model uncertainties that evolve over time [100]. These have seen wide application in physics [101], biology [102], chemistry [101], neuroscience [103], image processing [104], and engineering [63]. Classical models of stochastic processes are Wiener processes (i.e., Brownian motion), Markov processes, and Poisson processes [100].

Furthermore, stochastic processes that model system uncertainty/randomness can be incorporated into system modeling relations via stochastic differential equations (SDEs) to yield stochastic models [105]. Stochastic differential equations capture the evolution of a stochastic variable, whose behavior is characterized by the evolution of a probability density function. SDEs denote an extensively studied special case where a classical differential equation is perturbed by a random noise term (often a Wiener process) that models the randomness introduced in a dynamical system [106]. For instance, the diffusion of particles across a surface can be modeled via an SDE using a Wiener noise process and its solution (in terms of the resulting probability density function) yields the so-called Fokker-Planck equations for transient diffusion on a planar surface [63]. Similarly, stochastic chemical kinetics (i.e., reactions occurring at low concentrations such that random molecular behavior is prominent) can be modeled as SDEs in which the extents of reaction are characterized as Poisson processes; the solution of this system of SDEs yields the so-called Chemical Master Equations for a general reaction network (i.e., the forward Kolmogorov equation) [63]. These examples help illustrate the significance of using stochastic models to describe dynamical systems. A key caveat of SDE mod-

els is that the underlying stochastic process does not have a closed-form solution and thus necessitates the use of stochastic calculus and/or simulation techniques to obtain the system evolution [106]. However, there are a variety of software tools available to accomplish this task such as `DifferentialEquations.jl` [107], `SDE Toolbox` [108], and `SDE-MATH` [109]. We also note that the scope of stochastic processes and SDEs is generally limited to systems that evolve over time domains (as opposed to space-time domains).

Multi-stage stochastic optimization is a widely used modeling paradigm to make decisions under uncertainties (typically based on a stochastic process) that evolve over time. Such formulations employ discrete-time decision stages where each stage seeks to optimize a random cost function that is typically the conditional expectation given observed information from previous time stages [110]. Here, the time evolution of uncertainty is modeled via a scenario tree which seeks to enumerate the possible outcomes over the defined time stages; such trees can be derived by reducing an ensemble of realizations of a stochastic process that describes the underlying system uncertainty. Unfortunately, scenario trees grow exponentially with the number of stages [111]. Diverse schemes have been developed to circumvent this complexity, such as the so-called stochastic dual dynamic programming (SDDP) algorithm [112], which has been implemented in software tools such as `SDDP.jl` [113]. Multi-stage stochastic programming provides a natural framework to capture time-dependent uncertainties, but the representation of such uncertainties is inherently finite-dimensional (over discrete times). A key question that arises in this context is thus: how can we model stochastic optimization problems that involves uncertainties that propagate over continuous time? Moreover, how can we characterize uncertainty that propagates over other domains (e.g., space and space-time)?

Random field theory provides a powerful mathematical framework for characterizing uncertain factors that evolve over continuous (infinite) domains such as space and time. Under this paradigm, uncertainty is modeled as a random function/manifold/-field [114]. Here, each realization of a random field is a deterministic function defined over the infinite domain and there is an underlying probability density that character-

izes the probabilities of such realizations. This general representation captures a wide range of uncertainties such as stochastic processes (domain is continuous time), Gaussian processes (domain is space, time, or space-time), point processes (domain collapses to finite set of points), and random vectors/matrices (a field defined over a finite set of points). Random field models have been extensively adapted and developed in diverse disciplines which include: physics [114], neuroscience [115], computer graphics [116], geoscience [117, 118], and civil engineering [97]. This rich theoretical base has enabled advanced tools for stochastic modeling and analysis; for instance, random field models have been used for identifying abnormal behavior in brain imaging data (modeled as space-time random fields) [119]. Surprisingly, however, random fields have seen limited use in optimization. Moreover, random field theory provides tools that facilitate the characterization of behavior (e.g., excursion probabilities) and has fundamental connections with other fields of mathematics such as topology. To our best knowledge, random fields have only been used to model uncertainty of material properties in the context of topological optimization in civil engineering [97]. Moreover, random fields (Gaussian processes) have recently been used in the context of Bayesian optimization for machine learning and to build surrogate models in model predictive control, but such work does not account for uncertainties that evolve over continuous space-time domains.

In the context of our unifying abstraction for InfiniteOpt problems presented in Chapter 2, we observe that random field theory allows us to incorporate dependencies of a random parameter over other infinite domains (e.g., account for stochastic processes). The particular class of infinite variables discussed in previous chapters considered a random parameter $\xi$ that is invariant over other domains. This means that, although the infinite variables in this class of formulations can be classified as random fields, the input uncertainty $\xi$ is taken to be *static* random parameter that does not capture any spatial or temporal correlation. We can overcome this modeling limitation by extending our abstraction to consider infinite parameter functions (e.g., random parameters that are functions of other infinite parameters). For instance, we can define a time-dependent infinite pa-

rameter function $\xi(t) \in \mathcal{D}_{\xi(t)},\ t \in \mathcal{D}_t$ which denotes a known random field whose sample domain $\mathcal{D}_{\xi(t)}$ is a set of temporal functions. With this extension, we can now define the infinite variable $y(t, \xi(t))$, which denotes a random field variable (an infinite variable) where $\xi(t)$ is a known random field that can capture correlations of $\xi$ over time $t$. This important observation enables us to connect a wide breadth of optimization disciplines to random field theory.

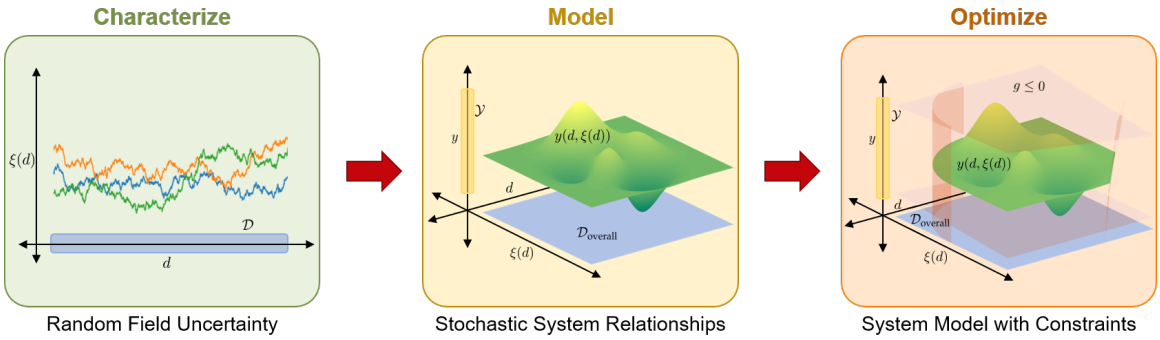



Figure 4.1: A visual summary of our proposed random field optimization modeling framework.

In this chapter, we formalize the incorporation of random field theory into InfiniteOpt problems, establishing a new modeling paradigm that we call *random field optimization* (RFO). This modeling paradigm aims to unify and facilitate the development of models that capture uncertainty that evolves over general continuous domains. Our proposed framework characterizes such uncertainties via random fields and aims to integrate associated mathematical objects (e.g., SDEs and kernel functions) directly into optimization formulations. Figure 4.1 summarizes this approach. RFO provides an intuitive framework that will serve as the foundation for future research in modeling, analyzing, and solving this new general class of InfiniteOpt problems. Moreover, this abstraction can enable further development of software tools such as `InfiniteOpt.jl` to make the optimization of infinite-dimensional systems more accessible.

## 4.2 Random Fields

In this section, we detail relevant definitions, properties, characterizations and metrics from random field theory and establish basic notation. The material covered here is not intended to be a comprehensive review; we refer the reader to [119, 120, 121] for a thorough introduction and review of random field theory.

### 4.2.1 *Definition*

Random field theory generally refers to the study of random functions defined over Euclidean spaces (e.g., space and/or time) where any analysis/properties considered for deterministic functions is also of interest (making this a broad subject whose complexity is compounded by the introduction of random behavior) [119]. A random field denotes a random function $\xi : \mathcal{D} \mapsto \mathcal{D}_{\xi(d)}$ whose input domain $\mathcal{D} \in \mathbb{R}^n$ is a general topological space:

$$\xi(d) \in \mathcal{D}_{\xi(d)},\ d \in \mathcal{D} \tag{4.1}$$

where $\mathcal{D}_{\xi(d)}$ is a function space that is the co-domain of the random field (i.e., its realizations are deterministic functions $\hat{\xi}(d) : \mathcal{D} \mapsto \mathbb{R}^{n_\xi}$). Here, a realization of the random field $\hat{\xi}(d) \in \mathcal{D}_{\xi(d)}$ (also called a sample function) is a deterministic function, as illustrated in Figure 4.2.

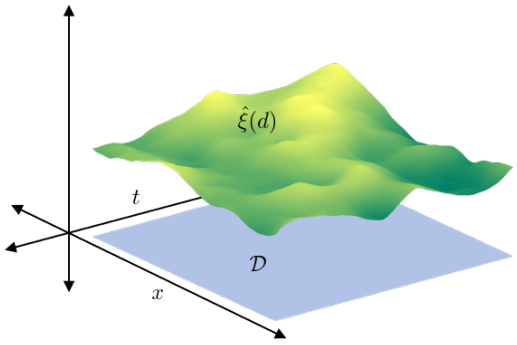



Figure 4.2: A depiction of a particular realization $\hat{\xi}(d)$ of a random field $\xi(d)$.

Random field theory is a flexible paradigm that is amenable to diverse engineering applications such as petroleum reservoir management, diffusive surfaces, fluid dynamics, atmospheric modeling, granular and porous materials, stochastic chemical kinetics, and molecular dynamics. Modeling the height of the ocean surface over a certain spatial area provides an intuitive example of a random field. Here, a realization denotes a snapshot that describes a possible ocean surface with defined waves. Moreover, we can add a temporal dependency such that a realization describes how a certain possible set of waves propagate over the surface over time. A key observation here is that the surface height exhibits correlation over space and time and such correlations can be captured using models proposed in random field theory.

Following the terminology from the recently-proposed unifying abstraction for InfiniteOpt problems, we use $d$ to denote an infinite parameter (e.g., space and time) that indexes the infinite domain $\mathcal{D}$ (typically referred to as a topological manifold in the random field literature) [30, 114]. Equivalently, we can interpret a random field $\xi(d)$ as a set of random variables $\xi_d$ indexed over the domain $\mathcal{D}$:

$$\{\xi_d : d \in \mathcal{D}\}. \tag{4.2}$$

Moreover, these are defined with a set of density functions $\{F_{d_1}, \ldots, F_{d_n}\}$ defined on the Borel space $\mathscr{B}^{nm}$ that satisfy:

$$F_{d_1}, \ldots, F_{d_n}[B] = P[(\xi(d_1), \ldots \xi(d_n)) \in B],\ B \in \mathscr{B}^{nm} \tag{4.3}$$

where $m$ is the dimension of $\mathcal{D}_{\xi(d)}$ and $n$ is an arbitrary positive integer. These are often referred to as the finite-dimensional densities of a random field and are central in establishing fundamental properties for a random field, such as those presented in Section 4.2.2.

Following (4.2), a random field can be interpreted as the *generalization of a multi-variate random variable* where the indexing set is continuous (infinite-dimensional). For instance,

a discrete time-series $\{\xi_t, t \in \mathcal{D}_t\}$ is a multivariate random variable and this can be generalized as a temporal random field when $\mathcal{D}_t$ becomes a continuous time interval. In this work, we will favor the use of (4.1), since function notation is more amenable to infinite-dimensional system models (which is the focus of this work). Note, however, that the term infinite-dimensional arises from the observation that system variables/parameters are indexed over a space $\mathcal{D}$ that exhibits infinite cardinality, as exemplified in Equation (4.2).

### 4.2.2 *Properties*

We begin by establishing the concept of mean and covariance functions of a random field, which are basic constructs needed to derive additional random field properties. These can be thought of as continuous generalizations of mean vectors and covariance matrices used in random matrix theory to infinite domains.

The mean function of a random field $\mu(d)$ is a deterministic function that denotes the expected value over the random field manifold $\mathcal{D}$:

$$\mu(d) := \mathbb{E}_{\xi(d)}[\xi(d)]. \tag{4.4}$$

The covariance function $\Sigma : (\mathcal{D}, \mathcal{D}) \mapsto \mathbb{R}$ is a deterministic function and captures the correlation/variation exhibited over the random field manifold:

$$\Sigma(d, d') := \mathbb{E}_{\xi(d)}\left[(\xi(d) - \mu(d))(\xi(d') - \mu(d'))\right]. \tag{4.5}$$

We can ready show that, if $\mathbb{E}_{\xi(d)}[\xi(d)^2]$ is finite for all $d \in \mathcal{D}$, then $\Sigma(d, d')$ is finite for all $d, d' \in \mathcal{D}$; moreover, $\Sigma(d, d')$ is assumed to be a non-negative definite function [119]. We will see in Section 4.2.3 that the mean and covariance functions are often useful in defining (Gaussian) random fields.

A common consideration in modeling random fields is that of *homogeneity*. A ran-

dom field is said to be homogeneous if its finite-dimensional densities are translationally invariant. Formally, this indicates that the joint density of any $k$ random variables $\{\xi(d_1), \ldots, \xi(d_k)\}$ is equivalent to that of the random variables $\{\xi(d_1 + \tau), \ldots, \xi(d_k + \tau)\}$ for arbitrary $\tau \in \mathbb{R}$. A homogeneous random field thus exhibits a constant mean function and its covariance function is only a function of the difference $d - d'$:

$$\begin{aligned} \mu(d) &= \beta \\ \Sigma(d, d') &= \Sigma(d - d') \end{aligned} \tag{4.6}$$

where $\beta \in \mathbb{R}^{n_\xi}$ is a constant. Subsequently, a random field is said to be weakly homogeneous or stationary if it only satisfies (4.6).

Another key consideration (that will be particularly relevant in Section 4.3) pertains to the continuity and the differentiability of random fields. Continuity and differentiability are key in capturing behavior because such properties naturally induce correlation; for example, these properties indicate that the value of a random field at a space-time location is not arbitrarily far away from that at a neighboring point. There are a considerable number of definitions used in the literature to derive and discuss these properties. A common group of definitions, that are convenient for our purposes, define the continuity and differentiability of random fields with respect to their sample functions (realizations). A random field is sample function continuous at a point $d^*$ if for every sequence $\{d_n\}$ that satisfies $||d_n - d^*|| \to 0$ as $n \to \infty$ we have that:

$$P\left\{ ||\xi(d_n) - \xi(d^*)|| \to 0 \ \text{ as } n \to \infty \right\} = 1. \tag{4.7}$$

This essentially amounts to almost surely having sample functions that are continuous at a point $d^*$. Similarly, we say a random field is sample function differentiable at a point $d^*$ if we have that:

$$P\left\{ \frac{\partial \xi(d)}{\partial d_\ell} \text{ exists} \right\} = 1 \tag{4.8}$$

where $d_\ell$ is the $\ell^{\text{th}}$ element of $d$. Hence, we can define a random field as continuous and/or differentiable if these conditions hold almost surely for its (deterministic) sample functions.

### 4.2.3 *Characterization*

Since random field theory broadly provides a framework for random functions, there exist numerous ways to model/characterize them. For an intuitive initial step, let us consider a simple analytic function $\xi(d;\omega)$ with a parametric dependence on a random variable $\omega$. Figure 4.3 depicts such a function, where $\xi(x;\omega) = \exp(x^\omega)$ denotes a random field. In practice, however, such functions can be modeled directly with random variables and need not be treated rigorously as random fields.

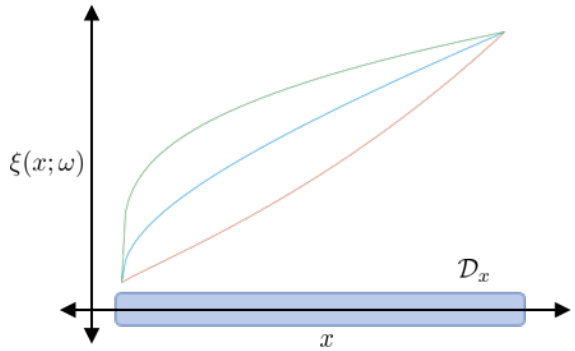



Figure 4.3: A depiction of realizations $\hat{\xi}(x;\hat{\omega})$ from a simple random function $\xi(x;\omega)$ which is a random field over spatial position $x \in \mathcal{D}_x$.

The most prevalent random field characterizations correspond to Gaussian random fields, which can be completely specified via their second-order statistics (i.e., a mean and covariance functions). More precisely, a random field is Gaussian if all of its finite-dimensional density functions are multi-variate Gaussian (this generalizes Gaussian processes over a general topological domain $\mathcal{D}$). This unique property enables a wealth of analysis and modeling techniques, since the mean and covariance are available in convenient analytical forms for multi-variate Gaussian distributions.

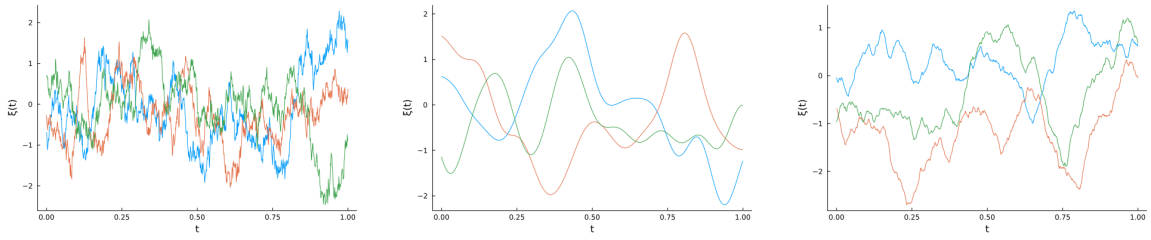
Without a loss of generality, the mean function is often assumed to be zero, such that

the field is fully characterized via the choice of the covariance function. The prevalence of Gaussian random fields/processes in a wide variety of disciple areas (e.g., data science, statistics, and engineering) has motivated extensive development of models of covariance functions (often referred to as kernel functions). Common covariance function choices include linear $\Sigma_{\mathrm{L}}(d, d')$, squared exponential $\Sigma_{\mathrm{SE}}(d, d')$, Ornstein-Uhlenbeck $\Sigma_{\mathrm{OU}}(d, d')$ (also called the exponential kernel), and Matérn $\Sigma_{\mathrm{M}}(d, d')$:

$$
\begin{aligned}
\Sigma_{\mathrm{L}}(d, d') &:= d^T d' \\
\Sigma_{\mathrm{SE}}(d, d') &:= \sigma \exp\left(-\frac{||d - d'||^2}{2\beta^2}\right) \\
\Sigma_{\mathrm{OU}}(d, d') &:= \sigma \exp\left(-\frac{||d - d'||}{\beta}\right) \\
\Sigma_{\mathrm{M}}(d, d') &:= \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||d - d'||}{\beta}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}||d - d'||}{\beta}\right)
\end{aligned}
\tag{4.9}
$$

where $\sigma, \beta \in \mathbb{R}$ are scalar parameters, $K_\nu$ is a modified Bessel function with order $\nu \in \mathbb{R}$, and $\Gamma(\nu)$ is the gamma function [122]. Selecting a particular covariance function models amounts to enforcing a prior on the random field that will affect the fundamental behavior of such field. For example, the Ornstein-Uhlenbeck function (which corresponds to Brownian motion) is nowhere differentiable, while the squared exponential function gives rise to smooth field realizations that are almost surely differentiable. Figure 4.4 illustrates this for various covariance functions. In addition to differentiability, we observe how the choice of the covariance function influences other field properties; for instance, covariance functions that only depend on the difference $d - d'$ are stationary and those that only depend on the distance $||d - d'||$ are stationary and isotropic (i.e., they have no directional dependence). There exist diverse software implementations of Gaussian random fields such as `GaussianRandomFields.jl` in `Julia`, `RandomFields` in `R`, and `GenPak` in `C++` that make it straightforward to build Gaussian fields and obtain sample functions. We refer the reader to [123] for an in-depth review of modeling Gaussian random fields.

Non-Gaussian field models have also been proposed in the literature, which include



(a) Ornstein-Uhlenbeck (b) Squared Exponential (c) Matérn

Figure 4.4: Realizations of Gaussian random fields with varied covariance functions; where appropriate we use $\beta = 0.1$ and $\nu, \sigma = 1$. Note how the smoothness of the realizations are affected by the form of the covariance function.

$\chi^2-$, $t-$, and $F-$ random fields [124]. These are constructed via a linear combination of Gaussian random fields that are independently and identically distributed [121]. For example, we can construct a $\chi^2-$ random field $\xi_{\chi^2}(d)$ via the linear combination:

$$\xi_{\chi^2}(d) = \sum_{i \in \mathcal{I}} \xi_{\mathrm{G},i}(d) \tag{4.10}$$

where $\xi_{\mathrm{G},i}(d)$, $i \in \mathcal{I}$, are i.i.d. Gaussian random fields with a zero mean and unit variance. Such fields can help capture a wider range of random field behavior. Moreover, we can construct random fields (and their associated covariance functions) via statistical analysis of the ensemble solution to an underlying system of RDEs (or SDEs) to model the uncertainty of interest [120, 125]. This typically requires the use of stochastic calculus, but there are tools such as `DifferentialEquations.jl` that can be used to obtain accurate numerical ensembles of RDE/SDEs [107].

### 4.2.4 *Measures*

Random fields are complex modeling objects that often need to be projected/reduced via measures in order to be analyzed [30]. Specifically, a measure will summarize the random field into a deterministic function or quantity. For instance, the expectation operator $\mathbb{E}_{\xi(d)}$ shown in (4.4) is a measure that summarizes the random field into a deterministic mean function $f(d)$ that only depends on the domain $\mathcal{D}$. Moreover, we could employ

more general measure operators $M_d$ (e.g., integral over the domain) to further reduce the mean function to a scalar value. Hence, the use of measure operators helps us develop summarizing statistics and quantities that help analyze the behavior of a random field. Such summarizing statistics are also key in formulating optimization problems, as such problems often require the representation of objectives and constraints as deterministic quantities.

One of the most prevalent measures used in random field theory is the so-called *excursion probability* (also known as the exceedance probability):

$$P_{\tilde{\zeta}(d)} \left\{ \sup_{d \in \mathcal{D}} \tilde{\zeta}(d) \geq u \right\} \tag{4.11}$$

where $u \in \mathbb{R}$ is a thresholding parameter, and we take $\tilde{\zeta}(d)$ to be an $\mathbb{R}$-valued field (this can be extended to vector valued fields, but that is beyond the scope of this work [126]). The measure (4.11) is at the focus of most of the random field literature [114]. This is because this measure summarizes the excursion behavior of a random field and thus helps characterize extreme values (i.e., the peaks of a function), which is useful in many applications. For instance, the excursion probability is used for hypothesis testing in neuroscience, geoscience, civil engineering and other fields to determine significant deviations relative to a certain threshold. Other potential uses include quantifying the likelihood that a field surpasses a certain limit/tolerance which has implications for quality control, computing escape probabilities and times (e.g., in molecular dynamics), and extreme event quantification.

To better understand (4.11) and discuss ways to compute it, it is necessary to define the notion of an excursion set $A_u$ for arbitrary function $f$:

$$A_u(f) := \{d \in \mathcal{D} : f(d) \geq u\}. \tag{4.12}$$

The excursion set is the subdomain of $\mathcal{D}$ where $f$ exceeds $u$. Figure 4.5 illustrates such a set for a 1D function. Here, we observe that the excursion set of a 1D function can

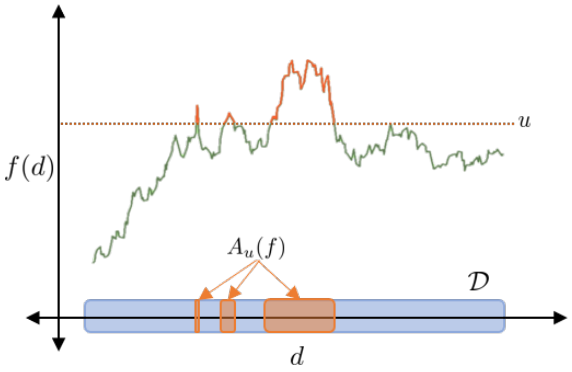



Figure 4.5: A depiction of a 1D excursion set $A_u(f)$ on an arbitrary function $f(d)$.

be characterized by a number of disjoint intervals (subdomains) in $\mathbb{R}$. This means that its topology can be described by the number of upcrossings that occur. This observation is the basis for extensive theoretical development for 1D random fields (e.g., stochastic processes over time) [127]. From this, a number of closed-form representations of (4.11) have been developed; for example, Rice's formula is commonly used for computing the expected number of upcrossings (approximating (4.11)) for a zero-mean, stationary, continuous Gaussian field on $d \in [0, 1]$):

$$\mathbb{E}[N_u] = \frac{\sqrt{\lambda}}{2\pi\sigma} \exp\left(\frac{-u^2}{2\sigma^2}\right) \tag{4.13}$$

where $N_u$ denotes the number of upcrossings, $\sigma^2 = \mathbb{E}_{\xi(0)}[|\xi(d)|^2] = \Sigma(d)$ is the covariance function evaluated at $d = 0$, and $\lambda = -\Sigma''(0)$ is the second derivative of the covariance function evaluated at $d = 0$ [114].

The notion of upcrossings is ill-defined for (random) functions with $|\mathcal{D}| > 1$, making the analysis of high-dimensional excursion sets/probabilities significantly more challenging. However, in these cases, it is possible to use the expected Euler characteristic (EC) $\mathbb{E}[\mathcal{X}(A_u(\xi(d)))]$ as an approximation. Moreover, it can be shown that, under certain conditions, the error of this approximation decays exponentially with increased $u$; this makes the expected EC a useful construct used for characterizing the behavior of random fields [114].

The EC measures the topology/geometry of excursion sets (e.g., measuring the number of disjoint intervals in 1D deterministic sets and the number of connected components minus the number of holes for 2D deterministic sets) and capture the critical points (i.e., extrema) of a field since they define its topological features [128]. We refer the reader to [128] for an intuitive introduction to computing the EC of field excursions.

For certain types of random fields,there exist analytic representations of the expected EC characteristic [119]; when these are not available, the expected EC can be computed via Monte Carlo (MC) sampling using the samples $\{\hat{\xi}_k(d) : k \in \mathcal{K}\}$:

$$\mathbb{E}_{\xi(d)}[\mathcal{X}(A_u(\xi(d)))] \approx |\mathcal{K}|^{-1} \sum_{k\in\mathcal{K}} \mathcal{X}(A_u(\hat{\xi}_k(d))). \tag{4.14}$$

We can also approximate the excursion probability in (4.11) directly via MC sampling [129] as:

$$P_{\xi(d)}\left\{\sup_{d\in\mathcal{D}} \xi(d) \geq u\right\} = \mathbb{E}_{\xi(d)}\left[\mathbb{1}\left(\sup_{d\in\mathcal{D}} \xi(d) \geq u\right)\right] \approx |\mathcal{K}|^{-1} \sum_{k\in\mathcal{K}} \mathbb{1}\left(\sup_{d\in\mathcal{D}} \hat{\xi}_k(d) \geq u\right) \tag{4.15}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

There are a number of alternative measures (metrics) used in the random field literature, but these are beyond the scope of this work. We refer the reader to [114] for more details.

## 4.3 Modeling with Random Field Uncertainty

In this section, we discuss how to incorporate random fields into mathematical models commonly encountered in applications. We also overview how such stochastic models can be transformed into finite representations that are amenable for simulation and/or optimization. Much of this discussion includes a general overview of random and stochastic differential equation theory; we refer the interested reader to [63, 99, 105, 130] for a rigorous treatment of these topics.

### 4.3.1 *Characterizations*

Once we have a random field representation of the uncertainty that affects a given system, we can incorporate it into the governing equations (e.g., differential and algebraic equations) to yield a stochastic model that we can simulate and/or use in an optimization formulation (which we discuss in Section 4.4). We refer to governing equations that embed random field components as random differential-algebraic equations (RDAEs). A key observation that we start with is that a function that depends on a random field is itself a random field (i.e., $f(\xi(d))$ is a random field for arbitrary $f$). Thus, as we construct algebraic and differential equations that incorporate a random field $\xi(d)$ they become inherently random in nature. The resulting stochastic model can be interpreted as a characterization for a high-dimensional random field model which further highlights the connection between random field theory and random differential equations, which is the focus of this section [125].

#### *Random Differential-Algebraic Equations*

In the most general sense, we can define our model by directly embedding random field uncertainty into the DAEs that describe a system giving rise to a set of RDAEs. We first consider system (state) variables $y : (\mathcal{D}, \mathcal{D}_{\xi(d)}) \mapsto \mathcal{Y} \subseteq \mathbb{R}^{n_y}$ which are functions over the system domain $\mathcal{D}$:

$$y(d, \xi(d)) \in \mathcal{Y} \tag{4.16}$$

where we explicitly show their dependence on $\xi(d)$ to emphasize their inherent randomness. Figure 4.6 depicts such a modeling variable.

Following our approach in [30], we define the differential operator $D : \mathcal{D} \mapsto \mathcal{D}$ that operates on variables $y(d, \xi(d))$:

$$Dy(d, \xi(d)) \in \mathbb{R}^{n_D} \tag{4.17}$$

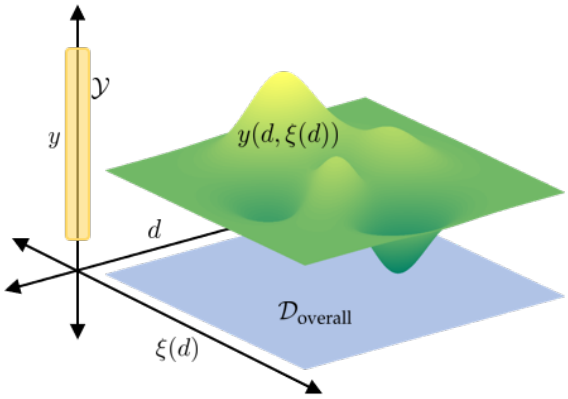



Figure 4.6: A high-level illustration of a random field variable $y(d, \xi(d))$.

which encapsulates entities such as partial derivatives $\partial y(d, \xi(d))/\partial d_\ell$, Laplace operators $\nabla^2 y(d, \xi(d))$, and stochastic differentials $dy(d, \xi(d))$. With these objects we can now define general random DAEs of the form:

$$g(Dy, y(d, \xi(d)), \xi(d), d) = 0 \tag{4.18}$$

where $g(\cdot) \in \mathbb{R}^{n_g}$ are vector-valued functions. This form is general and captures a large collection of application systems. An important note is that classical differential operators like $\partial/\partial d_\ell$ can only be used if $\xi(d)$ is sample function differentiable in accordance with (4.8). Otherwise, (4.18) needs to be carefully setup with SDEs derived from appropriate stochastic calculus constructs (e.g., Itô or Stratonovich calculus) [63, 105].

We also note that recently, concerted efforts have been made to develop theory for random DAEs (especially in the context of SDEs). However, these have been limited to a few prescribed forms thus far. Relevant recent works for further reading on this topic include [131, 132].

### *Random Differential Equations*

Due to the complexity associated with stochastic differential equations, theory is not well-developed for general random systems and the literature commonly considers a set

of RDEs of the form:

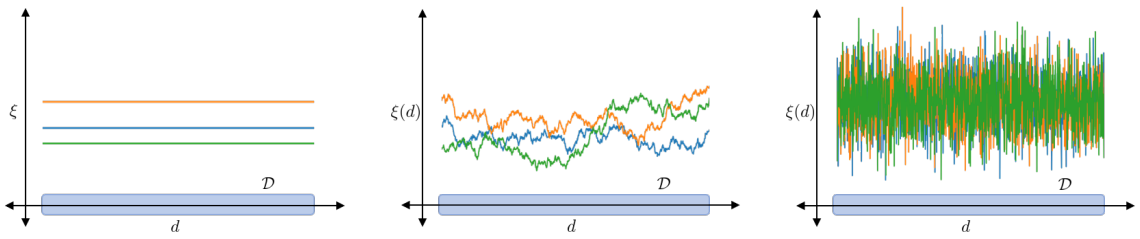
$$\begin{aligned} Dy(d,\xi(d)) &= f(y(d,\xi(d)),\xi(d),d) \\ q(y(d_0,\xi(d_0)),\xi(d_0),d_0) &= 0,\ d_0 \in \mathcal{D}_0 \end{aligned} \tag{4.19}$$

where $f : (\mathcal{Y}, \mathcal{D}_{\xi(d)}, \mathcal{D}) \mapsto \mathbb{R}^{n_d}$ is an arbitrary right-hand side function, $q : (\mathcal{Y}, \mathcal{D}_0) \mapsto \mathbb{R}$ encodes the initial/boundary conditions, and $\mathcal{D}_0$ is the set of initial and/or boundary points [105, 99]. Three types of RDEs are classified in the literature which are distinguished by the way randomness (characterized by $\xi(d)$) enters the equation; these are (in order of increasing difficulty to analyze and solve) random initial/boundary conditions, random inputs (via an inhomogeneous random term), and random coefficients [105].

Random initial/boundary conditions can be described by the special case where the underlying uncertainty is static $\xi(d) = \xi$ (if needed, implicit dependencies on $\xi$ can be converted into random initial/boundary conditions via augmenting the system of differential equations [105]):

$$\begin{aligned} Dy(d,\xi) &= f(y(d,\xi),d) \\ q(y(d_0,\xi),\xi,d_0) &= 0,\ d_0 \in \mathcal{D}_0. \end{aligned} \tag{4.20}$$

Figure 4.7 illustrates this distinction for realizations of different random field classifications. These pertain to cases where the underlying uncertainty can be described by a random variable instead of a random field. This often occurs in applications with uncertain parameters that are domain invariant (e.g., uncertain kinetic rate constants).




(a) Static (b) Correlated (c) Uncorrelated

Figure 4.7: Realizations of random field uncertainty $\xi(d)$ stemming from three distinct field types. Notice the static case corresponds to $\xi(d) = \xi$ and can be classified as a random variable (having no dependence on $d$).

RDEs that feature inhomogeneous random field input $h(d)\xi(d)$ can be denoted as:

$$\begin{aligned} Dy(d,\xi(d)) &= f(y(d,\xi(d)),d) + h(d)\xi(d) \\ q(y(d_0,\xi(d_0)),d_0) &= 0,\ d_0 \in \mathcal{D}_0 \end{aligned} \tag{4.21}$$

where $h : \mathcal{D} \mapsto \mathbb{R}$ is a deterministic $\mathbb{R}$-valued function. These typically denote deterministic differential equations $Dy(d) = f(y(d),d)$ that are subjected to random noise $h(d)\xi(d)$. For dynamic systems, these reduce to the well-studied area of SDEs and are typically represented:

$$\begin{aligned} dy(t,\xi(t)) &= f(y(t,\xi(t)),t)dt + h(t)d\xi(t) \\ y(0,\xi(0)) &= y_0 \end{aligned} \tag{4.22}$$

where $y_0 \in \mathbb{R}^{n_y}$ is the initial value and $d\xi(t) := \xi(t+dt) - \xi(d)$ is the differential of the random field $\xi(t)$ (a stochastic process) which is defined in accordance with an appropriate stochastic integral [63, 130]. Here, $\xi(t)$ typically denotes a Wiener process (i.e., white noise or Brownian motion), but can be generalized to any process that is a semi-martingale (i.e., it can be represented as a sum of local martingale processes) which includes sample function differentiable, Wiener, and Poisson processes [130]. Such SDEs have been behind important applications such as the random diffusive and kinetic systems discussed in Section 4.1.

The differential equation in (4.22) can be ambiguous and is generally understood to denote the integral equation:

$$y(t+s,\xi(t+s)) - y(t,\xi(t)) = \int_t^{t+s} f(y(t,\xi(t)),t)dt + \int_t^{t+s} h(t)d\xi(t) \tag{4.23}$$

where the first integral is a classical Riemann or Lebesgue integral and the second is a stochastic integral (typically an Itô or Stratonovich integral). Itô integrals are the most common choice since they integrate in a non-anticipative manner (making them well-suited for time evolutions) [133], while Stratonovich integrals are better suited for general topological manifolds (e.g., the input domains of general random fields) [134].

Finally, the most difficult types of Equation (4.19) correspond to ones where $\xi(d)$ is used as a random coefficient. These encompass a wide variety of RDEs which compounds the complexity in analysis to establish properties (e.g., the existence and uniqueness of solutions); however, these have been of increased interest in mathematical research [99]. An illustrative application would be a transient diffusive process that is subject to spatially random diffusivity $\xi(x)$:

$$\begin{aligned} \frac{\partial y_c(t,x,\xi(x))}{\partial t} &= \xi(x)\nabla_x^2 y_c(t,x,\xi(x)) + y_g(t,x) \\ y_c(0,x,\xi(x)) &= y_{c,t0} \\ y_c(t,x_0,\xi(x_0)) &= y_{c,x0},\ x_0 \in \mathcal{D}_{x0} \end{aligned} \tag{4.24}$$

where $y_c(\cdot)$ denotes the concentration/temperature, $y_g(\cdot)$ is a deterministic generative/reactive term, $y_{c,t0} \in \mathbb{R}$ is the initial concentration/temperature field, and $y_{c,x0} \in \mathbb{R}$ are the boundary concentration/temperature values. Such a system is the focus of the case study presented in Section 4.5.3.

### 4.3.2 *Stochastic Simulation*

Now that we can characterize random system models following our discussion in Section 4.3.1, let us establish how these characterizations can be simulated via finite transformations. The idea behind converting infinite-dimensional models into finite-dimensional forms via transformation follows from the unifying abstraction for InfiniteOpt problems proposed in [30]. Typical transformations can be broken down into three groups: MC, method of weighted residuals (MWR), and probability density function (pdf) solutions. Our focus in this work will be on MC, since these techniques are frequently used in numerical tools and are the most intuitive for incorporation into optimization formulations. However, we also provide some discussion with respect to MWR and pdf techniques for completeness.

***Monte Carlo***

We begin by considering the simulation of (4.20) in the special case that the underlying random field uncertainty can be expressed as a random variable $\xi(d) = \xi$ (i.e., we have a random initial/boundary condition model). With MC samples $\{\hat{\xi}_k : k \in \mathcal{K}\}$, we can express (4.20) as a system of $|\mathcal{K}|$ deterministic equations:

$$\begin{aligned} D\hat{y}_k(d) &= f(\hat{y}_k(d), d),\ k \in \mathcal{K} \\ q(\hat{y}_k(d_0), \hat{\xi}, d_0) &= 0,\ d_0 \in \mathcal{D}_0, k \in \mathcal{K} \end{aligned} \tag{4.25}$$

where $\hat{y}_k(d)$ denote the sample functions of the model random field $y(d, \xi)$ that correspond to an MC sample $\xi_k$. These denote standard deterministic differential equations which can be simulated via classical methods (e.g., finite difference or orthogonal collocation over finite elements). We refer the reader to [63, 135] for a review of these deterministic solution methods. In practice, these can be simulated numerically with software packages such as `DifferentialEquations.jl` in `Julia` or `odeint` in `Python`. The ensemble of MC solutions $\{\hat{y}_k(d) : k \in \mathcal{K}\}$ can then be used to approximate the moments of the random field solution $y(d, \xi(d))$ following the approaches detailed in [120] as described in Section 4.2.3.

A straightforward MC simulation approach for general RDAEs in (4.18) involves drawing sample function realizations $\{\hat{\xi}_k(d) : k \in \mathcal{K}\}$ from the underlying random field uncertainty $\xi(d)$. With these we can produce $|\mathcal{K}|$ deterministic sets of the DAEs from (4.18):

$$g(D\hat{y}_k, \hat{y}_k(d), \hat{\xi}_k(d), d) = 0,\ k \in \mathcal{K}. \tag{4.26}$$

These denote deterministic DAEs that we can solve using traditional techniques in analogous fashion to the deterministic ordinary differential equations (4.25). One general approach is to model derivative variables $D\hat{y}_k$ as auxiliary variables such that (4.26) can be treated as an algebraic set of equations in conjunction with a set of ODEs that relate the auxiliary derivative variables to the derivatives $D\hat{y}_k$ [30, 135]. Moreover, deterministic

DAEs can be readily simulated using software packages such as `DifferentialEquations.jl` and `InfiniteOpt.jl`. A key caveat to note about this MC approach for solving (4.18) is that $\xi(d)$ must be sample function continuous and differentiable such that we can avoid using stochastic differentials/integrals that require stochastic calculus techniques and cannot be readily collapsed into a deterministic representation. Moreover, each sample DAE set $g(D\hat{y}_k, \hat{y}_k(d), \hat{\xi}_k(d), d) = 0$ is anticipative over the domain $\mathcal{D}$ which may not be desirable for certain domains (e.g., time). Developing more advanced solution methods (that can avoid the above two caveats) for simulating (4.18) is an open-research area, since general RDAE theory is not yet well-developed [132]. Hence, this MC approach is utilized in most of the case studies presented in Section 4.5.

For system models that use RDEs with inhomogeneous noise terms, such as (4.21), we can employ SDE and SPDE numerical techniques to simulate them. Dynamic SDEs are a popular problem class following the form (4.23) with Itô integrals that can be approximated with finite difference methods such as Euler–Maruyama, which adapts the implicit Euler method to suit SDEs with Wiener noise $D\xi(t) = dW(t)$ [63]. For a set of discrete times $\{t_i : 1, \ldots, N\}$ the $k^{\text{th}}$ MC solution is computed as:

$$\hat{y}_k(t_{i+1}) = \hat{y}_k(t_i) + f(\hat{y}_k(t_i), t_i)(t_{i+1} - t_i) + h(t_i)(W(t_{i+1}) - W(t_i)),\ i = 1, \ldots, N-1 \quad (4.27)$$

where we have that $(W(t_{i+1}) - W(t_i)) \sim \mathcal{N}(0, t_{i+1} - t_i)$ and is sampled at each time step. This is the standard approach for simulating SDEs (often called Brownian dynamics simulation) and is non-anticipative (but it is only mildly accurate). High-order (more accurate) methods can also be implemented, and these techniques can be adapted for more general SPDEs. We refer the reader to review [136] for a thorough discussion on the properties and methodologies for implementing these techniques.

### *Other Transformations*

Despite the popularity of MC techniques, a number of other transformation techniques have been developed to simulate stochastic systems. The method of weighted residuals denotes a general class of transformations where the system variables are expressed via basis expansions. In the context of stochastic models, MWR transformations are often referred to as polynomial chaos expansion which emphasizes how the functional basis helps to propagate the randomness (chaos) through the model equations. We approximate the system variables with a linear combination of basis functions $\{\phi_i(d, \xi(d)) : i \in \mathcal{I}\}$:

$$y(d, \xi(d)) \approx \sum_{i \in \mathcal{I}} \tilde{y}_i \phi_i(d, \xi(d)) \tag{4.28}$$

where $\tilde{y}_i \in \mathbb{R}$ are the basis function coefficients. Typically, the basis functions are chosen such that they are orthogonal relative to the finite distribution functions of $\xi(d)$. Figure 4.8 provides a visual depiction of Equation (4.28). Note that this approximation becomes exact as $|\mathcal{I}| \to \infty$ if $y(d, \xi(d))$ resides in the same function space as the basis functions. We can then project the modeling equations onto the basis functions to yield a system of deterministic algebraic equations with variables $\tilde{y}$ that can be solved using classical methods. We refer interested readers to [136, 137] to learn more about performing MWR transformation techniques to simulate stochastic models.

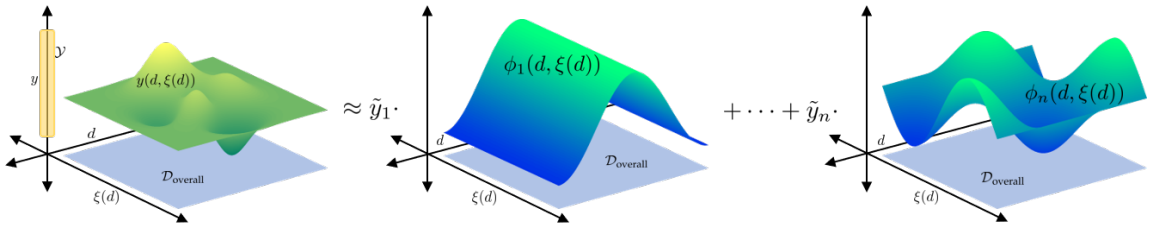



Figure 4.8: A depiction of an MWR approximation for $y(d, \xi(d))$ using a linear combination of orthogonal basis functions $\phi(d, \xi(d))$.

Another group of transformation methods involve representing the stochastic model in terms of its pdf to obtain a deterministic representation that can be solved using clas-

sical approaches. For instance, the Fokker-Planck equation for stochastic particle motion is derived by considering the expectation of an SDE and then applying a number of simplifications to yield a deterministic PDE in terms of the pdf. We refer the reader to [63] to learn more.

## 4.4 Random Field Optimization

In this section, we discuss how random field representations along with the stochastic system models that they produce can be incorporated in a general optimization framework. In particular, we discuss how these stochastic InfiniteOpt problems can be formulated and how these formulations can be solved via finite transformations building on the discussion provided in Section 4.3.

### 4.4.1 *Characterization*

Following Chapter 2, we extend the unifying abstraction for InfiniteOpt problems to directly incorporate random field uncertainty which constitutes our framework for random field optimization. Hence, below we outline this framework in this context and establish notation.

#### *Infinite Domain*

First, we consider the problem infinite domain $\mathcal{D} \subseteq \mathbb{R}^{n_d}$ which denotes the deterministic input domain of the variables (decision functions) of the InfiniteOpt problem. With $\mathcal{D}$, we define the infinite parameter $d \in \mathcal{D}$ which indexes the decision variables. As discussed in Section 4.2, we use $d \in \mathcal{D}$ to construct the manifold support for the random field uncertainty $\xi(d) \in \mathcal{D}_{\xi(d)}$ that our system encounters. With this we can then define the overall infinite domain of our InfiniteOpt problem as:

$$\mathcal{D}_{\text{overall}} := \mathcal{D} \times \mathcal{D}_{\xi(d)}. \tag{4.29}$$

Figure 4.9 provides a highlevel depiction of $\mathcal{D}_{\text{overall}}$.

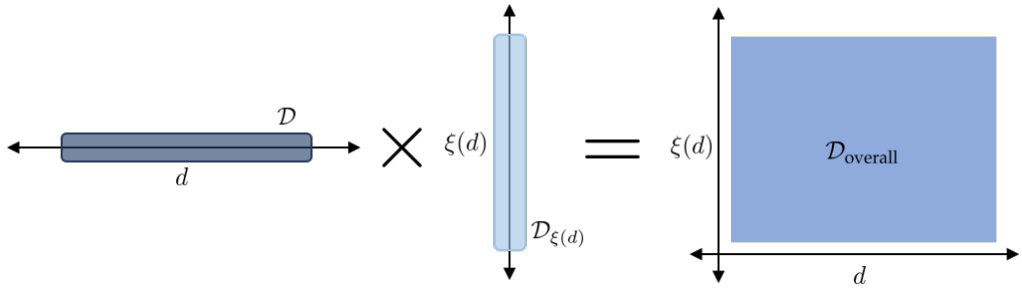



Figure 4.9: An illustration of the overall InfiniteOpt problem domain $\mathcal{D}_{overall}$ defined in Equation (4.29).

***Variables***

The random field (state) variables $y : \mathcal{D}_{\text{overall}} \mapsto \mathcal{Y}$ defined in Equation (4.16) can be interpreted as a generalization of recourse (second-stage) variables (from traditional two-stage stochastic formulations) which are indexed over $\mathcal{D}$. Moreover, we define deterministic decision variables $z : \mathcal{D} \mapsto \mathcal{Z} \subseteq \mathbb{R}^{n_z}$:

$$z(d) \in \mathcal{Z}. \tag{4.30}$$

These typically correspond to parameters and/or deterministic functions in the stochastic models considered in Section 4.3 which constitute the primary decision functions we wish to make. In this sense, the variables $z(d)$ can be interpreted as an infinite-dimensional generalization of the first-stage variables associated with classical two-stage stochastic formulations. These also capture finite variables $z(d) = z$ as a special case.

***Objective***

Objective functions employ a measure operator $M_{d,\xi(d)} : (\mathcal{D}, \mathcal{D}_{\xi(d)}) \mapsto \mathbb{R}$ which summarizes (scalarizes) an infinite-dimensional cost function $f(Dy, y(d, \xi(d)), z(d), \xi(d), d)$:

$$\min Mf(Dy, y(d, \xi(d)), z(d), \xi(d), d). \tag{4.31}$$

For convenience in notation, we will sometimes use $f(d, \xi(d)) := f(Dy, y(d, \xi(d)), z(d), \xi(d), d)$. Figure 4.10 illustrates Equation (4.31). For example, we can use a measure operator $M_{d,\xi(d)}$ that uses the random field expectation $\mathbb{E}_{\xi(d)}$ and Riemann integrals to obtain:

$$\min \mathbb{E}_{\xi(d)} \left[ \int_{d \in \mathcal{D}} f(Dy, y(d, \xi(d)), z(d), \xi(d), d) d(d) \right]. \tag{4.32}$$

For simplicity in presentation, this serves as the workhorse for the numerical studies presented in Section 4.5.

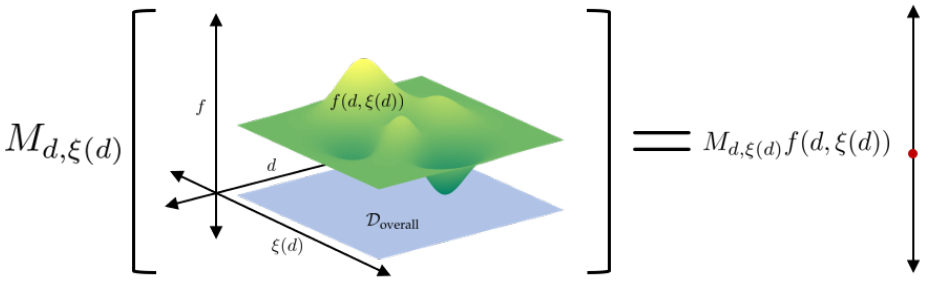



Figure 4.10: A depiction of how an RFO objective cost function $f(d, \xi(d))$ is summarized via the general measure operator $M_{d,\xi(d)}$.

However, we can envision developing other measure operators to shape cost functions $f(d, \xi(d))$. For instance, we can employ the excursion probability in (4.11):

$$\min \mathbb{P}_{\xi(d)} \left( \max_{d \in \mathcal{D}} f(d, \xi(d)) \geq u \right) \tag{4.33}$$

which minimizes the probability that the cost function $f(d, \xi(d))$ exceeds $u \in \mathbb{R}$. Another interesting class of measures would be to extend classical risk measures from stochastic optimization (SO) to operate on random field cost functions $f(d, \xi(d))$. The conditional-value-at-risk (CVaR) measure operator is popular in SO that seeks to measure the tail of the pdf of the cost function:

$$\text{CVaR}_{\xi}(f(\xi); \alpha) := \min_{\hat{f} \in \mathbb{R}} \left\{ \hat{f} + \frac{1}{1-\alpha} \mathbb{E}_{\xi}[\max(f(\xi) - \hat{f}, 0)] \right\} \tag{4.34}$$

where $\xi \in \mathcal{D}_\xi$ is a random variable, $\hat{f} \in \mathbb{R}$ is an auxiliary variable, and $\alpha \in [0,1)$ is the probability level that defines the portion of the tail that is measured. One way to generalize CVaR to random field cost functions would be to nest a measure $M_d$:

$$\text{CVaR}_{\xi(d)}(f(d,\xi(d));\alpha) := \min_{\hat{f}\in\mathbb{R}} \left\{ \hat{f} + \frac{1}{1-\alpha}\mathbb{E}_{\xi(d)}[\max(M_d f(d,\xi(d)) - \hat{f}, 0)] \right\} \tag{4.35}$$

where the measure $M_d f(d,\xi(d))$ can be a traditional deterministic measure such as the integral $\int_{d\in\mathcal{D}} f(d,\xi(d))d(d)$ or the maximization $\max_{d\in\mathcal{D}} f(d,\xi(d))$. We leave a rigorous exploration of such extensions to future work.

***Constraints***

We establish a couple of constraint classes, algebraic and measure constraints, which encode modeling equations (i.e., those presented in Section 4.3) and decision restrictions. Algebraic constraints are established by generalizing the random DAE presented in Equation (4.18):

$$g(Dy, y(d,\xi(d)), z(d), \xi(d), d) \leq 0,\ \xi(d) \in \mathcal{D}_{\xi(d)}, d \in \mathcal{D}. \tag{4.36}$$

This enforces the constraint $g \leq 0$ over each realization of $(\xi(d), d)$. Moreover, this captures point constraints (e.g., initial/boundary conditions) by enforcing this constraint and restricting the overall infinite domain $\mathcal{D}_{\text{overall}}$ to a particular point $(\xi(d^*), d^*)$. We can also capture the more structured R(P)DE equation forms presented in Section 4.3.1 as special cases. We provide a high-level representation of a simple algebraic constraint system in Figure 4.11.

Constraints that we enforce almost surely with respect to the random field uncertainty $\xi(d)$ can be overly burdensome (strict) in certain applications, producing excessively conservative solutions. Measure constraints help to alleviate this limitation by enforcing a constraint on a measured function:

$$Mh(y(d,\xi(d)), z(d), \xi(d), d) \leq 0 \tag{4.37}$$

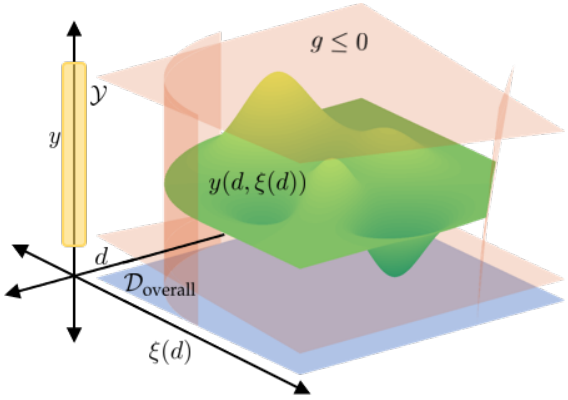



Figure 4.11: A depiction of an algebraic constrained RFO problem following Equation (4.36).

where $M$ is an arbitrary measure operator and $h(\cdot)$ is an $\mathbb{R}^{n_h}$-valued function. These constrain a summarizing statistic over the random field model rather than enforcing a condition on every function sample realization. For instance, we can adapt the excursion probability in Equation (4.11) to constrain the probability of a variable $y(d, \xi(d))$ exceeding a certain threshold $u$ to a probability level $\alpha \in [0, 1]$:

$$\mathbb{P}_{\xi(d)} \left( \max_{d \in \mathcal{D}} y(d, \xi(d)) \geq u \right) \leq \alpha. \tag{4.38}$$

We can interpret this as a generalization of classical chance constraints from SO. Constraint (4.38) can be reformulated:

$$\mathbb{E}_{\xi(d)} \left[ \mathbb{1}(\max_{d \in \mathcal{D}} y(d, \xi(d)) \geq u) \right] - \alpha \leq 0 \tag{4.39}$$

which is conforms to (4.37) by setting $My = \mathbb{E}[\mathbb{1}(\max_d y \geq u)] - \alpha$. We note that when $\alpha \to 0$ we seek to enforce the constraint:

$$\max_{d \in \mathcal{D}} y(d, \xi(d)) < u, \ \xi(d) \in \mathcal{D}_{\xi(d)} \tag{4.40}$$

which is equivalent to:

$$y(d, \xi(d)) < u,\ \xi(d) \in \mathcal{D}_{\xi(d)}, d \in \mathcal{D}. \tag{4.41}$$

Hence, the special case of (4.38) corresponds to an algebraic constraint in accordance with (4.36). Section 4.5.3 implements this proposed constraint class in an illustrative study.

#### *RFO Formulation*

We incorporate all the above modeling objects to define the general random field optimization problem:

$$\begin{aligned} \min \quad & Mf(Dy, y(d, \xi(d)), z(d), \xi(d), d) \\ \text{s.t.} \quad & g(Dy, y(d, \xi(d)), z(d), \xi(d), d) \leq 0, \quad \xi(d) \in \mathcal{D}_{\xi(d)}, d \in \mathcal{D} \\ & Mh(y(d, \xi(d)), z(d), \xi(d), d) \leq 0. \end{aligned} \tag{4.42}$$

This summarizes modeling elements that an RFO problem may entail. Moreover, we highlight that it is a special case of the general InfiniteOpt problem formulation presented in [30].

### 4.4.2 *Transformations*

Random field optimization problems are not readily amenable to classical optimization solution methods due to their infinite-dimensional nature. Hence, we seek to obtain approximate finite representations of RFO problems via transformation routines.

#### *Direct Transcription*

We can use MC sampling approaches to project the RFO problem equations onto a deterministic basis that can be solved with classical methods (e.g., direct transcription). The most straightforward approach is to collect MC sample functions $\hat{\mathcal{D}}_{\xi(d)} := \{\hat{\xi}_k(d) : k \in \mathcal{K}\}$

and define a set of sample points $\hat{\mathcal{D}} := \{\hat{d}_i : i \in \mathcal{I}\}$. Projecting the variables $y(d, \xi(d))$ and $z(d)$ onto these sets provides a collection of finite variables:

$$\begin{aligned} &y(\hat{d}, \hat{\xi}_k(\hat{d})) \in \mathcal{Y}, \quad \xi(\hat{d}) \in \hat{\mathcal{D}}_{\xi(d)}, \hat{d} \in \hat{\mathcal{D}} \\ &z(\hat{d}) \in \mathcal{Z}, \qquad\qquad \hat{d} \in \hat{\mathcal{D}} \end{aligned} \tag{4.43}$$

which we can equivalently write as $\{y_{ik} \in \mathcal{Y} : i \in \mathcal{I}, k \in \mathcal{K}\}$ and $\{z_i \in \mathcal{Z} : i \in \mathcal{I}\}$. This projection onto discrete points is called direct transcription.

We can use these variables to approximate the modeling equations and constraints using the methods discussed in Section 4.3.2. For instance, we can use the Euler–Maruyama method if the equations are RDEs with a non-homogeneous noise term to obtain a finite set of equations. Algebraic equations/constraints can be projected directly onto $\hat{\mathcal{D}}_{\xi(d)}$ and $\hat{\mathcal{D}}$:

$$g(y_{ik}, z_i, \hat{\xi}_{ik}, \hat{d}_i) \leq 0, \ i \in \mathcal{I}, k \in \mathcal{K}. \tag{4.44}$$

Similarly, we can project random DAE constraints directly by introducing auxiliary variables $Dy_{ik}$ for the derivatives $Dy(\hat{d}_i, \hat{\xi}_k(\hat{d}_i))$ which can be approximated using the support points with deterministic methods (e.g., explicit Euler) [30]. However, recall that this approach is anticipative and requires that the random field $\xi(d)$ be sample function differentiable.

Measures can be approximated directly using the sample functions/points. For instance, the objective function in (4.32) would be expressed:

$$|\mathcal{K}|^{-1} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} \beta_i f(Dy_{ik}, y_{ik}, z_i, \hat{\xi}_k(\hat{d}_i), \hat{d}_i) \tag{4.45}$$

where $\beta_i$, $i \in \mathcal{I}$ are constants (e.g., quadrature coefficients) chosen to approximate the integral measure. For transforming excursion probability measures, the analytical approaches described in Section 4.2.4 (e.g., Equation (4.13)) for computing the excursion probability may not always be useful due to the difficulty in propagating the moments

of random fields through an optimization formulation. In some cases Equation (4.14) can be used if the explicit form of $\mathcal{X}(A_u(\cdot)))$ is amenable to the desired optimization tools (e.g., if it is compatible with auto-differentiation to enable gradient optimization solution methods). Otherwise, we can approximate the excursion probability via our MC samples:

$$P_{\xi(d)}\left\{\sup_{d\in\mathcal{D}} f(d,\xi(d)) \geq u\right\} \approx |\mathcal{K}|^{-1} \sum_{k\in\mathcal{K}} \mathbb{1}\left(\sup_{i\in\mathcal{I}} f(\hat{d}_i,\hat{\xi}_k(\hat{d}_i)) \geq u\right). \tag{4.46}$$

Here the indicator function $\mathbb{1}(\cdot)$ can be treated as a nonlinear function, or we can reformulate Equation (4.46) into $vert\mathcal{K}|^{-1}\sum_{k\in\mathcal{K}} q_k$ using big-M constraints:

$$\begin{aligned} &\overline{f}_k \geq f(\hat{d}_i,\hat{\xi}_k(\hat{d}_i)), \quad i\in\mathcal{I}, k\in\mathcal{K} \\ &\overline{f}_k - u \leq q_k M, \qquad\quad k\in\mathcal{K} \end{aligned} \tag{4.47}$$

where $q_k \in \{0,1\}$ are binary variables and $\overline{f}_k \in \mathbb{R}$ are auxiliary variables. Following the relaxation approach proposed in [1], we can relax the binary variables to $q_k \in [0,1]$ (yielding a more tractable continuous formulation) and apply a rounding rule with the optimal values $q_k^*$ which often recovers the binary solution to high accuracy.

We note that transforming an RFO problem via direct transcription can quickly become intractable for even a moderate number of sample functions $\hat{\xi}_k$ and points $\hat{d}_i$ due to the inherent combinatorics involved in constructing the discretization grid. Decomposition approaches can help to alleviate this limitation such as the overlapping Schwarz decomposition method proposed in [138].

### *Other Transformations*

Drawing inspiration from Section 4.3.2, we can envision using more advanced transformation methods to obtain a finite problem formulation that can be used with classical optimization solution techniques. For instance, we can use generalized chaos expansion (i.e., MWR) approaches to project the problem over basis functions $\{\phi_i(d,\xi(d)) : i\in\mathcal{I}\}$ as shown in Equation (4.28). We can then use Galerkin projection to represent Problem

(4.42) in terms of finite coefficient variables to obtain a finite formulation. A similar approach is employed in [97] to project an RFO problem for topological design into a finite formulation. Developing a general MWR transformation framework for RFO problems is beyond the scope of this work, but denotes a promising area of future research that can build upon the general MWR approach for InfiniteOpt problems presented in [30].

As noted in Section 4.3, developing theory and solution methods for general random DAEs is an active area of research. Further research in this area will be invaluable for developing more advanced transformation techniques to apply RFO problems.

## 4.5 Case Studies

In this section, we present illustrative numerical studies that exemplify the principles discussed. These further motivate the advantage of capturing uncertainty over general domains using our proposed framework. All of these are evaluated using MC approaches implemented using `InfiniteOpt.jl v0.5.1` in combination with `GaussianRandomFields.jl v2.1.4` [30]. All the scripts and data are available at https://github.com/zavalab/JuliaBox/tree/master/RandomFieldOptCases.

### 4.5.1 *Dynamic Power Storage Design*

We use a dynamic optimal design study to illustrate a simple RFO problem that involves dynamic uncertainty. In particular, we consider a home that participates in a dynamic electricity pricing market with random field energy pricing $\xi(t) \in \mathbb{R}_+$ where we seek to optimally design the maximum capacity $z_b \in \mathbb{Z}_b \subseteq \mathbb{R}_+$ of a storage battery to fully take advantage of the market such that we minimize our daily electricity cost. Figure 4.12 details the linear network where electrical power $y_g(t, \xi(t)) \in \mathbb{R}_+$ is purchased from the grid with a unit price of $\xi(t)$ to charge a battery with capacity $y_b(t, \xi(t)) \in [0, z_b]$ that powers a home with electricity demand $z_d(t) \in \mathbb{R}_+$.

The random field uncertainty cost function $\xi(t)$ for a 24-hour period is characterized

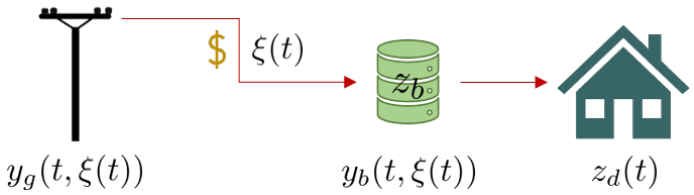



Figure 4.12: A schematic of the power network behind the case study in Section 4.5.1.

as a Gaussian random field with moments:

$$\begin{aligned} \mu(t) &= \sin\left(\frac{2\pi t}{24}\right) + 3 \\ \Sigma_{\text{SE}}(t, t') &= \sigma \exp\left(-\frac{(t-t')^2}{2\beta^2}\right) \end{aligned} \tag{4.48}$$

where we set $\sigma = 0.6$ and $\beta = 1.5$. The field is modeled with `GaussianRandomFields.jl` and Figure 4.13 shows three sample functions $\hat{\xi}_k(t)$ and the mean function $\mu(t)$.

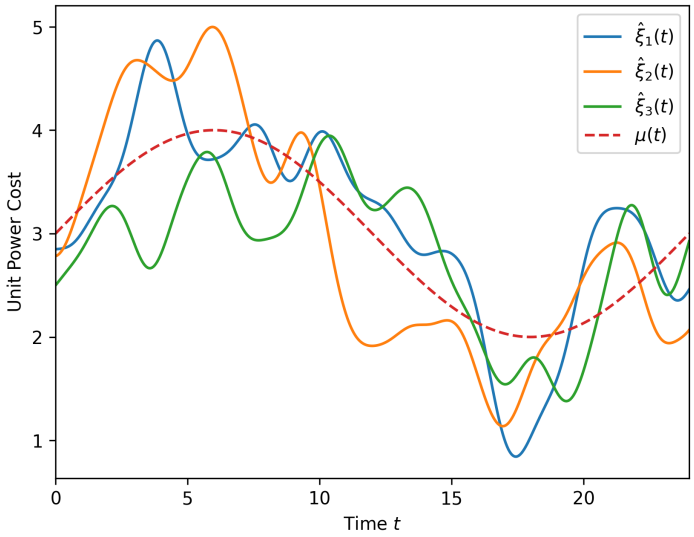



Figure 4.13: A plot of the three sample function realizations $\hat{\xi}_k(t)$ for $\xi(t)$ following the moments given in (4.48).

The expected total cost incurred is computed via operating on the cost function $f(t, \xi(t)) = \xi(t) y_g(t, \xi(t)) + 0.3 z_b$ with $M_{t,\xi(t)}$ from Equation (4.32):

$$\mathbb{E}_{\xi(t)}\left[\int_{t \in \mathcal{D}_t} \xi(t) y_g(t, \xi(t)) dt\right] + 0.3 z_b \tag{4.49}$$

where $\mathcal{D}_t = [0, 24]$. We model the power network by imposing a power balance at the battery node:

$$\frac{\partial y_b(t, \xi(t))}{\partial t} = y_g(t, \xi(t)) - z_d(t),\ \xi(t) \in \mathcal{D}_{\xi(t)}, t \in \mathcal{D}_t. \tag{4.50}$$

For simplicity in example, we set $z_d(t) = 1$. Equation (4.50) is coupled with periodic boundary conditions that exact the battery be at 50% capacity at the beginning and end of a 24-hour period:

$$\begin{aligned} y_b(0, \xi(0)) &= 0.5 z_b, \quad && \xi(t) \in \mathcal{D}_{\xi(t)} \\ y_b(24, \xi(24)) &= 0.5 z_b, && \xi(t) \in \mathcal{D}_{\xi(t)}. \end{aligned} \tag{4.51}$$

We can combine Equations (4.49)-(4.51) to simulate the operation of our model for fixed $y_g(t, \xi(t)$ and $z_b$ to derive an ensemble of battery response trajectories and costs. Such a simulation is trivial in this case since Equation (4.50) does not explicitly depend on $\xi(t)$ and thus can be treated deterministically for each $y_g(t, \hat{\xi}_k(t)$. Figure 4.14 shows the simulated ensemble of the cost function $f(t, \xi(t))$ with

$$y_g(t, \xi(t) = \begin{cases} 0 & t \leq 12 \\ 1 & t > 12 \end{cases} \tag{4.52}$$

that is generated via `InfiniteOpt.jl` using 100 MC samples of $\xi(t)$, 49 equidistant time points, and an implicit Euler evaluation method.

We proceed to characterize the dynamic RFO problem to optimally choose $y_g(t, \xi(t)$ and $z_b$. We enforce that the battery capacity be kept above 20% and below 100% to prevent battery degradation:

$$0.2 z_b \leq y_b(t, \xi(t)) \leq z_b,\ \xi(t) \in \mathcal{D}_{\xi(t)}, t \in \mathcal{D}_t. \tag{4.53}$$

With this last constraint, we can combine (4.49)-(4.53) to yield the dynamic RFO formula-

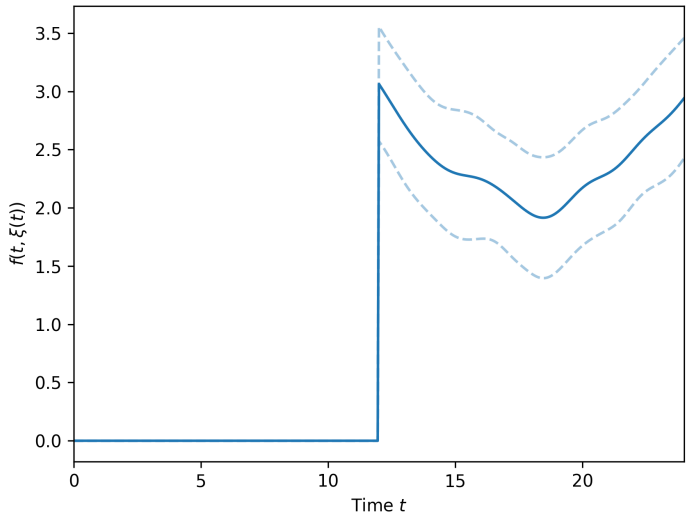



Figure 4.14: A plot the simulated MC ensemble of the operational cost $f(t, \xi(t))$.

tion:

$$
\begin{aligned}
\min_{y_g(\cdot), y_b(\cdot), z_b} \quad & \mathbb{E}_{\xi(t)}\left[\int_{t\in\mathcal{D}_t} \xi(t) y_g(t,\xi(t)) dt\right] + 0.3 z_b \\
\text{s.t.} \quad & \frac{\partial y_b(t,\xi(t))}{\partial t} = y_g(t,\xi(t)) - 1, && \xi(t) \in \mathcal{D}_{\xi(t)}, t \in \mathcal{D}_t \\
& y_b(0,\xi(0)) = 0.5 z_b, && \xi(t) \in \mathcal{D}_{\xi(t)} \\
& y_b(24,\xi(24)) = 0.5 z_b, && \xi(t) \in \mathcal{D}_{\xi(t)} \\
& 0.2 z_b \leq y_b(t,\xi(t)) \leq z_b, && \xi(t) \in \mathcal{D}_{\xi(t)}, t \in \mathcal{D}_t \\
& z_b \in \mathcal{Z}_b
\end{aligned}
\tag{4.54}
$$

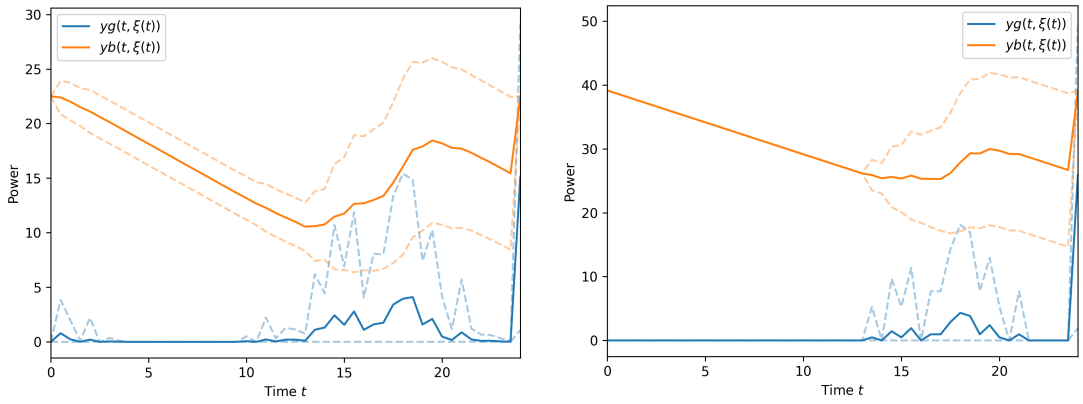
where we let $\mathcal{Z}_b = [0, 100]$. We implement (4.54) in `InfiniteOpt.jl` where we obtain a finite formulation via direct transcription using 49 time points and 100 random field samples $\hat{\xi}_k(t)$ in combination with `Gurobi v.9.1.2`. Moreover, we juxtapose this solution with the one obtained from solving (4.54) with deterministic pricing $\mu(t)$. We simulate the system response when using $z_{b,\text{deterministic}}$ against the same random field samples to compute the expected total cost in the deterministic case.

Table 4.1 shows the results from these studies. The stochastic solution incurs an expected cost that is 10.3% lower than the deterministic counterpart. This difference can

Table 4.1: Optimal solutions for Problem (4.54) and its deterministic variant.

| Formulation | $z_b^*$ | $M_{t,\xi(t)} f^*(t, \xi(t))$ |
|---|---|---|
| Stochastic | 45.0 | 49.5 |
| Deterministic | 78.3 | 55.2 |

be interpreted as an analogue to the value of stochastic solution (VSS) metric that is commonly used in stochastic programming for evaluating the effectiveness of stochastic solutions against their deterministic counterparts [6]. The cost savings can readily be attributed to how the stochastic formulation selects a more conservative maximum battery capacity $z_b$ that is adequate for the system. Figure 4.15 juxtaposes the simulated response of the power network using these choices of $z_b$. We observe how the RFO derived simulation better utilizes purchasing electricity when it is cheaper later in the period; whereas the deterministic derived simulation uses its larger battery capacity to avoid purchasing to a greater extent.




(a) Stochastic (b) Deterministic

Figure 4.15: The ensemble responses for operating the power network over a 24-hour period using the optimal maximum battery capacities $z_b^*$ chosen via the stochastic and deterministic variants of Problem (4.54).

### 4.5.2 *Atomic Layer Deposition Control*

In this case study, we exemplify the application of RFO in problems involving spatial uncertainty. In particular, we examine the atomic layer deposition (ALD) process from

Section 2.5.1. Now, we add the complexity of an uncertain reaction probability $\xi(x) : \mathcal{D}_{\xi(x)} \mapsto [0,1]$ that varies along the depth of the reaction substrate; it is taken to be a Gaussian random field with moments:

$$\begin{aligned} \mu(x) &= 2 \times 10^{-4} \\ \Sigma_{\text{SE}}(x, x') &= 0.0003 \exp\left(-\frac{(x-x')^2}{1.8 \times 10^5}\right) \end{aligned} \tag{4.55}$$

where we take $x$ to be in units of $\mu m$. Figure 4.16 depicts realizations of $\xi(x)$ along with its mean function $\mu(x)$.

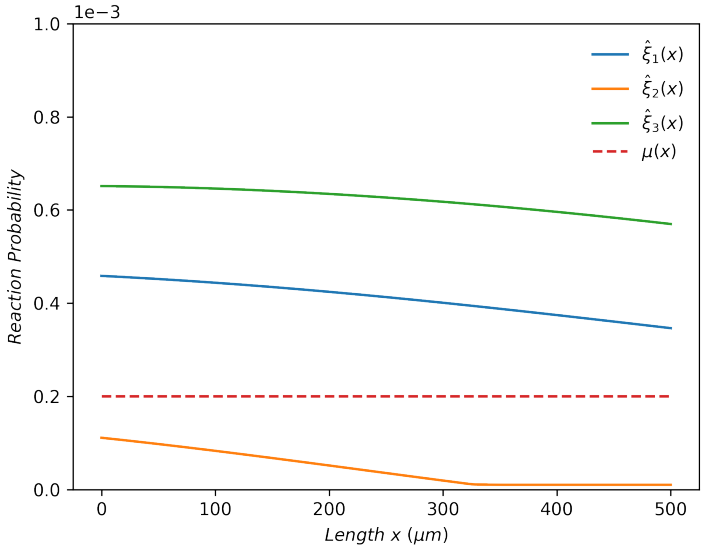



Figure 4.16: Three sample function realizations $\hat{\xi}_k(x)$ for uncertain reaction probability of ALD substrate.

The ALD model considers the precursor density $y_p(t, x, \xi(x)) \in \mathbb{R}_+$ in relation to the fraction of available sites in the substrate surface layer $y_\theta(t, x, \xi(x)) \in [0,1]$. Following the derivation presented in [72], we obtain the RPDE modeling equations:

$$\begin{aligned} \frac{\partial y_p}{\partial t} &= \kappa \frac{\partial^2 y_p}{\partial x^2} - \gamma \xi y_p y_\theta, \quad \xi \in \mathcal{D}_{\xi(x)}, (t,x) \in \mathcal{D}_{t,x} \\ \frac{\partial y_\theta}{\partial t} &= -\eta \xi y_p y_\theta, \qquad\qquad \xi \in \mathcal{D}_{\xi(x)}, (t,x) \in \mathcal{D}_{t,x} \end{aligned} \tag{4.56}$$

where $\mathcal{D}_{t,x} \in \mathcal{D}_t \times \mathcal{D}_x = [0, 10] \times [0, 500]$, $\kappa = 2.81 \times 10^6\ \mu m^2 s^{-1}$ is the diffusivity constant, $\gamma = 6.912 \times 10^8\ s^{-1}$ is the precursor species reaction constant, and $\eta = 1.538 \times 10^7\ \mu m^3 s^{-1}$ is the site reaction constant. For boundary conditions we enforce:

$$\begin{aligned}
&y_p(0, x, \xi(x)) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x \\
&y_p(t, 0, \xi(0)) = z_p, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_{t>0} \\
&\frac{\partial y_p(t, x, \xi(x))}{\partial x}\Big|_{x=500} = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t \\
&y_\theta(0, x, \xi(x)) = 1, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x
\end{aligned} \tag{4.57}$$

which enforce that the substrate not have any precursor initially, the ambient gaseous density be constant at $z_p \in \mathbb{R}$, no diffusion occurs beyond a 500 $\mu m$ depth, and all the sites are initially available, respectively. We simulate Equations (4.56) and (4.57) with $z_p = 0.07$ in `InfiniteOpt.jl` that is evaluated via direct transcription over 10 random field samples, 100 spatial support points, and 10 temporal support points. The resulting finite model is solved using `KNITRO v12.4`. Figure 4.17 shows the final coverage profile $1 - y_\theta(t, x, \xi(x))$ whose behavior is consistent with the results presented in [72].

To setup the RFO problem, we seek to optimally choose $z_p$ such that the coverage profile $1 - y_\theta(10, x, \xi(x))$ follows a desired setpoint $1 - \bar{y}_\theta(x)$. Here the setpoint function is defined via a modified inverse sigmoid function that resembles the desired profile presented in [72].

$$\bar{y}_\theta(x) = 1 - \frac{1}{1 + \exp\left(x - \frac{\rho_1}{2}\right)^{\rho_2}} \tag{4.58}$$

where $\rho_1 = 500$ and $\rho_2 = 0.15$. The objective seeks to minimize the expected setpoint tracking error at the final time:

$$\min \mathbb{E}_{xi(x)} \left[ \int_{x \in \mathcal{D}_x} (y_\theta(10, x, \xi(x)) - \bar{y}_\theta(x))^2 dx \right]. \tag{4.59}$$

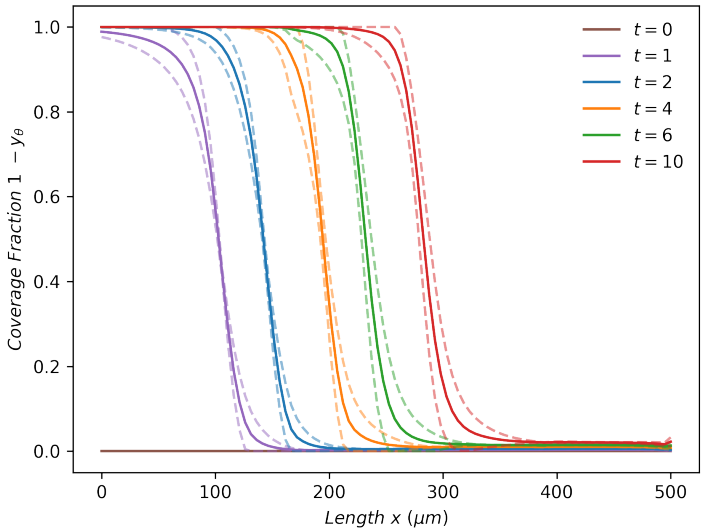



Figure 4.17: A plot the simulated MC ensemble of the coverage profile $1 - y_\theta(t, x, \xi(x))$ at select time instances following Equations (4.56) and (4.57).

Putting together (4.56), (4.57), and (4.59) we obtain the RFO problem:

$$
\begin{aligned}
\min \quad & \mathbb{E}_{xi(x)}\left[\int_{x\in\mathcal{D}_x} (y_\theta(10, x, \xi(x)) - \bar{y}_\theta(x))^2 dx\right] \\
\text{s.t.} \quad & \frac{\partial y_p}{\partial t} = \kappa \frac{\partial^2 y_p}{\partial x^2} - \gamma \xi y_p y_\theta, && \xi \in \mathcal{D}_{\xi(x)}, (t, x) \in \mathcal{D}_{t,x} \\
& \frac{\partial y_\theta}{\partial t} = -\eta \xi y_p y_\theta, && \xi \in \mathcal{D}_{\xi(x)}, (t, x) \in \mathcal{D}_{t,x} \\
& y_p(0, x, \xi(x)) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x \\
& y_p(t, 0, \xi(0)) = z_p, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_{t>0} \\
& \frac{\partial y_p(t, x, \xi(x))}{\partial x}\Big|_{x=500} = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t \\
& y_\theta(0, x, \xi(x)) = 1, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x.
\end{aligned}
\tag{4.60}
$$

This is implemented in `InfiniteOpt.jl` using the same solution configuration used for simulation. We solve the deterministic variant of (4.60) that uses $\mu(x)$ for $\xi(x)$, and we simulate the system response with different realizations of $\xi(x)$ with the deterministic optimal value of $z_p$

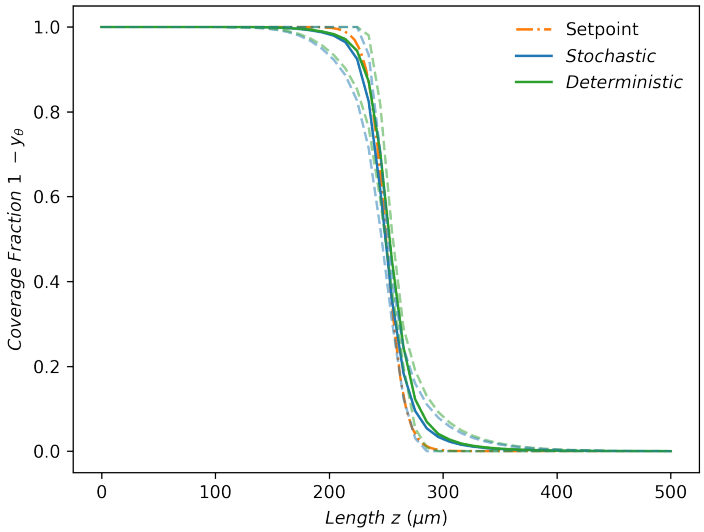



Figure 4.18: The simulated ensemble coverage profiles utilizing the stochastic and deterministic solutions of $z_p$. Both are able to well track the desired setpoint.

Figure 4.18 shows the optimal coverage MC ensemble profiles in juxtaposition to the setpoint profile. In both cases, the system is able to closely track the desired setpoint profile. Moreover, we observe the deterministic and stochastic solutions to be very similar in this case since their choices of $z_p$ are nearly identical (the values are 0.0391 and 0.0393, respectively). Thus, for the behavior of the random field uncertainty $\xi(x)$ in this case, the RFO formulation is not more advantageous than the deterministic one. This highlights that accounting for the uncertainty via an RFO formulation will not always significantly outperform its deterministic variant. However, we can envision other random fields that would incur a significant difference. Thus, in this case study we have demonstrated how our RFO framework readily enables us to characterize uncertainty that propagates over spatial position, simulate a system that is subjected to this uncertainty, and then form an RFO problem to optimally choose conditions at which to run the process.

### 4.5.3 *Optimal Diffusion Excursion Probability*

We further exemplify the principles discussed above by implementing an RFO problem that seeks to minimize the excursion probability of violating a certain concentration/temperature threshold (we refer to it in terms of temperature) in a diffusion system. In particular, we consider a two-dimensional transient diffusion system with temperature $y_c(t, x, \xi(x)) \in \mathbb{R}_+$ subject to spatially random diffusivity $\xi(x) \in \mathcal{D}_{\xi(x)}$ (a 2D random field) that we seek to heat with a spatially variable heating plate with temperature $y_g(t, x) \in [0, 0.1]$. Figure 4.19 provides an illustrative schematic of this system.

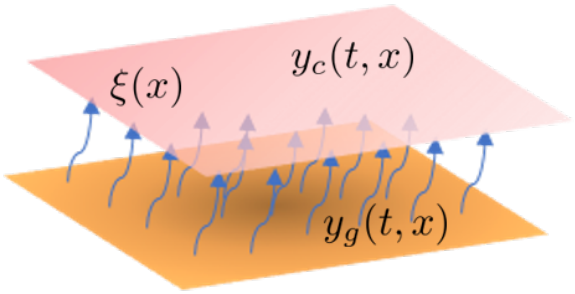



Figure 4.19: A representative schematic of the transient diffusion system considered in Section 4.5.3.

The random field diffusivity is characterized as a Gaussian random field with moments:

$$\begin{aligned} \mu(x) &= 0.5 \\ \Sigma_M(x, x') &= \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu}||x - x'||}{\beta} \right)^{\nu} K_\nu \left( \frac{\sqrt{2\nu}||x - x'||}{\beta} \right) \end{aligned} \tag{4.61}$$

where we set $\beta = 0.25$ and $\nu = 1.5$. We depict four sample function realizations of $\xi(x)$ in Figure 4.20.

Drawing from the transient diffusion relations in (4.24), we model our system:

$$\frac{\partial y_c(t, x, \xi(x))}{\partial t} = \xi(x) \nabla_x^2 y_c(t, x, \xi(x)) + y_g(t, x), \ \xi(x) \in \mathcal{D}_{\xi(x)}, (t, x) \in \mathcal{D}_{t,x} \tag{4.62}$$

where $\mathcal{D}_{t,x} = \mathcal{D}_t \times \mathcal{D}_x = [0, 1] \times [-1, 1]^2$. For boundary conditions, we use Dirichlet

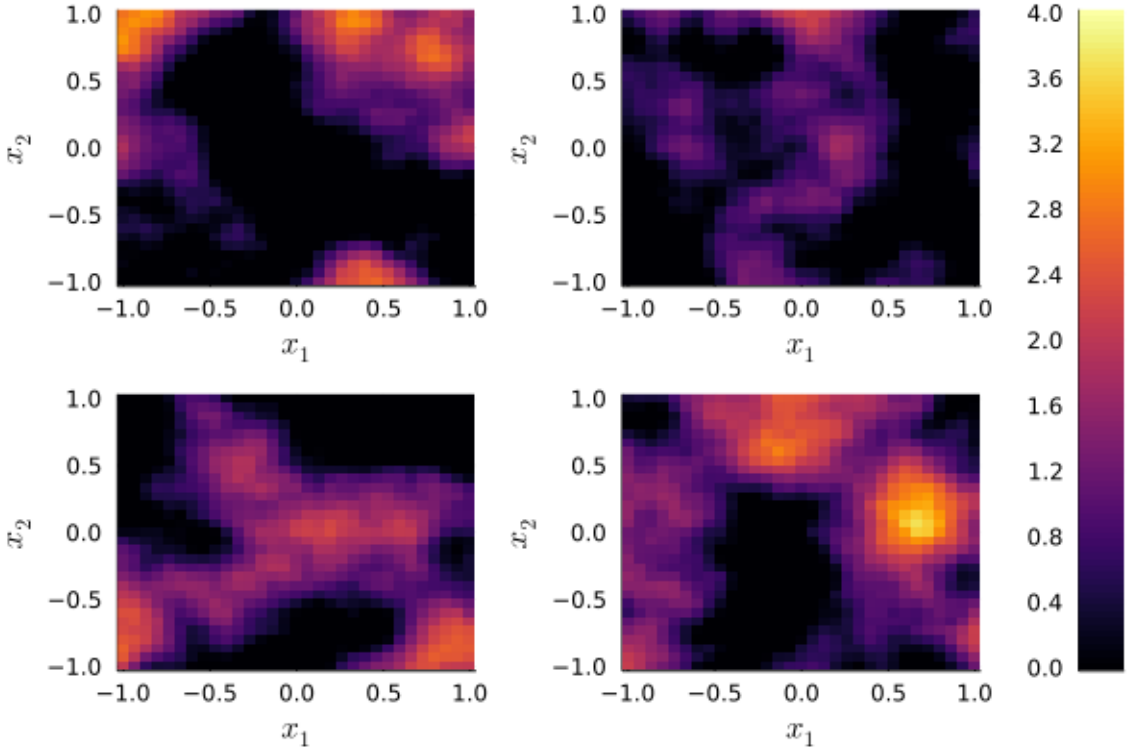



Figure 4.20: Four heatmap realizations of the random diffusivity $\hat{\xi}_k(x)$.

conditions:

$$
\begin{aligned}
&y_c(0, x, \xi(x)) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x \\
&y_c(t, [-1, x_2], \xi([-1, x_2])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t, x_2 \in \mathcal{D}_{x_2} \\
&y_c(t, [1, x_2], \xi([1, x_2])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t, x_2 \in \mathcal{D}_{x_2} \\
&y_c(t, [x_1, -1], \xi([x_1, -1])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t, x_1 \in \mathcal{D}_{x_1} \\
&y_c(t, [x_1, 1], \xi([x_1, 1])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, t \in \mathcal{D}_t, x_1 \in \mathcal{D}_{x_1}.
\end{aligned}
\tag{4.63}
$$

We seek to optimally raise and maintain the plate temperature to a setpoint $y_c(t, x, \xi(x)) = 0.2$ by enforcing the objective:

$$
\min \mathbb{E}_{\xi(x)} \left[ \int_{(t,x) \in \mathcal{D}_{t,x}} (y_c(t, x) - 0.2)^2 dt \right]. \tag{4.64}
$$

Moreover, we limit how the plate temperature exceeds a safety threshold $u = 0.25$ via the

excursion probability constraint:

$$\mathbb{P}_{\xi(x)}\left(\max_{(t,x)\in\mathcal{D}_{t,x}} y_c(t,x,\xi(x)) > 0.25\right) \leq \alpha \tag{4.65}$$

where $\alpha \in [0,1]$ is a probability level. Putting together (4.62) - (4.65), we obtain the RFO problem:

$$\begin{aligned}
\min \quad & \mathbb{E}_{\xi(x)}\left[\int_{(t,x)\in\mathcal{D}_{t,x}} (y_c(t,x)-0.2)^2 dt\right] & \\
\text{s.t.} \quad & \frac{\partial y_c}{\partial t} = \xi\nabla_x^2 y_c + y_g, & \xi\in\mathcal{D}_{\xi(x)}, (t,x)\in\mathcal{D}_{t,x} \\
& y_c(0,x,\xi(x)) = 0, & \xi(x)\in\mathcal{D}_{\xi(x)}, x\in\mathcal{D}_x \\
& y_c(t,[-1,x_2],\xi([-1,x_2])) = 0, & \xi(x)\in\mathcal{D}_{\xi(x)}, (t,x_2)\in\mathcal{D}_{t,x_2} \\
& y_c(t,[1,x_2],\xi([1,x_2])) = 0, & \xi(x)\in\mathcal{D}_{\xi(x)}, (t,x_2)\in\mathcal{D}_{t,x_2} \\
& y_c(t,[x_1,-1],\xi([x_1,-1])) = 0, & \xi(x)\in\mathcal{D}_{\xi(x)}, (t,x_1)\in\mathcal{D}_{t,x_1} \\
& y_c(t,[x_1,1],\xi([x_1,1])) = 0, & \xi(x)\in\mathcal{D}_{\xi(x)}, (t,x_1)\in\mathcal{D}_{t,x_1} \\
& \mathbb{P}_{\xi(x)}\left(\max_{(t,x)\in\mathcal{D}_{t,x}} y_c(t,x,\xi(x)) > 0.25\right) \leq \alpha &
\end{aligned} \tag{4.66}$$

which seeks a tradeoff policy $y_g^*(t,x)$ that optimally minimizes the expected tracking error while not exceeding the excursion threshold. This inherent bi-objective tradeoff behavior is controlled with the choice of $\alpha$. Following the approach in [1], we can invert Problem

(4.66) to obtain the equivalent $\epsilon$-constrained form:

$$\begin{aligned}
\min \quad & \mathbb{P}_{\xi(x)}\left(\max_{(t,x)\in\mathcal{D}_{t,x}} y_c(t,x,\xi(x)) > 0.25\right) \\
\text{s.t.} \quad & \frac{\partial y_c}{\partial t} = \xi\nabla_x^2 y_c + y_g, && \xi \in \mathcal{D}_{\xi(x)}, (t,x) \in \mathcal{D}_{t,x} \\
& y_c(0,x,\xi(x)) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, x \in \mathcal{D}_x \\
& y_c(t,[-1,x_2],\xi([-1,x_2])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, (t,x_2) \in \mathcal{D}_{t,x_2} \\
& y_c(t,[1,x_2],\xi([1,x_2])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, (t,x_2) \in \mathcal{D}_{t,x_2} \\
& y_c(t,[x_1,-1],\xi([x_1,-1])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, (t,x_1) \in \mathcal{D}_{t,x_1} \\
& y_c(t,[x_1,1],\xi([x_1,1])) = 0, && \xi(x) \in \mathcal{D}_{\xi(x)}, (t,x_1) \in \mathcal{D}_{t,x_1} \\
& \mathbb{E}_{\xi(x)}\left[\int_{(t,x)\in\mathcal{D}_{t,x}} (y_c(t,x) - 0.2)^2 dt\right] \leq \epsilon
\end{aligned} \tag{4.67}$$

where $\epsilon \in \mathbb{R}_+$ is the tracking error budget (i.e., the Pareto parameter). We can reformulate the excursion probability following the big-$M$ constraint approach discussed in Section 4.4.2:

$$\mathbb{P}_{\xi(x)}\left(\max_{(t,x)\in\mathcal{D}_{t,x}} y_c(t,x,\xi(x)) > 0.25\right) = \mathbb{E}_{\xi(x)}\left[y_b(\xi(x))\right] \tag{4.68}$$

with constraints:

$$\begin{aligned}
& \overline{y}_c(\xi(x)) \geq y_c(t,x,\xi(x)), && \xi(x) \in \mathcal{D}_{\xi(x)}, (t,x) \in \mathcal{D}_{t,x} \\
& \overline{y}_c(\xi(x)) - 0.25 \leq y_b(\xi(x))M, && \xi(x) \in \mathcal{D}_{\xi(x)}
\end{aligned} \tag{4.69}$$

where $y_b(\xi(x)) \in \{0,1\}$, $\overline{y}_c(\xi(x)) \in \mathbb{R}_+$, and $M \in \mathbb{R}_+$ is a suitable upper bound. This reformulation introduces the binary variable $y_b(\xi(x))$ which compounds the problem complexity, but this can be alleviated via the continuous relaxation proposed in [1] and discussed in Section 4.4.2 where we use the relaxation $y_b(\xi(x)) \in [0,1]$ to obtain Pareto solutions which we can then round to integrality using an appropriate rounding rule (often recovering the optimal mixed-integer solution to high accuracy).

We implement Problem (4.67) with the continuous big-$M$ constraint reformulation of

the excursion probability in `InfiniteOpt.jl` to obtain the Pareto pairs that correspond to:

$$\epsilon \in \{0.0947, 0.095, 0.096, 0.097, 0.098\}. \tag{4.70}$$

We transform the model via direct transcription using 7 realizations of $\xi(x)$ and 9,600 grid points over $(t, x) \in \mathcal{D}_{t,x}$ (10 for $t$ and 31 for $x_1, x_2$), and is solved using `Gurobi v.9.1.2`. The coarseness of the transcription grid shows how the combinatorics behind such transformations quickly increase the problem size. Hence, more sophisticated transformation methods are required for higher fidelity transformations as discussed in Section 4.4.2.

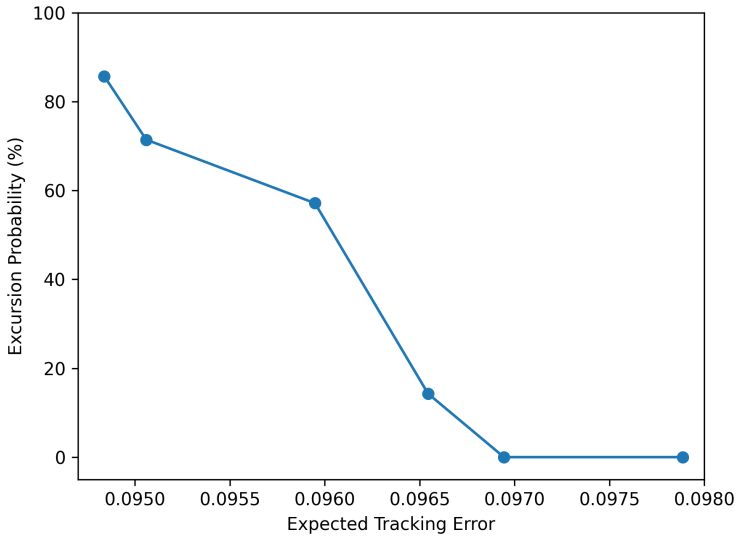
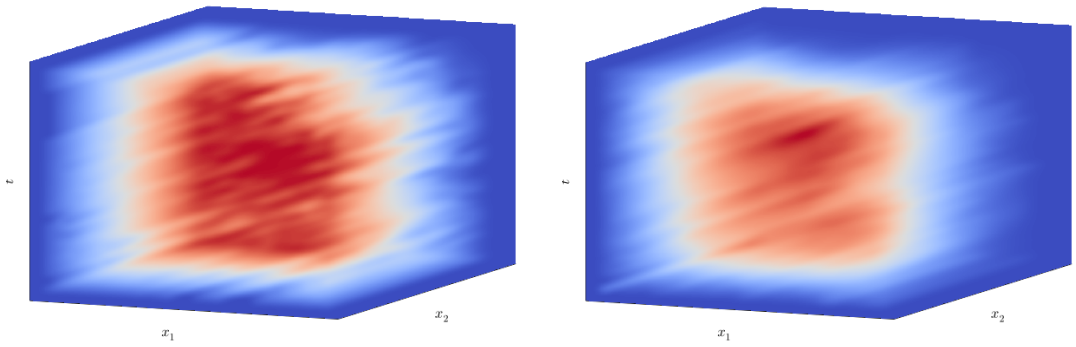



Figure 4.21: The optimal Pareto frontier which interrogates the tradeoff between minimizing the excursion probability in (4.65) with the expected tracking error of (4.64).

Figure 4.21 shows the Pareto frontier obtained from the values of $\epsilon$ in (4.70). This clearly exhibits the tradeoff between tightly tracking the temperature setpoint and minimizing the temperature excursion relative to the sample function realizations of $\xi(x)$ used in this study. Figure 4.22 shows the optimal Pareto pair corresponding to $\epsilon = 0.098$ where we observe the compromised deterministic heating policy $y_g^*(t, x)$ that is used to obtain the optimal response $y_c^*(t, x, \xi(x))$. This highlights how we can use excursion probabili-

ties to shape the policies derived from RFO problems in a manner analogous to what is accomplished with chance constraints in stochastic programming.




(a) $y_g^*(t,x)$ (b) $\mathbb{E}_{\xi(x)}[y_c^*(t,x,\xi(x))]$

Figure 4.22: Heatmap depictions of the optimal heating policy and response of Problem (4.67) that corresponds to $\epsilon = 0.098$.

# Chapter 5

## OTHER ABSTRACTION-DRIVEN INNOVATIONS

The content of this chapter is published in [30].

## 5.1 Introduction

So far, we have demonstrated two ways that our proposed unifying abstraction for InfiniteOpt problems presented in Chapter 2 facilitates innovation. In Chapter 3, we demonstrated how risk measures from stochastic optimization can be generalized to arbitrary infinite parameters and provide useful objects for shaping dynamic trajectories. Chapter 4 presented a new area of optimization called random field optimization that leveraged our abstraction to facilitate the incorporation of random field theory into InfiniteOpt problems, enabling us to propagate uncertainty over general domains.

In this chapter, we further our discussion on other innovative modeling approaches that are enabled by the proposed InfiniteOpt abstraction. Specifically, these innovations are facilitated by the ability to transfer modeling techniques and constructs across disciplines. For instance, in the proposed abstraction, there is no explicit notion of time, space, or random domains (all domains are treated mathematically in the same way); as such, one can easily identify analogues of modeling elements across disciplines.

In particular, we introduce a new class of constraints objects called event constraints

that enforce conditions on measures of functions that encode logical system conditions. We also discuss how abstracting InfiniteOpt problems following our abstraction promotes enhanced analysis techniques; and finally, we establish a new methodology for conducting continuous-time parameter estimation using discrete time-series data.

## 5.2 Event-Constrained Optimization

In this section, we present a new class of constraints that we call event constraints which are an interesting modeling paradigm that arises from the proposed abstraction.

### 5.2.1 *Theory*

These constraints generalize the notion of chance constraints, excursion set conditions, and exceedance probabilities that are used in different scientific disciplines (e.g., stochastic optimization, reliability engineering, and random fields). This unified view also promotes transfer of modeling concepts across disciplines; for instance, we will see that chance constraints in a random domain are analogous to exceedance times in a time domain.

To exemplify the notion of event constraints, consider the so-called *excursion time*; this measure is widely used in reliability analysis of dynamical systems and is defined as the fraction of time that a function $h(t),\ t \in \mathcal{D}_t$, is above a given threshold [139]. Here, we consider a zero threshold value to give the event constraint:

$$\mathbb{P}_t\left(\{\exists t \in \mathcal{D}_t : h(t) > 0\}\right) \leq \alpha. \tag{5.1}$$

where $\alpha \in [0, 1]$. In the context of an InfiniteOpt problem, the function $h(t)$, $t \in \mathcal{D}_t$ can denote a constraint $h(y(t), z, t)$, $t \in \mathcal{D}_t$. The excursion time is expressed as a probability-

like measure of the form:

$$\mathbb{P}_t\left(\{\exists t \in \mathcal{D}_t : h(t) > 0\}\right) = \int_{t\in\mathcal{D}_t} \mathbb{1}\left[\{\exists t \in \mathcal{D}_t : h(t) > 0\}\right] w(t)dt \tag{5.2}$$

where $w : \mathcal{D}_t \to [0, 1]$ is a weighting function satisfying $\int_{t\in\mathcal{D}_t} w(t)dt = 1$. The excursion time measure can be interpreted as the *fraction of time* under which the event of interest occurs; for instance, in safety analysis, one might be interested in determining the fraction of time that a constraint is violated and to ensure that this is not greater than some fraction $\alpha$. Alternatively, we could also search to minimize this measure (by using it as an objective). The excursion time constraint is an event constraint that can help shape a time-dependent trajectory in interesting and non-intuitive ways.

One can generalize the excursion time measure by constructing complex events. For instance, consider that we want to determine the fraction of time that any of the time-varying constraints $h_k(t),\ k \in \mathcal{K}$ crosses a threshold. This can be expressed as:

$$\mathbb{P}_t\left(\bigcup_{jk\in\mathcal{K}} \{\exists t \in \mathcal{D}_t : h_k(t) > 0\}\right). \tag{5.3}$$

Here, $\bigcup$ is a logical `or` operator. If we want to determine fraction of time that all constraints are violated then we use:

$$\mathbb{P}_t\left(\bigcap_{h\in\mathcal{K}} \{\exists t \in \mathcal{D}_t : h_k(t) > 0\}\right). \tag{5.4}$$

Here, $\bigcap$ is a logical `and` operator.

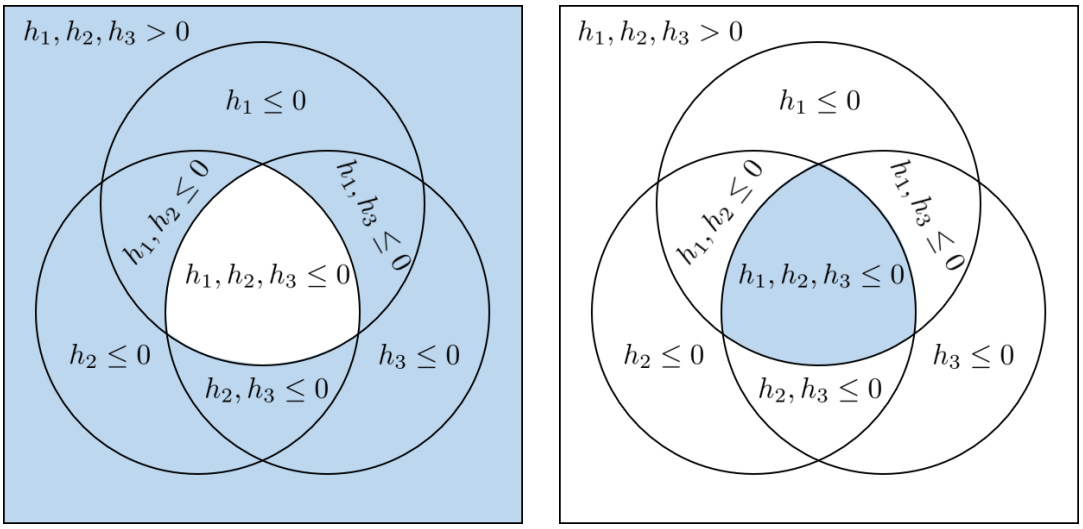
To highlight transfer across different disciplines, we recognize that the excursion time is directly analogous to a chance constraint (operating in the random space, as opposed to time) and our previous analysis suggests that one can construct chance constraints that

capture complex events. For instance, consider the event constraint:

$$\mathbb{P}_{\xi}\left(\bigcup_{k\in\mathcal{K}}\{\exists\xi\in\mathcal{D}_{\xi}:h_k(\xi)>0\}\right)\leq\alpha. \tag{5.5}$$

Here, we see that the constraint is directly analogous to the event constraint (5.3). Figure 5.1a illustrates the logical event space that constraint (5.5) shapes. We thus see that events can be defined in a general form over different infinite domains; to highlight this fact, we consider the event constraint:

$$\mathbb{P}_{d}\left(\bigcup_{k\in\mathcal{K}}\{\exists d\in\mathcal{D}:h_k(d)>0\}\right)\leq\alpha. \tag{5.6}$$




(a) Excursion Constraint (b) Joint-Chance Constraint

Figure 5.1: Logical event regions (shown in blue) constrained by the event constraints (5.5) and (5.7). In particular, they constrain the condition $h_1 > 0 \cup h_2 > 0 \cup h_3 > 0$ and the condition $h_1 \leq 0 \cap h_2 \leq 0 \cap h_3 \leq 0$, respectively.

The previous event constraint is different to traditional joint-chance constraints used in stochastic optimization:

$$\mathbb{P}_{\xi}\left(\bigcap_{k\in\mathcal{K}}\{\exists\xi\in\mathcal{D}_{\xi}:h_k(\xi)\leq 0\}\right)\leq\alpha. \tag{5.7}$$

This makes it more readily apparent that traditional joint-chance constraints consider the logical event space that is complementary to that of constraint (5.5). Figure 5.1b shows this region which is the logical complement of Figure 5.1a.

Another interesting insight that we draw from event constraints is that logical operators (e.g., $\cap$ and $\cup$) can be used to model complex decision-making logic. For example, following the constraint system shown in Figure 5.1; we might consider the logical event region derived from the condition that $h_1 \leq 0 \cap (h_2 \leq 0 \cup h_3 \leq 0)$ giving the event constraint:

$$\mathbb{P}_{\xi}\left(\{\forall \xi \in \mathcal{D}_{\xi} : h_1 \leq 0 \cap (h_2 \leq 0 \cup h_3 \leq 0)\}\right) \geq \alpha. \tag{5.8}$$

This is depicted in Figure 5.2; we note that this event constraint encapsulates a wider probabilistic region relative to that of the traditional joint-chance constraint (5.7).

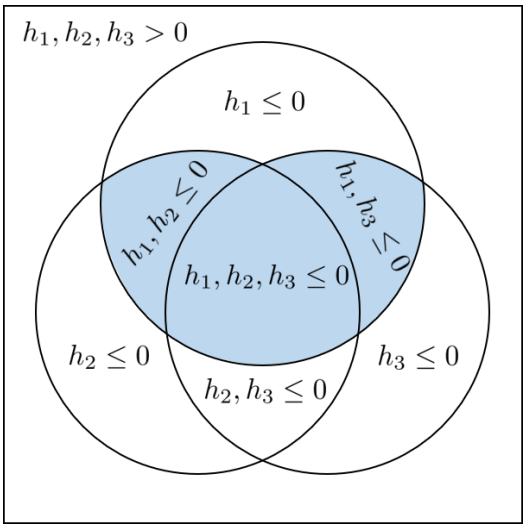



Figure 5.2: Illustration of logical event region captured by constraint (5.8).

In summary, the presented examples illustrate that excursion time constraints and chance constraints as special cases of event constraints. This crossover also led us to representing joint-chance constraints with logical operators which introduce the notion of embedding problem-specific logic to shape the probabilistic region these constraints consider. We illustrate this example further with a stochastic optimal power flow case study below.

### 5.2.2 *Case Study: Power Network Control*

We apply the event constraints featured above to a stochastic optimal power flow (SOPF) formulation (a stochastic optimization problem). We base our SOPF formulation as a variant of the chance-constrained formulation presented in [140]. This considers DC power grid networks subject to random power demands $\xi \in \mathcal{D}_\xi \subseteq \mathbb{R}^{n_\xi}$. The optimal policy defines the power generation $y_g(\xi) \in \mathcal{Y}_g \subseteq \mathbb{R}^{n_g}$ and the branch power flow $y_b(\xi) \in \mathcal{Y}_b \subseteq \mathbb{R}^{n_b}$ recourse functions to satisfy the demands where the respective feasible sets denote the engineering limits (i.e., $\mathcal{Y}_g = [0, \overline{y_g}]$ and $\mathcal{Y}_b = [-\overline{y_b}, \overline{y_b}]$). The power model enforces a linear energy balance at each node of the form:

$$A y_b(\xi) + C_g y_g(\xi) - C_\xi \xi = 0, \ \xi \in \mathcal{D}_\xi \tag{5.9}$$

where $A \in \mathbb{R}^{n_n \times n_b}$ is the incidence matrix, $C_g \in \mathbb{R}^{n_n \times n_g}$ maps the generators to the correct nodes, and $C_\xi \in \mathbb{R}^{n_n \times n_\xi}$ maps the demands to the correct nodes.

A traditional joint-chance constraint enforces limits to a certain probabilistic threshold $\alpha$:

$$\mathbb{P}_\xi \left( \left( \bigcap_{i \in \mathcal{I}_g} \{ \forall \xi \in \mathcal{D}_\xi : y_{g,i}(\xi) \leq \overline{y_{g_i}} \} \right) \bigcap \left( \bigcap_{i \in \mathcal{I}_b} \{ \forall \xi \in \mathcal{D}_\xi : -\overline{y_{b_i}} \leq y_{b,i}(\xi) \leq \overline{y_{b_i}} \} \right) \right) \geq \alpha \tag{5.10}$$

where $\mathcal{I}_g$ is the set of generator indices and $\mathcal{I}_b$ is the set of branch indices. The non-negativity constraints on the generators are excluded such that they are enforced almost surely. The joint-chance constraint thus enforces that all the engineering limits are satisfied to at least a probability $\alpha$ and constraint (5.10) is equivalent to:

$$\mathbb{E}_\xi \left[ \mathbb{1} \left[ \left( \bigcap_{i \in \mathcal{I}_g} y_{g,i}(\xi) \leq \overline{y_{g_i}} \right) \bigcap \left( \bigcap_{i \in \mathcal{I}_b} -\overline{y_{b_i}} \leq y_{b,i}(\xi) \leq \overline{y_{b_i}} \right) \right] \right] \geq \alpha. \tag{5.11}$$

This representation can be reformulated into algebraic constraints by introducing an infi-

nite binary variable $y_w(\xi) \in \{0,1\}$ and an appropriate upper-bounding constant $U \in \mathbb{R}_+$ [57]:

$$\begin{aligned} & y_{g,i}(\xi) - \overline{y_{g_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_g \\ & -y_{b,i}(\xi) - \overline{y_{b_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_b \\ & y_{b,i}(\xi) - \overline{y_{b_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_b \\ & \mathbb{E}_\xi \left[ 1 - y_w(\xi) \right] \geq \alpha. \end{aligned} \tag{5.12}$$

Similarly, we can apply the excursion probability constraint; this enforces the probability that any engineering limit violation be no more than $1 - \alpha$:

$$\mathbb{P}_\xi \left( \left( \bigcup_{i \in \mathcal{I}_g} \{ \exists \xi \in \mathcal{D}_\xi : y_{g,i}(\xi) > \overline{y_{g_i}} \} \right) \bigcup \left( \bigcup_{i \in \mathcal{I}_b} \{ \exists \xi \in \mathcal{D}_\xi : |y_{b,i}(\xi)| > \overline{y_{b_i}} \} \right) \right) \leq 1 - \alpha. \tag{5.13}$$

This constraint is equivalent to a joint-chance constraint; this becomes apparent when we reformulate constraint (5.13) as:

$$\mathbb{E}_\xi \left[ \mathbb{1} \left[ \left( \bigcup_{i \in \mathcal{I}_g} y_{g,i}(\xi) > \overline{y_{g_i}} \right) \bigcup \left( \bigcup_{i \in \mathcal{I}_b} |y_{b,i}(\xi)| > \overline{y_{b_i}} \right) \right] \right] \leq 1 - \alpha. \tag{5.14}$$

and then use $y_w(\xi)$ and $U$ to obtain:

$$\begin{aligned} & y_{g,i}(\xi) - \overline{y_{g_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_g \\ & -y_{b,i}(\xi) - \overline{y_{b_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_b \\ & y_{b,i}(\xi) - \overline{y_{b_i}} \leq y_w(\xi) U, && \xi \in \mathcal{D}_\xi, i \in \mathcal{I}_b \\ & \mathbb{E}_\xi \left[ y_w(\xi) \right] \leq 1 - \alpha. \end{aligned} \tag{5.15}$$

These are equivalent to the set of constraints (5.12) since $\mathbb{E}_\xi \left[ 1 - y_w(\xi) \right] \geq \alpha$ implies $\mathbb{E}_\xi \left[ y_w(\xi) \right] \leq 1 - \alpha$.

As an example of leveraging logical operators (e.g., $\cap$ and $\cup$) to constraint more complex regions, we consider the probability that all the generator limits being satisfied or all

the branch limits being satisfied:

$$\mathbb{E}_{\xi}\left[\mathbb{1}\left[\left(\bigcap_{i\in\mathcal{I}_g} y_{g,i}(\xi)\leq\overline{y_{g_i}}\right)\bigcup\left(\bigcap_{i\in\mathcal{I}_b}-\overline{y_{b_i}}\leq y_{b,i}(\xi)\leq\overline{y_{b_i}}\right)\right]\right]\geq\alpha. \tag{5.16}$$

This encapsulates a wider probabilistic region relative to that of the joint-chance constraint (5.11). This can be reformulated into a system of algebraic constraints by following the same methodology outlined for the other event constraints; however, we need to use multiple infinite binary variables $y_{w,g}(\xi), y_{w,b}(\xi), y_{w,o}(\xi)$:

$$\begin{aligned}
&y_{g,i}(\xi)-\overline{y_{g_i}}\leq y_{w,g}(\xi)U, && \xi\in\mathcal{D}_{\xi}, i\in\mathcal{I}_g\\
&-y_{b,i}(\xi)-\overline{y_{b_i}}\leq y_{w,b}(\xi)U, && \xi\in\mathcal{D}_{\xi}, i\in\mathcal{I}_b\\
&y_{b,i}(\xi)-\overline{y_{b_i}}\leq y_{w,b}(\xi)U, && \xi\in\mathcal{D}_{\xi}, i\in\mathcal{I}_b\\
&y_{w,o}(\xi)\geq y_{w,g}(\xi)+y_{w,b}(\xi)-1, && \xi\in\mathcal{D}_{\xi}\\
&\mathbb{E}_{\xi}\left[1-y_{w,o}(\xi)\right]\geq\alpha.
\end{aligned} \tag{5.17}$$

We can now define the SOPF formulation; here, we aim to compare the use of constraint (5.11) with the use of constraint (5.16):

$$\begin{aligned}
\min_{y_g(\xi),y_b(\xi)}\quad & \mathbb{E}_{\xi}\left[c_g^T y_g(\xi)\right]\\
\text{s.t.}\quad & Ay_b(\xi)+C_g y_g(\xi)-C_{\xi}\xi=0, && \xi\in\mathcal{D}_{\xi}\\
& y_g(\xi)\geq 0, && \xi\in\mathcal{D}_{\xi}\\
& (5.11)\text{ or }(5.16)
\end{aligned} \tag{5.18}$$

where $c_g\in\mathbb{R}^{n_g}$ are generation unit costs. We apply this SOPF formulation to the 4-bus power system depicted in Figure 5.3. We set $\overline{y_{g_i}}=10$, $\overline{y_{b_i}}=4$, $U=100$, $c_g^T=[1\ 10]$, and $\xi\sim\mathcal{N}(\mu,\Sigma)$ where

$$\mu=\begin{bmatrix}3\\5\end{bmatrix},\quad \Sigma=\begin{bmatrix}2&0\\0&2\end{bmatrix}. \tag{5.19}$$

The matrices $A$, $C_g$, and $C_\xi$ are determined by the topology shown in Figure 5.3.

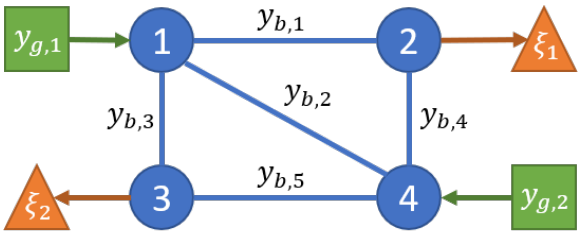



Figure 5.3: Sketch of the 4-bus power network topology with its bus nodes (blue circles), branches (blue lines), generators (green squares), and demand loads (orange triangles).

**Code Snippet 5.1: Formulation (5.18) with constraints (5.12) via `InfiniteOpt.jl` to obtain Pareto solutions.**

```julia
using InfiniteOpt, Distributions, Gurobi

# Initialize the model
model = InfiniteModel(Gurobi.Optimizer)

# Define the parameters
@finite_parameter(model, α == 0.95)
@infinite_parameter(model, ξ[1:2] ~ MvNormal(μ, Σ), num_supports = 1000)

# Define the variables
@variables(model, begin
    0 <= yg[1:2], Infinite(ξ)
    yb[1:5], Infinite(ξ)
    yw, Infinite(ξ), Bin
end)

# Set the objective
@objective(model, Min, 𝔼(cg' * yg, ξ))

# Add the constraints
@constraint(model, A * yb .+ Cg * yg .- Cξ * ξ .== 0)
@constraint(model, yg - yg_lim .<= yw * U)
@constraint(model, - yb - yb_lim .<= yw * U)
@constraint(model, yb - yb_lim .<= yw * U)
@constraint(model, chance, 𝔼(1 - yw, ξ) >= α)

# Solve for the Pareto solutions
objs = zeros(length(αs))
probs = zeros(length(αs))
for (i, prob) in enumerate(αs)
    set_value(α, prob)
    optimize!(model)
    objs[i] = objective_value(model)
    probs[i] = value(chance)
end
```

We implement both variants of (5.18) in `InfiniteOpt.jl` and use direct transcription with 1,000 MC samples of $\xi$ to transform the InfiniteOpt problem into finite-dimensional form. Code Snippet 5.1 shows an excerpt of the script used in `InfiniteOpt.jl`. Note that the algebraic reformulations of each event constraint are used (e.g., constraints (5.12) in

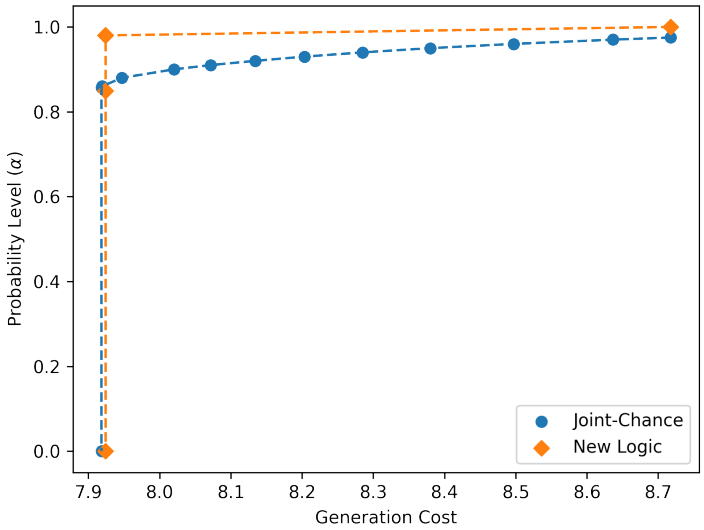



Figure 5.4: Pareto frontiers associated formulation (5.18) where the joint-chance curve refers to using constraint (5.11) and the new logic curve corresponds to constraint (5.16).

place of constraint (5.11)). Each formulation is solved over a range of $\alpha$ values to obtain Pareto pairs, which are shown in Figure 5.4. We observe that the traditional joint-chance constraint requires a higher power generation cost for a given probability $\alpha$. This makes sense, because the alternate formulation captures a larger probabilistic event region (it is less constraining). This highlights how logic affects the behavior of event-constrained optimization formulations. In summary, in this case study we have used our unifying abstraction to explore the use of event constraints. This is facilitated by studying the analogy between excursion probability constraints and joint-chance constraints through the lens our abstraction.

## 5.3 Problem Analysis

Here we highlight some of the advantages that arise from characterizing InfiniteOpt problems directly in accordance with formulation (2.24), in contrast to the standard practice of expressing them solely via finite reformulations. For instance, within the area of dynamic

optimization, it is commonplace to abstract and formalize problem classes in discrete time (i.e., in a transcribed form) [89]. This practice tends to make problems more difficult to formulate since they are inherently coupled with the transformation scheme employed (e.g., orthogonal collocation or explicit Euler). Hence, decoupling the problem definition from its transformation helps to ease its formalization. Arguably, this decoupling better defines a particular problem and promotes the use of diverse formulations and transformation techniques. For instance, by operating at a different level of abstraction, one might more easily identify alternative modeling and solution techniques: different measures, non-traditional support schemes, alternative derivative approximations, and/or advanced problem transformations. For instance, analyzing the problem in its infinite-dimensional form is what inspired the generalized risk-measures discussed in Chapter 3; this example shows how to use CVaR as a way to manipulate time-dependent trajectories.

Establishing theoretical properties for InfiniteOpt formulations is also often facilitated when these problems are expressed in their native form. This is exemplified in the recent work of Faulwasser and Grüne [29], where the authors utilize continuous and discrete time formulations to derive properties of the turnpike phenomenon in the field of optimal control. The turnpike phenomenon refers to the propensity of optimal control trajectories to remain within a certain region for a significant portion of the time horizon until later departing it. Figure 5.5 illustrates this for a dynamic variable $y(t)$. The properties discussed by the authors with regard to turnpike behavior are beyond the scope of this work but, interestingly, they observe that a considerable amount of analysis has been done for finite time formulations whereas many conceptual gaps remain for the continuous-time case. This conceptual disparity between the continuous and discrete time cases can at least in part be attributed to the rather standard practice of expressing optimal control formulations in discrete time. This observation is not unique to the optimal control community and there exists much to be explored for InfiniteOpt formulations in their native forms throughout their respective communities in general. Some potential avenues of research for InfiniteOpt formulations include systematic initialization techniques that consider the

formulation infinite domain, generalized pre-solving methods (e.g., feasibility checking), and enhanced transformations (e.g., basis function approaches used in [20] and [62]).

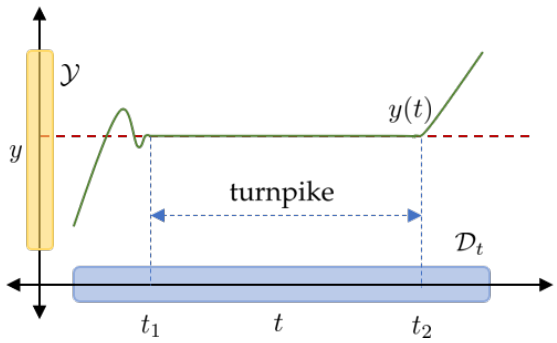



Figure 5.5: Illustration of the turnpike phenomenon for a time-dependent trajectory $y(t)$ where the turnpike occurs on the interval $[t_1, t_2]$.

## 5.4 Parameter Estimation for Dynamical Systems

The proposed abstraction also encourages a more systematic treatment of infinite-dimensional data and modeling objects. To illustrate this, we consider dynamic parameter estimation formulations and show how lifting a finite-dimensional data representation into an infinite-dimensional form might be beneficial.

### 5.4.1 *Theory*

We consider conducting dynamic experiments $k \in \mathcal{K}$ that collect the set of observations (data) $\{\tilde{y}_k(t) : t \in \hat{\mathcal{D}}_{t_k},\ k \in \mathcal{K}\}$ over the set of time points $t \in \hat{\mathcal{D}}_{t_k}$. A dynamic parameter estimation formulation then seeks the optimal choice of parameters $z \in \mathcal{Z}$ to fit and verify the efficacy of a candidate model $g(y(t), z, t) = 0$ relative to empirical data [14, 141]. For simplicity in example, we consider a least-squares approach that yields the following

canonical discrete-time estimation formulation:

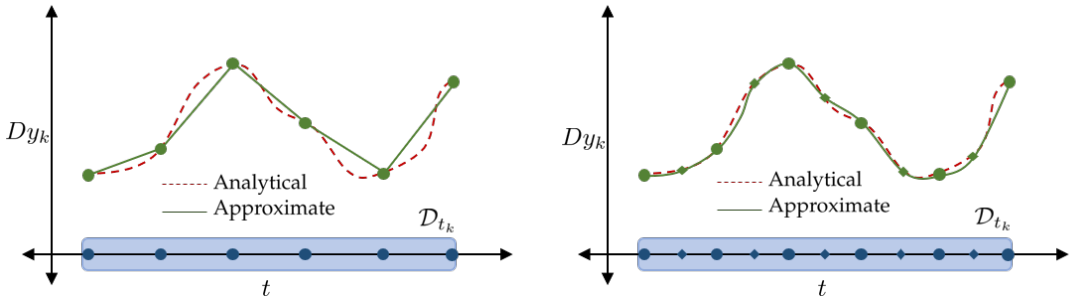
$$\begin{aligned} \min_{y_k(\cdot),z} \quad & \sum_{k\in\mathcal{K}} \sum_{t\in\hat{\mathcal{D}}_{t_k}} (y_k(t) - \tilde{y}_k(t))^2 \\ \text{s.t.} \quad & \hat{g}(y(t), z, t) = 0, \quad t \in \hat{\mathcal{D}}_{t_k}, k \in \mathcal{K} \\ & z \in \mathcal{Z} \end{aligned} \tag{5.20}$$

where $\hat{g}(\cdot)$ is the discretized dynamic model and $y_k(\cdot)$ are the model predicted variables. This discrete representation is guided by the nature of the experimental data $\tilde{y}_k(t)$ available, which corresponds to a finite set of time points $t \in \hat{\mathcal{D}}_{t_k}$. This limits the model to a particular transcribed domain; however, we can express formulation (5.20) in continuous time by representing the experimental data with appropriate infinite-dimensional lifting functions $f_k(t), t \in \mathcal{D}_{t_k}$ such that $f_k(t) = \tilde{y}(t)$ at $t \in \hat{\mathcal{D}}_{t_k}$:

$$\begin{aligned} \min_{y_k(\cdot),z} \quad & \sum_{k\in\mathcal{K}} \left( \int_{t\in\mathcal{D}_{t_k}} (y_k(t) - \tilde{y}_k(t))^2 dt \right) \\ \text{s.t.} \quad & g(y(t), z, t) = 0, && t \in \mathcal{D}_{t_k}, k \in \mathcal{K} \\ & \tilde{y}_k(t) = f_k(t), && t \in \mathcal{D}_{t_k}, k \in \mathcal{K} \\ & z \in \mathcal{Z}. \end{aligned} \tag{5.21}$$

We now have a formulation that fits into our unifying InfiniteOpt abstraction; as such, we can begin to consider general modeling elements and transformations. This means, for instance, that we might want to consider alternative time-dependent measures or more accurate derivative approximations (e.g., orthogonal collocation over finite elements) [142]. Figure 5.6 demonstrates this principle for a certain derivative $D\,y_k$. This approach also has the potential to alleviate the large computational burden associated with the noisy experimental data that often plague dynamic estimation [143], since the chosen empirical data functions $f_k(\cdot)$ smooth the empirical domains as a preprocessing step (see Figure 5.7 in the case study). The data functions also facilitate the computation of data derivatives; which can be used in estimation techniques such as `SINDy`. We leave a more rigorous

analysis of formulation (5.21) to future work, but we hope that this discussion helps illustrate how lifting can help identify new perspectives to tackle problems that are typically treated as finite-dimensional. We study this approach further in the biological dynamic parameter estimation case study presented below.




(a) Traditional (b) New

Figure 5.6: Comparison between the derivative approximation approaches common to traditional dynamic estimation formulations and higher-order ones possible using our new formulation (e.g., using orthogonal collocation).

### 5.4.2 *Case Study: Microbial Communities*

We consider a dynamic parameter estimation problem for a biological system model; this study aims to demonstrate how the unifying abstraction inspires new formulation classes by lifting formulations into infinite dimensional spaces. This follows from the discussion above with regard to formulations (5.20) and (5.21).

To juxtapose the utility of formulations (5.20) and (5.21), we consider a microbial community consisting of the 12 species described in Table 5.1 using the generalized Lotka-Volterra (gLV) model:

$$\frac{dy_{x,i}(t)}{dt} = \left(z_{\mu,i} + \sum_{j \in \mathcal{I}} z_{\alpha,ij} y_{x,j}(t)\right) y_{x,i}(t), \quad i \in \mathcal{I} \tag{5.22}$$

where $\mathcal{I}$ represents the set of microbial species, $y_{x,i}(t)$ is the estimated absolute abundance of species $i \in \mathcal{I}$, $z_{\mu,i}$ is the growth rate of species $i$, and $z_{\alpha,ij}$ is a species interaction coefficient which describes the effect of the recipient species $i$ on the growth of the donor

Table 5.1: Species membership of the microbial community.

| Species Name | Abbreviation |
|---|---|
| Blautia hydrogenotrophica | BH |
| Collinsella aerofaciens | CA |
| Bacteroides uniformis | BU |
| Prevotella copri | PC |
| Bacteroides ovatus | BO |
| Bacteroides vulgatus | BV |
| Bacteroides thetaiotaomicron | BT |
| Eggerthella lenta | EL |
| Faecalibacterium prausnitzii | FP |
| Clostridium hiranonis | CH |
| Desulfovibrio piger | DP |
| Eubacterium rectale | ER |

species $j$ [14]. We use the gLV model parameters presented in [141] to generate simulated experimental data with random noise $\epsilon \sim \mathcal{N}(0, 0.01)$ for 12 mono-species and 66 pairwise experiments. This will enhance our assessment of Formulations (5.20) and (5.21), since we have an established ground truth for the model parameters.

Incorporating (5.22) into Formulation (5.20) provides us with our discrete dynamic biological community estimation formulation:

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{I}} \sum_{t\in\hat{\mathcal{D}}_{t_k}} (y_{x,ik}(t) - \tilde{y}_{x,ik}(t))^2 \\
\text{s.t.} \quad & \frac{dy_{x,ik}(t)}{dt} = \left(z_{\mu,i} + \sum_{j\in\mathcal{I}} z_{\alpha,ij} y_{x,jk}(t)\right) y_{x,ik}(t), \quad t \in \hat{\mathcal{D}}_{t_k}, i \in \mathcal{I}, k \in \mathcal{K} \\
& 0.09 \leq z_{\mu,i} \leq 2.1, \qquad\qquad i \in \mathcal{I} \\
& -10 \leq z_{\alpha,ii} \leq 0, \qquad\qquad i \in \mathcal{I} \\
& -10 \leq z_{\alpha,ij} \leq 10, \qquad\qquad (i, j \neq i) \in \mathcal{I} \times \mathcal{I}.
\end{aligned}
\tag{5.23}
$$

Note that we sum over each species $i$ for each experiment $k$ in accordance with (5.22). Also, we recall that the derivative terms are limited to finite difference approximation schemes that employ the support points in $\hat{\mathcal{D}}_{t_k}$.

To derive an explicit form of Formulation (5.21), we first fit an empirical function

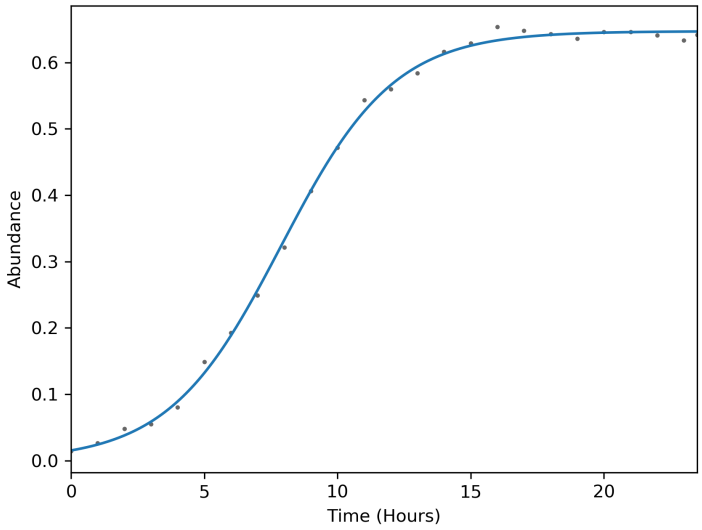



Figure 5.7: Empirical fit for a mono-species experiment of Bacteroides vulgatus using (5.24).

$f_{ik}(t_k)$ to each experiment $k$ and each species $i$. Our goal is not to create a generalizable predictive representation but rather to construct a continuous, infinite-dimensional function that represents the dynamic behavior observed in each experiment. We observe that the experiments appear to exhibit sigmoidal characteristics, and thus we use least-squares to fit each experiment $k$ and species $i$ to an empirical function of the form:

$$f_{ik}(t) := \frac{\beta_{1,ik}}{\beta_{2,ik} + \beta_{3,ik} e^{\beta_{4,ik}(t-\beta_{5,ik})}},\ t \in \mathcal{D}_{t_k}, i \in \mathcal{I}, k \in \mathcal{K} \tag{5.24}$$

where $\beta_{1,ik}$, $\beta_{2,ik}$, $\beta_{3,ik}$, $\beta_{4,ik}$, and $\beta_{5,ik}$ are fitting parameters. Equation (5.24) fits our experimental datasets well as demonstrated for the particular experiment shown in Figure 5.7 which is indicative of overall fit qualities we observed. Thus, we substitute Equations

(5.22) and (5.24) into Formulation (5.21) to obtain:

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}}\left(\int_{t\in\mathcal{D}_{t_k}}(y_{x,ik}(t)-\tilde{y}_{x,ik}(t))^2dt\right) \\
\text{s.t.} \quad & \frac{dy_{x,ik}(t)}{dt} = \left(z_{\mu,i}+\sum_{j\in\mathcal{I}}z_{\alpha,ij}y_{x,jk}(t)\right)y_{x,ik}(t), && t\in\mathcal{D}_{t_k}, i\in\mathcal{I}, k\in\mathcal{K} \\
& \tilde{y}_{x,ik}(t) = \frac{\beta_{1,ik}}{\beta_{2,ik}+\beta_{3,ik}e^{\beta_{4,ik}(t-\beta_{5,ik})}}, && t\in\mathcal{D}_{t_k}, i\in\mathcal{I}, k\in\mathcal{K} \\
& 0.09 \le z_{\mu,i} \le 2.1, && i\in\mathcal{I} \\
& -10 \le z_{\alpha,ii} \le 0, && i\in\mathcal{I} \\
& -10 \le z_{\alpha,ij} \le 10, && (i,j\neq i)\in\mathcal{I}\times\mathcal{I}.
\end{aligned}
\tag{5.25}
$$

**Code Snippet 5.2: Formulation (5.25) using `InfiniteOpt.jl`.**

```
using InfiniteOpt, Ipopt

# Initialize the model
model = InfiniteModel(Ipopt.Optimizer)

# Set the infinite parameters
@infinite_parameter(model, t[k ∈ K] ∈ [0, T[k]], num_supports = 15, independent = true,
                    derivative_method = OrthogonalCollocation(4))

# Set the finite variables
@variable(model, 0.09 ≤ zμ[i ∈ I] ≤ 2.1)
@variable(model, -10 ≤ zα[i ∈ I, j ∈ I] ≤ zα_max[i, j])

# Set the infinite variables
@variable(model, yx[i ∈ I, k ∈ K] ≥ 0, Infinite(t[k]))

# Set the empirical functions using fitted q[i, k](t) functions
@parameter_function(model, yx_tilde[i ∈ I, k ∈ K] == q[i, k](t[k]))

# Set the least−squares objective
@objective(model, Min, sum(∫((yx[i, k] - yx_tilde[i, k]) ^ 2, t[k]) for i ∈ I, k ∈ K))

# Define the gLV equations
@constraint(model, [i ∈ I, k ∈ K],
            ∂(yx[i, k], t[k]) == (zμ[i] + sum(zα[i, j] * yx[j, k] for j ∈ I)) * yx[i, k])

# Optimize and get the results
optimize!(model)
zα_opt = value.(zα)
zμ_opt = value.(zμ)
yx_opt = value.(yx)
ts = supports(t)
```

We solve formulations (5.23) and (5.25) using `InfiniteOpt.jl` via its automated transcription capabilities. Code Snippet 5.2 presents an illustrative summary of the syntax for implementing Formulation (5.25). We employ 15 time supports for each experiment

in Formulation (5.25) and necessarily use the measurement times as supports for formulation (5.23). We use orthogonal collocation over finite elements to approximate the differential equations; we use two nodes per finite element (necessarily using only the boundary points) in Formulation (5.23) and a range of node amounts with formulation (5.25) to investigate their effect on solution quality. Figure 5.8 summarizes the model fits of both solution sets relative to the experimental data. Although both estimation problems are able to choose parameters that characterize the data well, there are significant deviations between the profiles across a few experiments.

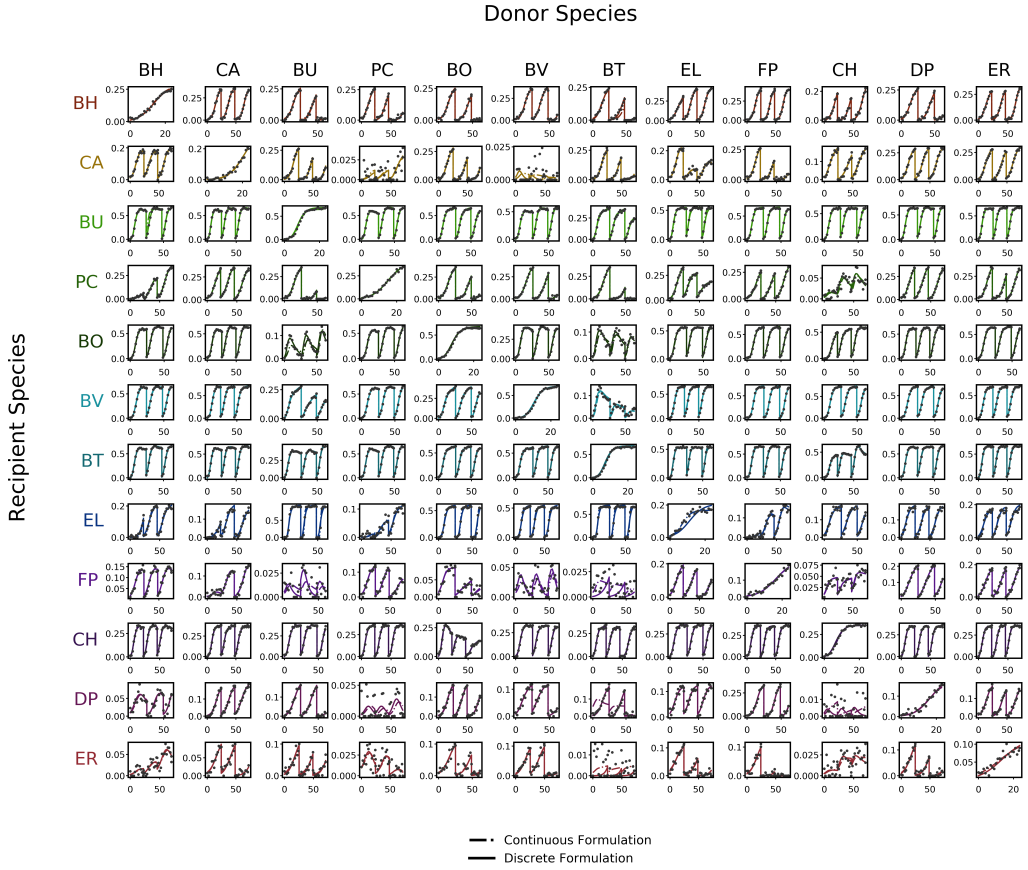



Figure 5.8: Optimal trajectories from formulations (5.23) and (5.25) using orthogonal collocation over finite elements to approximate the derivatives with two and four points, respectively. All shown in comparison to the experimental data. The x-axis is time in hours, and the y-axis is the absolute abundance of the recipient species in contact with the corresponding donor species. The results for the mono-species experiments are observed on the diagonal with the rest being pairwise.

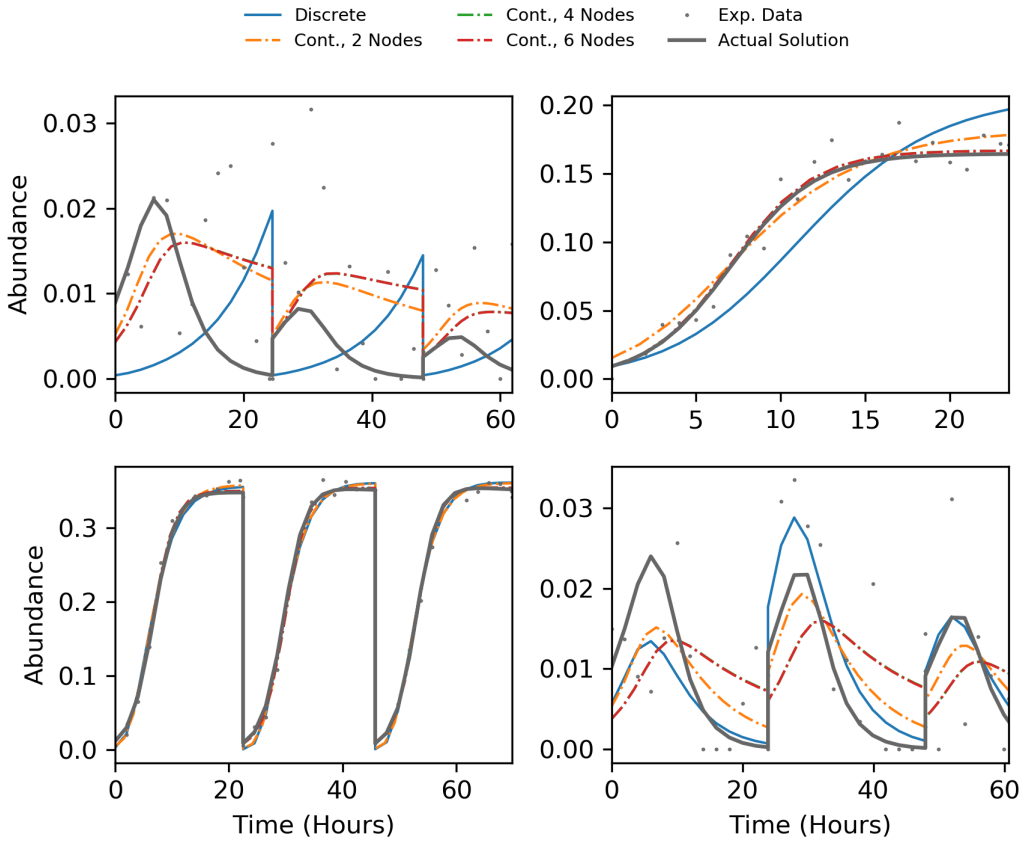



Figure 5.9: Optimal profiles for select experiments using different formulations and collocation node amounts (top-left: (FP, BT), top-right: (EL, EL), bottom-left: (CH, PC), bottom-right: (FP, BU)).

Figure 5.9 shows a representative subset of experiments to demonstrate how the solutions vary with respect to the estimation formulation and the number of collocation points. We find that the discrete formulation solution generally deviates from the analytical solution to a greater extent than the solutions procured via the continuous formulation. However, it is difficult to conclude which estimation problem best represents the data when the measurement noise is near the magnitude of the absolute abundance because no model matches the true analytical solution particularly well; the continuous formulation solutions, however, in general seem to better represent the trend of the system. This suggests that Formulation (5.25) has effectively smoothed over noisy experiments, leading to a better fit. Furthermore, we observe that the ability of Formulation

(5.25) to enable arbitrary collocation nodes has a significant effect on the accuracy of the solutions. This increased accuracy seems to effectively taper off at four collocation nodes in this case.

We seek to substantiate our qualitative observation that the continuous formulation is able to better represent the experimental data by comparing the sum-squared-errors (SSE) between the true parameters used to generate the experimental data to those approximated from formulations (5.23) and (5.25). Specifically, we consider the error metrics:

$$SSE_{z_\mu} := \sum_{i \in \mathcal{I}} (z_{\mu,i} - \bar{z}_{\mu.i})^2$$

$$SSE_{z_\alpha} := \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{I}} (z_{\alpha,ij} - \bar{z}_{\alpha,ij})^2$$

where $\bar{z}_{\mu.i}$ and $\bar{z}_{\alpha,ij}$ denote the actual parameters used in the simulations to generate the experimental data. The results are shown in Table 5.2 and demonstrate that the continuous formulation solutions yield significantly smaller sum-squared-errors. Moreover, increased collocation nodes for each finite element are able to reduce the overall error by more than an order of magnitude.

Table 5.2: Sum-squared-errors between the actual and estimated parameters for each formulation and number of collocation nodes per finite element.

| Formulation | $SSE_{z_\alpha}$ | $SSE_{z_\mu}$ |
|---|---|---|
| Discrete, 2 Nodes | $1.02 \times 10^{2}$ | $4.72 \times 10^{-2}$ |
| Continuous, 2 Nodes | $2.14 \times 10^{1}$ | $2.46 \times 10^{-2}$ |
| Continuous, 4 Nodes | $1.59 \times 10^{1}$ | $9.05 \times 10^{-4}$ |
| Continuous, 6 Nodes | $1.59 \times 10^{1}$ | $9.10 \times 10^{-4}$ |

# Part II

ABSTRACTIONS FOR FLEXIBILITY AND RELIABILITY ANALYSIS

# Chapter 6

FLEXIBILITY MEASURES

The content of the chapter is published in [52, 144]. The notation used here in Part II differs from that of Part I such that it is consistent with existing flexibility analysis literature.

## 6.1 Introduction

Flexibility seeks to quantify the ability of a physical system to counteract uncertainty (e.g. disturbances, parameters) in order to maintain feasible operation [46]. Grossmann and co-workers have proposed diverse formulations and algorithms to assess system flexibility [46, 37, 145]. The so-called *flexibility test problem* seeks to find recourse variables $\mathbf{z} \in \mathbb{R}^{n_z}$ that counteract the uncertain parameters $\boldsymbol{\theta} \in T \subseteq \mathbb{R}^{n_\theta}$ in order to satisfy the system constraints $f_j(\mathbf{z}, \boldsymbol{\theta}) \leq 0$, $j \in J$. The flexibility test problem is given by:

$$\chi := \max_{\boldsymbol{\theta} \in T} \quad \psi(\boldsymbol{\theta}). \tag{6.1}$$

Here, $\psi(\boldsymbol{\theta})$ is a function that tests the feasibility of the system at a fixed value of the parameters $\boldsymbol{\theta}$. This function is evaluated by searching for a recourse $\mathbf{z}$ that minimizes the

largest constraint violation:

$$\psi(\boldsymbol{\theta}) := \min_{\mathbf{z}} \max_{j \in J} \quad f_j(\mathbf{z}, \boldsymbol{\theta}). \tag{6.2}$$

If $\psi(\boldsymbol{\theta}) \leq 0$ the system has feasible operation at $\boldsymbol{\theta}$ and thus $\chi \leq 0$ indicates that the system has feasible operation over the entire uncertainty set $T$ (and is thus deemed to be flexible).

Problem (6.2) can be reformulated by using an upper-bounding variable $u \in \mathbb{R}$ as:

$$\begin{aligned} \psi(\boldsymbol{\theta}) = \quad & \min_{\mathbf{z}, u} \quad u \\ & \text{s.t.} \quad f_j(\mathbf{z}, \boldsymbol{\theta}) \leq u, \quad j \in J. \end{aligned} \tag{6.3}$$

The so-called *flexibility index problem* seeks to identify the largest uncertainty set $T(\delta)$ (where $\delta \in \mathbb{R}_+$ is a variable that scales the set) under which the system remains feasible. Formally, the flexibility index $F \in \mathbb{R}_+$ is defined as the solution of the problem:

$$\begin{aligned} F := \quad & \max_{\delta \in \mathbb{R}_+} \quad \delta \\ & \text{s.t.} \quad \max_{\boldsymbol{\theta} \in T(\delta)} \quad \psi(\boldsymbol{\theta}) \leq 0. \end{aligned} \tag{6.4}$$

Parameterizing the uncertainty set $T(\delta)$ in terms of a single scalar variable $\delta$ is in general difficult. Most studies reported in the literature assume a *hyperbox* set of the form:

$$T_{box}(\delta) = \{\boldsymbol{\theta} : \bar{\boldsymbol{\theta}} - \delta \Delta\boldsymbol{\theta}^- \leq \boldsymbol{\theta} \leq \bar{\boldsymbol{\theta}} + \delta \Delta\boldsymbol{\theta}^+\} \tag{6.5}$$

where $\bar{\boldsymbol{\theta}}$ is a nominal value and $\Delta\boldsymbol{\theta}^-, \Delta\boldsymbol{\theta}^+$ are maximum lower and upper deviations [46]. Figure 6.1 illustrates the use of this uncertainty set with a flexibility index problem.

Problems (6.1) and (6.4) are conceptually attractive, but are computationally challenging due to their nested nature [146]. Swaney and Grossmann proposed search procedures for the special case in which the solution of problems (6.1) and (6.4) correspond to vertices of $T_{box}$ and $T_{box}(\delta)$, as is the case when the system constraints $f_j(\cdot), j \in J$, are convex

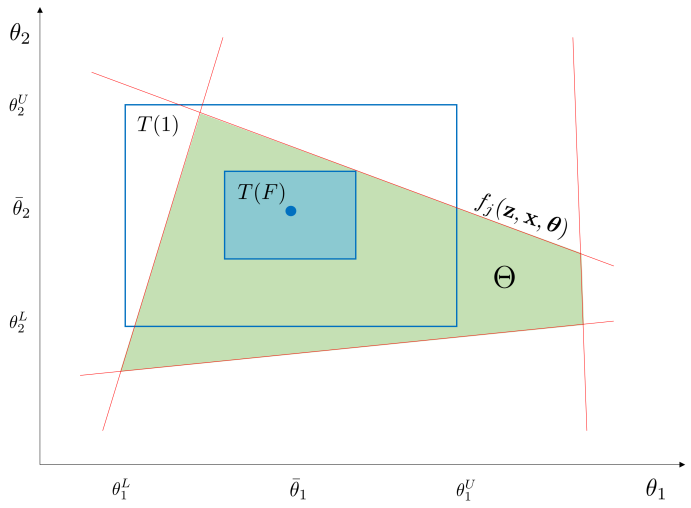



Figure 6.1: Illustration of flexibility index problem under a hyperbox uncertainty set.

[37, 145]. Vertex exploration suffers from computational limitations because an exponential number of vertices ($2^{n_\theta}$) must be evaluated [46]. Grossmann and Floudas developed an alternative approach that uses an active-set approach to compute the flexibility index under hyperbox sets [147]. For the case of linear constraints and hyperbox sets, this approach casts the flexibility index problem as a mixed-integer linear programming formulation.

Unfortunately, hyperbox representations of $T(\delta)$ do not adequately capture correlations in the parameters [148]. To address this issue, Rooney and Biegler [148] presented a methodology which considers ellipsoidal sets of the form:

$$T_{ellip} = \{\boldsymbol{\theta} : (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \chi^2_{n_p}(\alpha)\} \tag{6.6}$$

where $V_{\boldsymbol{\theta}} \in \mathbb{R}^{n_\theta \times n_\theta}$ is the covariance matrix of $\boldsymbol{\theta}$ (assumed to be symmetric positive definite), $n_\theta$ is the dimension of the uncertainty region, and $\chi^2_{n_\theta}(\alpha)$ is the critical value of a $\chi$-squared distribution with $n_\theta$ degrees of freedom and at probability level $\alpha$. Ellipsoidal uncertainty sets can be constructed from the level sets of multivariate Gaussian random variables, and thus can be used to capture correlations. In the approach of Rooney and

Biegler, the uncertainty set is discretized along the longest axis and the discrete points are used in combination with the flexibility test problem to assess the flexibility of the system. This approach is intuitive and practical, but only represents an approximation [148].

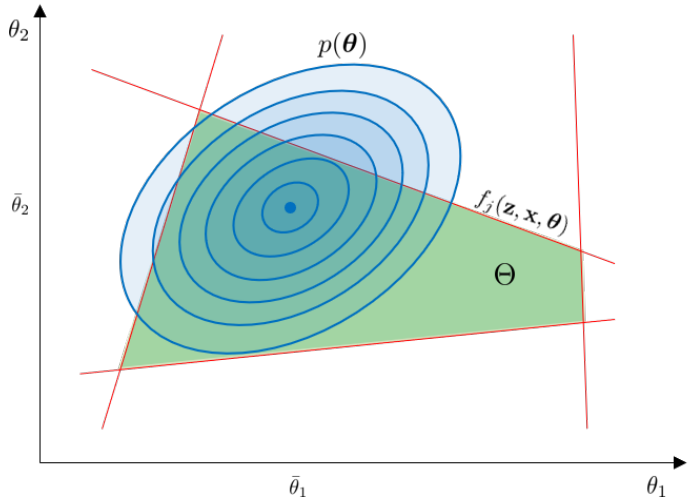



Figure 6.2: Illustration of stochastic flexibility index problem.

To capture more general characterizations of uncertainty, Straub and Grossmann developed a fundamentally different approach to flexibility analysis. This was done in terms of what they called the *stochastic flexibility index* [47]:

$$SF := \int_{\boldsymbol{\theta} \in \Theta} p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{6.7}$$

where $\Theta := \{\boldsymbol{\theta} \,|\, \psi(\boldsymbol{\theta}) \leq 0\}$ is the feasible set (projected onto the uncertainty space) and the random parameter $\boldsymbol{\theta}$ is characterized by the joint probability density function $p : \mathbb{R}^{n_\theta} \rightarrow \mathbb{R}$. Pistikopoulos and Mazzuchi proposed a similar definition in [149]. The stochastic flexibility index is interpreted as the probability that the system remains feasible. More specifically, it represents the probability of finding a recourse variable $z$ such that that the system constraints $f_j(\mathbf{z}, \boldsymbol{\theta}) \leq 0$, $j \in J$ are satisfied. Figure 6.2 illustrates this index over a simple constraint system. Consequently, we observe that this index shares common ground with joint chance constraints, in which one seeks to enforce the system constraints

with a desired level of probability. In particular, it is not difficult to see that the stochastic flexibility index satisfies $SF = \mathbb{P}(\psi(\boldsymbol{\theta}) \leq 0) = F_{\psi(\boldsymbol{\theta})}(0)$, where $F_{\psi(\boldsymbol{\theta})} : \mathbb{R} \to [0,1]$ is the cumulative density function of the random variable $\psi(\boldsymbol{\theta}) = \min_z \ \max_{j \in J} \ \ f_j(\mathbf{z}, \boldsymbol{\theta})$.

Index $SF$ can be obtained directly by integrating $p(\cdot)$ over the feasible region projected in the $\boldsymbol{\theta}$ space [46]. This can be done using Monte Carlo (MC) sampling and requires checking for feasibility at every sampled realization. It is well-known that MC has asymptotic convergence properties but might require an extremely large number of samples to cover the uncertainty space [49, 48]. This is particularly important in the context of flexibility analysis because limiting behavior is often found in the boundaries of the feasible region. To circumvent these issues, Straub and Grossmann proposed a quadrature scheme [47], which explores the random space in a more systematic manner and thus require far fewer samples than MC. Significant advances in quadrature schemes have been achieved in recent years due to the development of sparse grid techniques [150]. However, quadrature still suffers from severe computational limitations in high dimensions [50, 47].

This chapter proposes an approach to compute the flexibility index when uncertainty can be represented using multivariate Gaussian random variables. We show that the problem can be cast as a mixed-integer conic program (MICP) that seeks to find the largest radius of an ellipsoidal uncertainty set under which the system maintains feasible operation. This approach is based on the key observation that the active-set approach of Grossmann and Floudas can be applied to general uncertainty sets that can be parameterized using a scalar variable $\delta$. The proposed approach offers the ability to capture correlations and to provide probabilistic estimates of feasible operation. In particular, we show that the flexibility index can be used to compute a lower bound for the stochastic flexibility index. Moreover, we describe further extensions to traditional flexibility analysis which include: a generalization of the uncertainty set representation by using intersecting sets, systematic approaches for obtaining nominal points, and a methodology for identifying and ranking flexibility-limiting constraints.

## 6.2 Ellipsoidal Uncertainty Sets

In this section, we show ellipsoidal uncertainty sets can be incorporated into the flexibility index problem to accurately characterize Gaussian uncertainty.

### 6.2.1 *Mixed-Integer Formulations of Flexibility Index*

Grossmann and Floudas presented a mixed-integer programming approach for solving the flexibility test and index problems (6.1) and (6.4) [147]. This approach replaces the inner minimization problem (6.3) by its first-order Karush-Kuhn-Tucker (KKT) conditions and reformulates the complementarity conditions using binary variables:

$$\sum_{j \in J} \lambda_j = 1 \tag{6.8a}$$

$$\sum_{j \in J} \lambda_j \frac{\partial f_j(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} = 0 \tag{6.8b}$$

$$\lambda_j (f_j(\mathbf{z}, \boldsymbol{\theta}) - u) = 0 \quad j \in J \tag{6.8c}$$

$$f_j(\mathbf{z}, \boldsymbol{\theta}) - u + s_j = 0 \quad j \in J \tag{6.8d}$$

$$\lambda_j \geq 0, \; s_j \geq 0 \quad j \in J \tag{6.8e}$$

where $\lambda_j$ are the Lagrange multipliers of the system constraints and $s_j$ are the corresponding slack variables. When the system constraints are convex, the KKT conditions are necessary and sufficient. Otherwise, the first-order conditions can also be satisfied by saddle points and maxima. Note that, since this transformation is applied for a fixed value of $\boldsymbol{\theta}$, no restrictions on the shape of the uncertainty set are imposed in this derivation. Because the optimal solution of (6.3) occurs at $\psi(\boldsymbol{\theta}) = u$, the feasibility test problem

can be written as:

$$\begin{aligned} \chi = \max_{\boldsymbol{\theta} \in T} \quad & u \\ \text{s.t.} \quad & \sum_{j \in J} \lambda_j = 1 \\ & \sum_{j \in J} \lambda_j \frac{\partial f_j(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} = 0 \\ & \lambda_j (f_j(\mathbf{z}, \boldsymbol{\theta}) - u) = 0 \quad j \in J \\ & f_j(\mathbf{z}, \boldsymbol{\theta}) - u + s_j = 0 \quad j \in J \\ & \lambda_j \geq 0, \; s_j \geq 0 \qquad\quad j \in J. \end{aligned} \tag{6.9}$$

Problem (6.9) is equivalent to the feasibility constraint in the flexibility index problem (6.4). The complementarity conditions (6.8c) are nonlinear expressions that are difficult to handle computationally. To avoid this issue, we define binary variables $y_j \in \{0, 1\}$ and write the system of constraints:

$$\begin{aligned} & s_j \leq U(1 - y_j) \quad j \in J \\ & \lambda_j \leq y_j \qquad\quad j \in J \end{aligned} \tag{6.10}$$

where $U$ is a suitable upper bound for the slack variables $s_j$.

Previous work by Swaney and Grossmann has established conditions under which there exist exactly $n_z + 1$ active constraints [145]. This remarkable result is formalized in Theorem 1.

**Theorem 1.** *If the set of gradients* $\frac{\partial}{\partial \mathbf{z}}[f_j(\mathbf{z}, \boldsymbol{\theta})], j \in J$, *are linearly independent, then there exists* $n_z + 1$ *active constraints* $f_j(\boldsymbol{\theta}, \mathbf{z}) \leq 0$ *at the solution of* (6.9).

*Proof.* Given that the set of gradients $\frac{\partial}{\partial \mathbf{z}}[f_j(\mathbf{z}, \boldsymbol{\theta})], j \in J$, are linearly independent, (6.8b) requires that there exist $n > n_z$ nonzero Lagrange multipliers $\lambda_j$ since (6.8a) necessitates that at least one $\lambda_j$ be nonzero. The complete set of active constraints form a system of $n + n_\theta$ equations with $n_z + n_\theta + 1$ variables, thus a general solution requires $n \leq n_z + 1$. Hence, there must exist $n_z + 1$ active constraints at the solution to problem (6.9). □

This is a summary of the proof provided in [37] (which has been adapted to our context and introduced here for completeness). Theorem 1 can be used to bound the binary variables as:

$$\sum_{j \in J} y_j = n_z + 1. \tag{6.11}$$

This constraint can greatly facilitate the search of active constraints.

We now let (6.10) and (6.11) replace the constraints in (6.8c) and (6.8d) in (6.9) to produce the following mixed-integer formulation of the flexibility test problem:

$$\begin{aligned}
\chi = \max_{u,\boldsymbol{\theta},\mathbf{z},\lambda_j,s_j,y_j} \quad & u \\
\text{s.t.} \quad & f_j(\mathbf{z},\boldsymbol{\theta}) + s_j = u \qquad j \in J \\
& \sum_{j \in J} \lambda_j = 1 \\
& \sum_{j \in J} \lambda_j \frac{\partial f_j(\mathbf{z},\boldsymbol{\theta})}{\partial \mathbf{z}} = 0 \\
& s_j \leq U(1 - y_j) \qquad j \in J \\
& \lambda_j \leq y_j \qquad j \in J \\
& \sum_{j \in J} y_j = n_z + 1 \\
& \boldsymbol{\theta} \in T \\
& \lambda_j, s_j \geq 0, y_j \in \{0,1\} \quad j \in J.
\end{aligned} \tag{6.12}$$

When the system constraints are linear, the feasibility test problem is a mixed-integer linear program (MILP).

Swaney and Grossmann also proved that the flexibility index problem can be expressed as the minimum scaling value $\delta$ satisfying $\psi(\boldsymbol{\theta}) = 0$. Consequently, (6.4) can be formulated as:

$$\begin{aligned}
F = \min_{\delta \in \mathbb{R}_+,\ \boldsymbol{\theta} \in T(\delta)} \quad & \delta \\
\text{s.t.} \quad & \psi(\boldsymbol{\theta}) = 0.
\end{aligned} \tag{6.13}$$

This result is established in the following theorem and was originally proved in [37].

**Theorem 2.** *If $T(\delta)$ is a compact set and the system constraints $f_j(\mathbf{z}, \boldsymbol{\theta}), j \in J$, are continuous in $\mathbf{z}$ and $\boldsymbol{\theta}$, the flexibility index problem can be written as (6.13).*

*Proof.* Suppose that there exists a bounded solution $\delta^*$, $\boldsymbol{\theta}^*$ to problem (6.4) and that the system constraints $f_j(\mathbf{z}, \boldsymbol{\theta}), j \in J$, are continuous in both $\mathbf{z}$ and $\boldsymbol{\theta}$. Now assume that the solution $\boldsymbol{\theta}^*$ is such that $\psi(\boldsymbol{\theta}^*) < 0$. For any $\delta^* < \hat{\delta}$ we have

$$\max_{\boldsymbol{\theta} \in T(\delta^*)} \psi(\boldsymbol{\theta}) = \psi(\boldsymbol{\theta}^*) \leq \psi(\hat{\boldsymbol{\theta}}) = \max_{\boldsymbol{\theta} \in T(\hat{\delta})} \psi(\boldsymbol{\theta}). \tag{6.14}$$

Since $f_j(\mathbf{z}, \boldsymbol{\theta}), j \in J$, are continuous in both $\mathbf{z}$ and $\boldsymbol{\theta}$, it follows that $\psi(\boldsymbol{\theta})$ is a continuous function in $\boldsymbol{\theta}$ (as shown in [151]). Hence, for some $\epsilon > 0$ and a $\tilde{\delta}$ in the neighborhood of the solution $\delta^*$ we have

$$|\psi(\boldsymbol{\theta}^*) - \psi(\tilde{\boldsymbol{\theta}})| < \epsilon, \tag{6.15}$$

with $\psi(\tilde{\boldsymbol{\theta}}) = \max_{\boldsymbol{\theta} \in T(\tilde{\delta})} \psi(\boldsymbol{\theta})$. Rearranging (6.15) we obtain

$$\psi(\tilde{\boldsymbol{\theta}}) - \epsilon < \psi(\boldsymbol{\theta}^*) < 0 \tag{6.16}$$

and for $\epsilon$ sufficiently small it follows that

$$\psi(\tilde{\boldsymbol{\theta}}) < 0. \tag{6.17}$$

For a choice of $\tilde{\delta}$ arbitrary close to $\delta^*$ and which satisfies $\tilde{\delta} > \delta^*$, we have from (6.14) and (6.17) that $\tilde{\delta}$ is a feasible solution to problem (6.4). However, this is a contradiction to $\delta^*$ being a solution to (6.4). Thus, the assumption that $\psi(\boldsymbol{\theta}^*) < 0$ does not hold and $\psi(\boldsymbol{\theta}^*) = 0$.

Hence, for any compact set $T(\delta)$, the solution $\delta^*$ must lie on the boundary of the feasible region which is given by $\psi(\boldsymbol{\theta}) = 0$. Furthermore, the largest uncertainty set that can be fully inscribed in the feasible region is given by the smallest value of $\delta$ that satisfies

$\psi(\boldsymbol{\theta}) = 0$. Thus, problem (6.4) can be reformulated as (6.13). □

This result implies that the constraints of (6.12) can be substituted into problem (6.13). This gives the mixed-integer formulation:

$$
\begin{aligned}
F = \min_{\delta,\boldsymbol{\theta},\mathbf{z},\lambda_j,s_j,y_j} \quad & \delta \\
\text{s.t.} \quad & f_j(\mathbf{z},\boldsymbol{\theta}) + s_j = 0 && j \in J \\
& \sum_{j\in J} \lambda_j = 1 \\
& \sum_{j\in J} \lambda_j \frac{\partial f_j(\mathbf{z},\boldsymbol{\theta})}{\partial \mathbf{z}} = 0 \\
& s_j \le U(1-y_j) && j \in J \\
& \lambda_j \le y_j && j \in J \\
& \sum_{j\in J} y_j = n_z + 1 \\
& \boldsymbol{\theta} \in T(\delta) \\
& \lambda_j, s_j \ge 0, y_j \in \{0,1\} && j \in J.
\end{aligned}
\tag{6.18}
$$

When the system constraints are linear, this problem is also a MILP.

A key observation is that Theorems 1 and 2 hold for any compact set $T(\delta)$. A hyperbox representation of $T$ and $T(\delta)$ (denoted $T_{box}$ and $T_{box}(\delta)$) is often used because this yields a MILP formulation (provided that the constraints are linear). As we have discussed, however, hyperbox representations cannot adequately capture correlation information.

### 6.2.2 *Mixed-Integer Conic Formulation for Multivariate Gaussian Uncertainty*

We propose an approach to compute the flexibility index when uncertainty is represented as a multivariate Gaussian variable $\boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$. Although ellipsoidal regions have been explored in the literature, no approach has been proposed to identify the largest ellipsoidal region $T_{ellip}(\delta)$ for which feasible operation can be achieved. We now proceed

to show that this problem can be cast as a mixed-integer conic program and that the resulting flexibility index provides an alternative metric that captures parameter correlations. We will also show that this new index can be used to obtain a lower bound for the stochastic flexibility index.

We consider an ellipsoidal set of the form:

$$T_{ellip}(\delta) = \{\boldsymbol{\theta} : (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \delta\} \tag{6.19}$$

where $\delta \in \mathbb{R}_+$ is the radius of the ellipsoid (this variable is used to scale the set). The flexibility index problem that we propose thus seeks to find the largest radius $\delta$ for which feasible operation can be obtained. As we have seen, any compact set can be embedded in the flexibility index problem (6.18). We thus embed (6.19) into the flexibility index problem to obtain:

$$\begin{aligned}
\delta^* = \min_{\delta, \mathbf{z}, \boldsymbol{\theta}, \lambda_j, s_j, y_j} \quad & \delta \\
\text{s.t.} \quad & f_j(\mathbf{z}, \boldsymbol{\theta}) + s_j = 0 && j \in J \\
& \sum_{j \in J} \lambda_j = 1 \\
& \sum_{j \in J} \lambda_j \frac{\partial f_j(\mathbf{z}, \boldsymbol{\theta})}{\partial \mathbf{z}} = 0 \\
& s_j \leq U(1 - y_j) && j \in J \\
& \lambda_j \leq y_j && j \in J \\
& \sum_{j \in J} y_j = n_z + 1 \\
& (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \delta \\
& \lambda_j, s_j \geq 0; \quad y_j \in \{0, 1\} && j \in J.
\end{aligned} \tag{6.20}$$

Problem (6.20) is a mixed-integer conic program that directly characterizes the ellipsoidal uncertainty region via a conic constraint, thus avoiding the sub-optimality prob-

lems associated with ellipsoidal approximations [148]. The recent emergence of commercial MICP solvers can be exploited to make the solution of Problem (6.20) more computationally attractive [152].

Now we establish properties of the solution to Problem (6.20). We denote the solution of the problem as $(\delta^*, \boldsymbol{\theta}^*)$. We recall the feasible set of the system is given by $\Theta = \{\boldsymbol{\theta} \,|\, \psi(\boldsymbol{\theta}) \leq 0\}$ and note that its boundary is given by $\partial\Theta = \{\boldsymbol{\theta} \,|\, \psi(\boldsymbol{\theta}) = 0\}$. We also note that the boundary of the uncertainty set $T_{ellip}(\delta^*)$ is given by $\partial T_{ellip}(\delta^*) = \{\boldsymbol{\theta} \,|\, (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}^{-1}} (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) = \delta^*\}$. We recall that Theorem 2 proves that the critical parameter $\boldsymbol{\theta}^*$ lies in the boundary of the feasible set (i.e., $\boldsymbol{\theta}^* \in \partial\Theta$). We now proceed to show that the critical point also lies at the boundary of the uncertainty set $T_{ellip}(\delta^*)$ and that the entire uncertainty set $T_{ellip}(\delta^*)$ lies inside the feasible region $\Theta$ as shown in Figure 6.3.

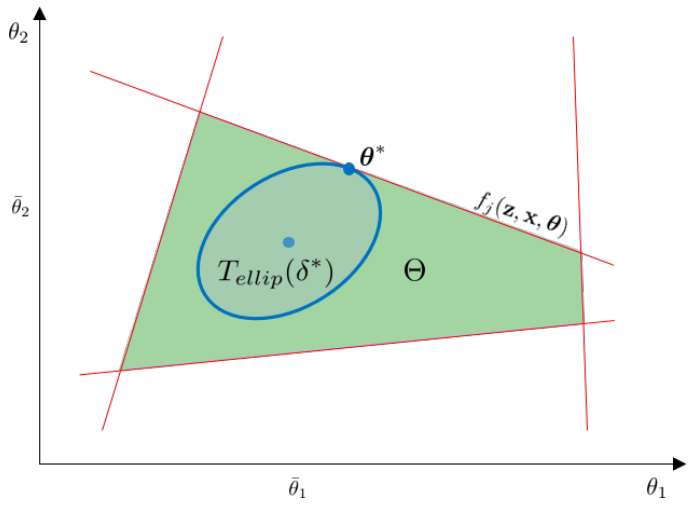



Figure 6.3: Illustration of Theorem 3.

**Theorem 3.** *The solution pair $(\delta^*, \boldsymbol{\theta}^*)$ satisfies the following properties: i) The uncertainty set is contained in the feasible set ($T_{ellip}(\delta^*) \subseteq \Theta$), ii) the critical parameter $\boldsymbol{\theta}^*$ lies on the boundary of the uncertainty set ($\boldsymbol{\theta}^* \in T_{ellip}(\delta^*)$), and iii) the critical parameter $\boldsymbol{\theta}^*$ lies in the intersection of the boundaries of the feasible and uncertainty sets ($\boldsymbol{\theta}^* \in \partial\Theta \cap \partial T_{ellip}(\delta^*)$).*

*Proof.* To prove i) assume there exists $\tilde{\boldsymbol{\theta}} \in T_{ellip}(\delta^*)$ but $\tilde{\boldsymbol{\theta}} \notin \Theta$. Since $\tilde{\boldsymbol{\theta}}$ is infeasible, we have that $\psi(\tilde{\boldsymbol{\theta}}) > 0$ which implies that $\psi(\tilde{\boldsymbol{\theta}}) > \psi(\boldsymbol{\theta}^*)$ because $\psi(\boldsymbol{\theta}^*) = 0$. This is a

contradiction because $\boldsymbol{\theta}^* \in \text{argmax}_{\boldsymbol{\theta} \in T_{ellip}(\delta^*)} \psi(\boldsymbol{\theta})$ and thus $\psi(\boldsymbol{\theta}^*) = 0$ is the maximum possible value of $\psi(\boldsymbol{\theta})$ in $T_{ellip}(\delta^*)$. To prove ii) we note that $\boldsymbol{\theta}^* \in \partial\Theta$ holds and, therefore, if $\boldsymbol{\theta}^* \notin \partial T_{ellip}(\delta^*)$ then there exists $\tilde{\boldsymbol{\theta}} \in T_{ellip}(\delta^*)$ with $\tilde{\boldsymbol{\theta}} \notin \Theta$. The result then follows from the argument used to prove i). The proof of iii) follows trivially from the observation that $\boldsymbol{\theta}^*$ lies on the boundary of both $\Theta$ and $T_{ellip}(\delta^*)$. □

Given the solution $\delta^*$ of the flexibility index problem (6.20), we can also compute the following quantity (that we refer to as the *confidence level*):

$$\alpha^* := F_{n_\theta}(\delta^*) \tag{6.21}$$

where $F_{n_\theta}(\delta^*)$ denotes the cumulative density of a $\chi$-squared distribution with $n_\theta$ degrees of freedom and at critical value $\delta^*$. The cumulative density can be computed from:

$$F_{n_\theta}(\delta^*) = \frac{\gamma(\frac{n_\theta}{2}, \frac{\delta^*}{2})}{\Gamma(\frac{n_\theta}{2})} \tag{6.22}$$

where $\gamma(\cdot)$ and $\Gamma(\cdot)$ are the incomplete and complete gamma functions [153]. The reasoning behind the definition of the confidence level becomes clear in the following theorem.

**Theorem 4.** *The flexibility index $\delta^*$ and associated confidence level $\alpha^*$ satisfy the following properties: i) $\alpha^* = \mathbb{P}(\boldsymbol{\theta} \in T_{ellip}(\delta^*))$, ii) $\alpha^*$ is a lower bound for the stochastic flexibility index (i.e., $\alpha^* \leq SF$), and iii) $\delta^*$ is the critical value (quantile) associated with confidence level $\alpha^*$.*

*Proof.* Because the covariance matrix is symmetric positive definite, we can apply an eigenvalue decomposition of the form $V_{\boldsymbol{\theta}}^{-1} = Q\Lambda Q^{-1}$ to express the ellipsoidal constraint $(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1}(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \delta$ as $\mathbf{x}^T\mathbf{x} \leq \delta$ or $\sum_{i=1}^{n_\theta} x_i^2 \leq \delta$. Here, $\mathbf{x} = \sqrt{\Lambda}Q^{-1}(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})$, $\Lambda \in \mathbb{R}^{n_\theta \times n_\theta}$ is a diagonal matrix containing the eigenvalues of the covariance matrix, and where $Q \in \mathbb{R}^{n_\theta \times n_\theta}$ is an orthogonal matrix that contains the corresponding eigenvectors (thus $Q^{-1} = Q^T$). We also have that $x_i \sim \mathcal{N}(0,1)$ for all $i = 1, ..., n_\theta$ (this can be shown by using the property that $\mathbf{x}$ is a linear transformation of the Gaussian random variable $\boldsymbol{\theta}$ and thus is also Gaussian). Consequently, $\mathbf{x}^T\mathbf{x} = \sum_{i=1}^{n_\theta} x_i^2$ follows a $\chi$-squared distribution

with $n_\theta$ degrees of freedom and $\mathbb{P}(\boldsymbol{\theta} \in T_{ellip}(\delta)) = \mathbb{P}(\mathbf{x}^T\mathbf{x} \leq \delta)$. The cumulative density of a $\chi$-squared random variable at critical value $\delta$ is given by (6.21). Consequently, we have that $\alpha^* = \mathbb{P}(\boldsymbol{\theta} \in T_{ellip}(\delta^*))$, establishing i).

To prove ii), we recall from Theorem 3 that $T_{ellip}(\delta^*) \subseteq \Theta$ and thus:

$$\begin{aligned}
\alpha^* &= \mathbb{P}(\boldsymbol{\theta} \in T_{ellip}(\delta^*)) \\
&= \int_{\boldsymbol{\theta} \in T(\delta^*)} p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&\leq \int_{\boldsymbol{\theta} \in \Theta} p(\boldsymbol{\theta}) d\boldsymbol{\theta} \\
&= \mathbb{P}(\boldsymbol{\theta} \in \Theta) \\
&= SF,
\end{aligned}$$

Consequently, $\alpha^*$ provides a lower bound for the stochastic flexibility index. To prove iii), we note that inversion of the cumulative density function (6.21) is given by its quantile (critical value), which is given by $\delta^*$. □

**Remarks:** As noted in the proof of Theorem 4, the conic constraint $(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1} (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \delta$ can be expressed as a sum of squares constraint $\sum_{i=1}^{n_\theta} x_i^2 \leq \delta$ by using an eigenvalue decomposition of the covariance matrix. This can facilitate numerical implementation. We also note that the shapes of the feasible region $\Theta$ and of the uncertainty set $T_{ellip}(\delta^*)$ affect the gap between the confidence level $\alpha^*$ and the stochastic flexibility index $SF$.

## 6.3 Generalized Uncertainty Sets

Following the results from Section 6.2, Problem (6.18) can use any compact set $T(\delta)$ whose size can be parameterized in terms of a scalar variable $\delta$. The selection of $T(\delta)$ will depend on the application and on the nature of the uncertain parameters $\boldsymbol{\theta}$. We can envision a number of uncertainty sets that can be used in conjunction with Problem (6.18). Table 6.1 summarizes such sets. We note that the ellipsoidal norm set in Table 6.1 is equivalent

to $T_{ellip}(\delta)$ as defined in (6.6) provided that $A = V_{\boldsymbol{\theta}}^{-1}$. The ellipsoidal norm set can also be used to approximate polytopes [154]. We also note that $T_{box}(\delta)$ is equivalent to $T_{\infty}(\delta)$ when $\Delta\boldsymbol{\theta}_i^- = \Delta\boldsymbol{\theta}_i^+ = 1$. Li et.al. provide an analysis of the $\ell_1$, $\ell_2$, and $\ell_\infty$-norm uncertainty sets along with their mathematical properties in the context of robust optimization [155]. The conditional value at risk (CVaR) norm, originally proposed by Pavlikov and Uryasev in [156], has gained interest in the robust and stochastic optimization communities because of its ability to approximate $\ell_p$ norms via linear programming (precisely reproducing the $\ell_1$ and $\ell_\infty$ norms as extreme cases) [157].

Table 6.1: Potential representations for the uncertainty set $T(\delta)$.

| Name | Uncertainty Set |
|---|---|
| Ellipsoidal Norm | $T_{ellip}(\delta) = \{\boldsymbol{\theta} : \|\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|\|_A^2 \leq \delta\}$ |
| Hyperbox Set | $T_{box}(\delta) = \{\boldsymbol{\theta} : \bar{\boldsymbol{\theta}} - \delta\Delta\boldsymbol{\theta}^- \leq \boldsymbol{\theta} \leq \bar{\boldsymbol{\theta}} + \delta\Delta\boldsymbol{\theta}^+\}$ |
| $\ell_\infty$ Norm | $T_{\infty}(\delta) = \{\boldsymbol{\theta} : \|\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|\|_\infty \leq \delta\}$ |
| $\ell_1$ Norm | $T_1(\delta) = \{\boldsymbol{\theta} : \|\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|\|_1 \leq \delta\}$ |
| $\ell_2$ Norm | $T_2(\delta) = \{\boldsymbol{\theta} : \|\|\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\|\|_2 \leq \delta\}$ |
| CVaR norm | $T_{CVaR}(\delta) = \{\boldsymbol{\theta} : \langle\langle\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}\rangle\rangle_\alpha \leq \delta\}$ |

Li et. al. also analyzed sets that result from intersections (compositions) of $\ell_p$-norm sets (e.g., $T_2(\delta) \cap T_\infty(\delta)$) [155]. Inspired by this, we observe that $T(\delta)$ can be represented by any combination of the uncertainty sets in Table 6.1 or other sets (provided that the resulting combined set is compact). This provides the ability to incorporate *physical knowledge* on the uncertain parameters and/or to create a wider range of uncertainty set *shapes*. For instance, the positive (truncated) ellipsoidal set $T_{ellip+}(\delta) := T_{ellip}(\delta) \cap \mathbb{R}_+^{n_\theta}$, where $\mathbb{R}_+^{n_\theta} := \{\boldsymbol{\theta} : \boldsymbol{\theta} \geq 0\}$ can be used to represent parameters that are known to be non-negative (e.g., demands, prices). Similarly, we can define the positive hyperbox set $T_{box+}(\delta) := T_{box}(\delta) \cap \mathbb{R}_+^{n_\theta}$. Compositions of sets can be naturally accommodated in conjunction with Problem (6.18), by imposing the constraints corresponding to each set. For instance, $T_{ellip+}(\delta)$ is represented by adding the constraints $(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^T V_{\boldsymbol{\theta}}^{-1}(\boldsymbol{\theta} - \bar{\boldsymbol{\theta}}) \leq \delta$ and $\boldsymbol{\theta} \geq 0$ to the MIP formulation.

## 6.4 Nominal Point Selection

Most uncertainty sets require a nominal point $\bar{\boldsymbol{\theta}}$. The choice of the nominal point $\bar{\boldsymbol{\theta}}$ is critical; for example, a nominal point that is close to the boundary of the feasible region will have limited flexibility. In some applications, specifying $\bar{\boldsymbol{\theta}}$ might be difficult if not enough data is available or if a system is not routinely operated at any given point (e.g., a power grid). Also, optimal design centering problems have been of general interest [151]. Thus, we introduce methods that can be employed to find well-centered nominal points. We note that one would be tempted to let $\bar{\boldsymbol{\theta}}$ be a free variable in Problem (6.18) to find a nominal point, but this approach leads to trivial solutions. This is because the objective is minimized when the nominal point $\bar{\boldsymbol{\theta}}$ is placed on the boundary of $\boldsymbol{\theta}$. Consequently, this naive approach cannot be used to compute a nominal point.

We propose an extension of the *analytic center* (commonly used in interior-point methods) to compute a nominal point. The analytic center $\bar{\boldsymbol{\theta}}_{ac}$ is given by:

$$\begin{aligned} \bar{\boldsymbol{\theta}}_{ac} := \quad & \underset{\boldsymbol{\theta},\mathbf{z},\mathbf{x},s_j}{\operatorname{argmax}} \quad \sum_{j\in J} \log\left(s_j\right) \\ & \text{s.t.} \quad f_j(\mathbf{z},\mathbf{x},\boldsymbol{\theta}) + s_j \leq 0, \quad j \in J \\ & \qquad\quad h_i(\mathbf{z},\mathbf{x},\boldsymbol{\theta}) = 0, \qquad\quad i \in I. \end{aligned} \tag{6.23}$$

Here, $s_j \in \mathbb{R}$ are slack variables. This problem pushes the constraints to the interior of the feasible region. Any bounded solution of the above problem satisfies $s_j > 0$ and thus $f_j(\mathbf{z},\mathbf{x},\bar{\boldsymbol{\theta}}_{ac}) < 0$ for all $j \in J$. Consequently, we have that $\psi(\bar{\boldsymbol{\theta}}_{ac}) < 0$ and thus $\bar{\boldsymbol{\theta}}_{ac} \in \Theta$ (the analytic center is feasible). We also note that the analytic center is the point that maximizes the geometric mean of the constraints (slack variables). By changing the objective to $\sum_{j\in J} s_j$ one finds a center point that maximizes the arithmetic mean.

We also propose a center point $\bar{\boldsymbol{\theta}}_{fc}$ (that we call the *feasible center*) and that we define

as:

$$\begin{aligned} \bar{\boldsymbol{\theta}}_{fc} := \quad & \underset{\boldsymbol{\theta},\mathbf{z},\mathbf{x},s}{\operatorname{argmax}} \quad s \\ & \text{s.t.} \quad f_j(\mathbf{z},\mathbf{x},\boldsymbol{\theta}) + s \leq 0, \quad j \in J \\ & \qquad\quad h_i(\mathbf{z},\mathbf{x},\boldsymbol{\theta}) = 0, \qquad i \in I. \end{aligned} \tag{6.24}$$

Here, $s \in \mathbb{R}$ is a slack variable. The feasible center is the point that maximizes the worst-case slack variable (as opposed to the geometric mean used in the analytic center). We also note that $\bar{\boldsymbol{\theta}}_{fc} = \operatorname{argmin}_{\boldsymbol{\theta}} \ \psi(\boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} \ -\psi(\boldsymbol{\theta})$ and that $\psi(\bar{\boldsymbol{\theta}}_{fc}) \leq 0$. Moreover, (6.24) is equivalent to (6.3) when the parameter $\theta$ is set as a free variable. Figure 6.4 juxtaposes these two different center point definitions.

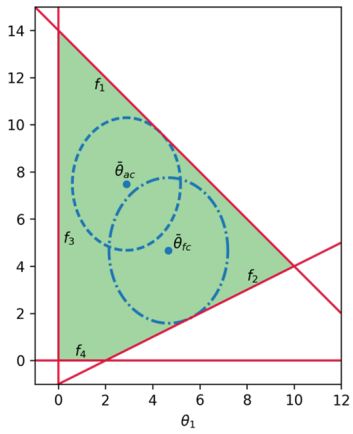



Figure 6.4: Illustration of the analytical and feasible centers for a simple constraint system.

The feasible center is an approximation of the center point:

$$\bar{\boldsymbol{\theta}}_d := \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \ d(\boldsymbol{\theta}, \partial\Theta) \tag{6.25}$$

where $d(\boldsymbol{\theta}, \partial\Theta) = \min_{\boldsymbol{\theta}' \in \partial\Theta} \ \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$. In other words, $\bar{\boldsymbol{\theta}}_d$ is the point that maximizes the distance to the boundary of feasible set. This point is a natural center point, but it is difficult to compute due to the nested max and min operators. We now proceed to show that $|-\psi(\boldsymbol{\theta})|$ provides a lower bound of $d(\boldsymbol{\theta}, \partial\Theta)$. Consequently, because the feasible center $\bar{\boldsymbol{\theta}}_{fc}$ maximizes $-\psi(\boldsymbol{\theta})$, we have that this also implicitly maximizes $d(\boldsymbol{\theta}, \partial\Theta)$. This

property can be established as follows, if the constraints are Lipschitz continuous (as often assumed in the flexibility analysis literature), we have that $\psi(\cdot)$ is Lipschitz as well and thus:

$$|\psi(\hat{\boldsymbol{\theta}}) - \psi(\boldsymbol{\theta})| \leq L\|\bar{\boldsymbol{\theta}}_d - \boldsymbol{\theta}\| \tag{6.26}$$

where $\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}' \in \partial\Theta} \ \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$ for a particular value of $\boldsymbol{\theta}$. Because $\hat{\boldsymbol{\theta}} \in \partial\Theta$, we have that $\psi(\hat{\boldsymbol{\theta}}) = 0$ and thus,

$$\begin{aligned} |\psi(\hat{\boldsymbol{\theta}}) - \psi(\boldsymbol{\theta})| &= |-\psi(\boldsymbol{\theta})| \\ &\leq L\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\| \\ &= L \min_{\boldsymbol{\theta}' \in \partial\Theta} \ \|\boldsymbol{\theta}' - \boldsymbol{\theta}\| \\ &= L\, d(\boldsymbol{\theta}, \partial\Theta). \end{aligned} \tag{6.27}$$

We thus conclude that:

$$d(\boldsymbol{\theta}, \partial\Theta) \geq \frac{1}{L}|-\psi(\boldsymbol{\theta})|. \tag{6.28}$$

Consequently, maximizing $-\psi(\boldsymbol{\theta})$ implicitly maximizes $d(\boldsymbol{\theta}, \partial\Theta)$. Moreover, we note that $d(\boldsymbol{\theta}, \partial\Theta) = -\psi(\boldsymbol{\theta}) = 0$ if $\boldsymbol{\theta} \in \partial\Theta$.

## 6.5 Design Comparison

The flexibility index is a valuable metric that can be used to compare system designs. The utility and interpretation of the index for a system will depend on the choice of $T(\delta)$. For instance, the index can be used to determine the confidence level $\alpha^*$ if the set $T_{ellip}(\delta)$ is used (when $\boldsymbol{\theta} \ \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$). The index might not have a clear interpretation or utility for some choices of $T(\delta)$ but can still be useful to quantify *improvements in flexibility* from design modifications (retrofits). We illustrate this feature by using the following example.

Consider two system designs that are each subjected to two normal random variables and that have no recourse variables $\mathbf{z}$ as summarized in Table 6.2. The nominal point $\bar{\boldsymbol{\theta}}$

Table 6.2: System constraints of Design A and Design B.

| | Design A | Design B |
|---|---|---|
| $f_1:$ | $\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2 - 14 \leq 0$ | $\frac{3}{4}\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2 - 14 \leq 0$ |
| $f_2:$ | $\boldsymbol{\theta}_1 - 2\boldsymbol{\theta}_2 - 2 \leq 0$ | $\frac{3}{4}\boldsymbol{\theta}_1 - 2\boldsymbol{\theta}_2 - 2 \leq 0$ |
| $f_3:$ | $-\boldsymbol{\theta}_1 \leq 0$ | $-\boldsymbol{\theta}_1 \leq 0$ |
| $f_4:$ | $-\boldsymbol{\theta}_2 \leq 0$ | $-\boldsymbol{\theta}_2 \leq 0$ |

and the covariance matrix $V_{\boldsymbol{\theta}}$ are taken to be:

$$\bar{\boldsymbol{\theta}} = \begin{bmatrix} 4 \\ 5 \end{bmatrix} \qquad V_{\boldsymbol{\theta}} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}. \tag{6.29}$$

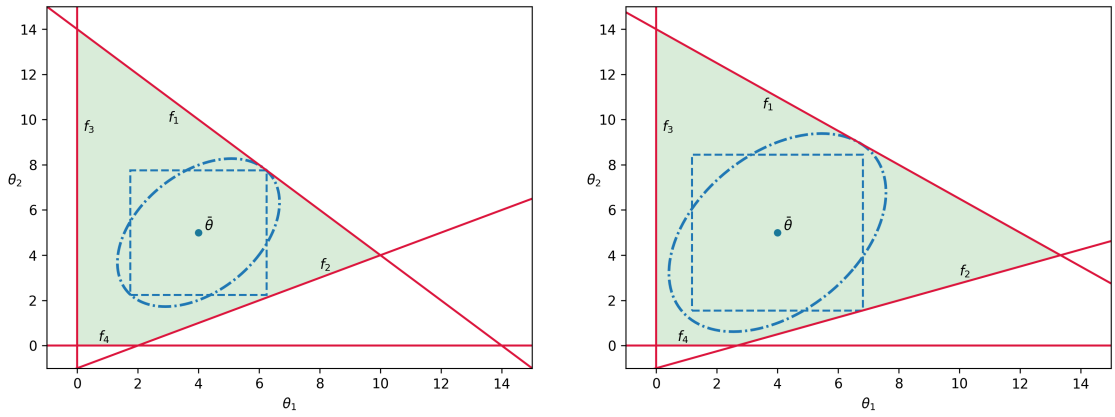
Given that the parameters are multivariate Gaussian, we choose to use $T_{ellip}(\delta)$. We also compute $F_{box}$ using $T_{box}(\delta)$, where $\Delta\boldsymbol{\theta}^-$ and $\Delta\boldsymbol{\theta}^+$ are both taken to be $(4.243, 5.196)$. These bounds correspond to $\bar{\boldsymbol{\theta}}_i \pm 3\sigma_i$ confidence bounds where $\sigma_i$ is the standard deviation of $\boldsymbol{\theta}_i$. For comparison, the $SF$ index is rigorously computed using 100,000 MC samples. Table 6.3 summarizes the results. Both flexibility indexes indicate that Design B is more flexible. We also see that the confidence level $\alpha^*$ also provides a conservative approximation of $SF$.

Table 6.3: Comparing the flexibility of Design A and Design B.

| | $F_{box}$ | $F_{ellip}$ | $\alpha^*$ (%) | $SF$-MC (%) |
|---|---|---|---|---|
| Design A | 0.53 | 3.57 | 83.1 | 96.6 |
| Design B | 0.66 | 6.40 | 95.9 | 98.9 |

Figure 6.5 depicts the two different systems. It is apparent that Design B has a larger feasible region $\Theta$ and thus has a larger $SF$ index. A key observation is that the choice of $T(\delta)$ significantly affects how it measures system flexibility. For instance, for Design B, the ellipsoidal and hyperbox sets identify different limiting constraints, and thus exhibit

distinct behavior. In particular, index $F_{box}$ is limited by constraint $f_2$ and would not change if constraints $f_1$, $f_3$, and/or $f_4$ were varied to increase the area of $\Theta$, even though the overall system flexibility would be improved. On the other hand, index $F_{ellip}$ is limited by constraint $f_1$. This conservative behavior parallels what is observed with the use of uncertainty sets in robust optimization, where such sets are used to optimize against the worst case scenario such that the solution is robust in the face of uncertainty, and the conservativeness of a solution depends on the chosen shape of the uncertainty set [158].




(a) Design A (b) Design B

Figure 6.5: Elliptical and hyperbox uncertainty sets relative to feasible regions of Designs A and B.

## 6.6 Limiting Constraint Ranking

An important observation that we make is that the flexibility index problem also implicitly identifies inequality constraints that limit flexibility. Such information can be valuable in design or operating procedures. Specifically, at a solution of Problem (6.18), the binary variables $y_j$ indicate which inequality constraints are active and therefore limit flexibility. Problem (6.18) can thus be resolved by excluding this first set of limiting constraints, so that the next set of limiting constraints can be identified. This step can be repeated to rank system constraints to a desired extent (as long as the solution of Problem (6.18) is bounded). This is done by noticing that each set of limiting constraints will have an

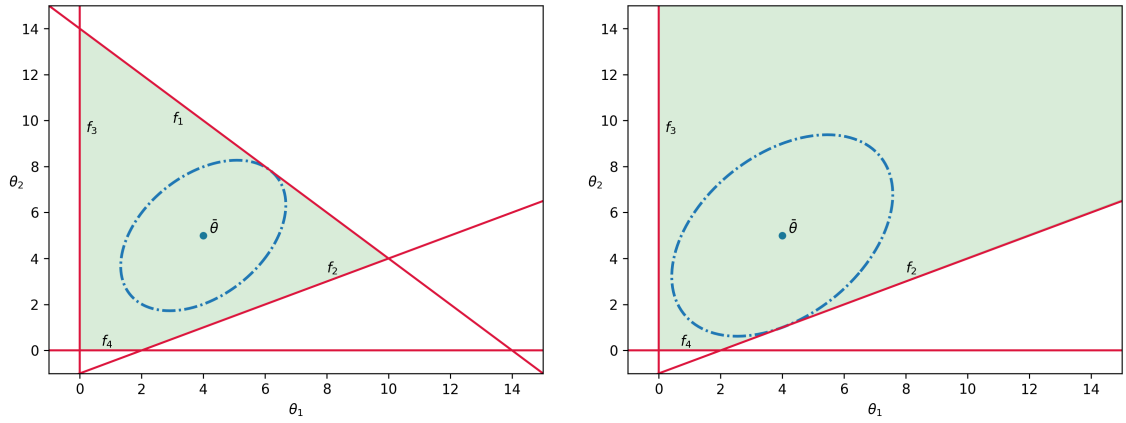
associated flexibility index, which indicates how limiting a particular set of constraints is relative to the other sets. We note that this analysis operates under the assumption that the inequality constraints can be eliminated/relaxed such that the system still has physical meaning. However, this assumption is met in many applications since constraints that cannot be removed (e.g., physical laws) typically correspond to equality constraints.

This methodology can also be used to identify and rank *system components* that limit flexibility in order to guide design improvements or quantify value of specific components, since such identification is useful to understand and identify system vulnerabilities. Thus, this ranking becomes useful for complex systems where it is difficult to determine vulnerable components. For example, this approach can identify which heat-exchangers in a heat-exchanger network could be retrofitted to most increase the system flexibility, and similarly identify heat-exchangers that would not lead to increased flexibility when improved. Limiting constraint information can also be used to quantify the impact of *failure* of system components (e.g., a production facility).

This proposed approach shares some common ground with optimal design problems where design variables $\mathbf{d}$ (added to the system constraints $f_j(\mathbf{z}, \mathbf{x}, \boldsymbol{\theta})$ and $h_i(\mathbf{z}, \mathbf{x}, \boldsymbol{\theta})$) are selected to minimize cost and satisfy a specified index $F$ [159, 160, 161]. This provides a straightforward approach for selecting a system design that is sufficiently flexible prior to its determination. However, our approach differs in that it can be used to fully characterize how each component of a particular system design impacts system flexibility instead of choosing design variables to improve flexibility to desired extent. This becomes useful for intelligently retrofitting existing systems and for understanding which components most limit flexibility (i.e., are most vulnerable to uncertainties) that therefore warrant close monitoring.

We illustrate the constraint ranking methodology using Design A (described in Section 6.5). Here, we choose the set $T_{ellip}(\delta)$. Problem (6.18) is solved to identify the limiting constraints, which are then eliminated to solve the problem again to identify a new set of limiting constraints. In this case, the problem is solved four times as only one constraint

becomes active at the solution of each problem. The results are summarized in Table 6.4. We observe that constraint $f_1$ limits the flexibility index the most. Specifically, the value of $F_{ellip}$ corresponding to the next limiting constraint $f_2$ is 79.3% larger. Diminishing changes in $F_{ellip}$ are obtained for subsequent limiting constraints, showing that constraints $f_2$-$f_4$ have little impact on the system flexibility relative to $f_1$.




(a) Rank 1: Constraint $f_1$ is limiting

(b) Rank 2: Constraint $f_2$ is limiting

(c) Rank 3: Constraint $f_3$ is limiting

(d) Rank 4: Constraint $f_4$ is limiting

Figure 6.6: Identifying and ranking limiting constraints using an elliptical uncertainty set.

Figure 6.6 depicts the solutions corresponding to the results presented in Table 6.4. It is apparent that the uncertainty set is larger after each subsequent ranking step. We again make the observation that the limiting constraints are determined by the shape of the uncertainty set. Thus, different uncertainty sets can yield distinct classifications of

limiting constraints.

Table 6.4: Constraint ranking results for Design A.

| | Active Constraint | $F_{ellip}$ | Flexibility Increase (%) |
|---|---|---|---|
| Rank 1 | $f_1$ | 3.57 | – |
| Rank 2 | $f_2$ | 6.40 | 79.3 |
| Rank 3 | $f_3$ | 8.00 | 124.1 |
| Rank 4 | $f_4$ | 8.40 | 135.3 |

## 6.7 Case Studies

We apply the flexibility analysis framework to distribution networks. We consider a simple three-node network and power distribution networks from standard libraries (IEEE-14 and 141-Bus Network). These results seek to highlight interesting insights that can be gained with the framework. The data used in the power networks is obtained from the `MATPOWER` package [162]. All formulations are implemented in `FlexibilityAnalysis.jl v0.1.0` (https://github.com/pulsipher/FlexibilityAnalysis.jl), a Julia package we have developed that extends `JuMP` [163] to streamline the implementation of the flexibility analysis framework presented in this paper. These formulations are solved using Pavito to leverage Gurobi 7.5.2 in combination with IPOPT for efficient solution of general convex MIPs or using Pajarito to leverage Gurobi 7.5.2 for efficient solution of conic MIPs.

### 6.7.1 *Physical Model*

Each network is modeled by performing balances at each node $n \in \mathcal{C}$ and enforcing capacity constraints on the arcs $a_k, k \in \mathcal{A}$, and on the suppliers $s_b, b \in \mathcal{S}$. The demands $d_m, m \in \mathcal{D}$, are assumed to be the uncertain parameters. The flexibility index problem thus seeks to identify the largest set of simultaneous demand withdrawals that the system can

tolerate. The deterministic network model is given by:

$$\sum_{k \in \mathcal{A}_n^{rec}} a_k - \sum_{k \in \mathcal{A}_n^{snd}} a_k + \sum_{b \in \mathcal{S}_n} s_b - \sum_{m \in \mathcal{D}_n} d_m = 0, \quad n \in \mathcal{C} \tag{6.30a}$$

$$-a_k^C \leq a_k \leq a_k^C, \quad k \in \mathcal{A} \tag{6.30b}$$

$$0 \leq s_b \leq s_b^C, \quad b \in \mathcal{S} \tag{6.30c}$$

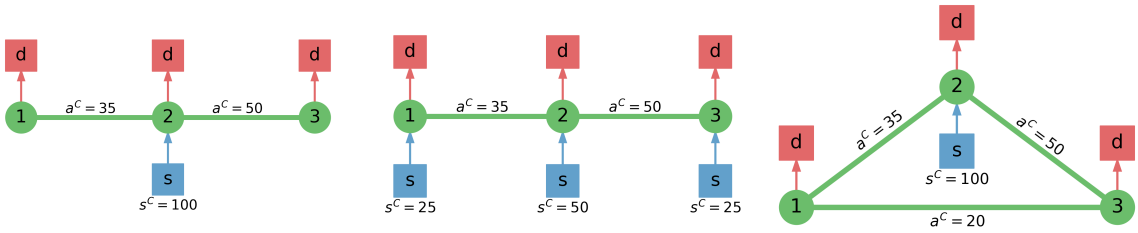
where $\mathcal{A}_n^{rec}$ denotes the set of receiving arcs at node $n$, $\mathcal{A}_n^{snd}$ denotes the set of sending arcs at $n$, $\mathcal{S}_n$ denotes the set of suppliers at $n$, $\mathcal{D}_n$ denotes the set of demands at $n$, $a_k^C$ are the arc capacities, and $s_b^C$ are the supplier capacities. The Lagrange multipliers associated with the constraints in (6.30b) are denoted as $\lambda_k^L$ (corresponding to the lower arc bounds) and $\lambda_k^U$ (corresponding to the upper bounds). Similarly, we define multipliers corresponding to the supply constraints in (6.30c) as $\gamma_b^L$ and $\gamma_n^U$.

### 6.7.2 *3-Node Network*

We consider three-node distribution network designs (see Figure 6.7). Design 1 corresponds to a centralized supplier base case [164]. Design 2 differs from Design 1 in that it employs a decentralized supply scheme, and Design 3 differs from Design 1 in that it contains an additional transportation arc. Each design is subjected to multivariate Gaussian demands $\boldsymbol{\theta} = (d_1, d_2, d_3) \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$. The nominal point $\bar{\boldsymbol{\theta}}$ is set to be the analytic and feasible centers. The center points obtained are given by $\bar{\boldsymbol{\theta}}_{ac} = (0.0, 50.0, 0.0)$ and $\bar{\boldsymbol{\theta}}_{fc} = (0.0, 37.4, 0.0)$. The covariance matrix $V_{\boldsymbol{\theta}}$ is assumed to be:

$$V_{\boldsymbol{\theta}} = \begin{bmatrix} 80 & \beta & \beta \\ \beta & 80 & \beta \\ \beta & \beta & 120 \end{bmatrix} \tag{6.31}$$

where $\beta \in \{-40, 0, 50\}$ is a parameter that captures covariance. These choices of $\bar{\boldsymbol{\theta}}$ and $V_{\boldsymbol{\theta}}$ are used in conjunction with four types of uncertainty sets: $T_{ellip}(\delta)$, $T_{ellip+}(\delta)$, $T_{box}(\delta)$, and $T_{box+}(\delta)$. The hyperbox deviations $\Delta\boldsymbol{\theta}^-, \Delta\boldsymbol{\theta}^+$ are taken to be $(26.833, 26.833, 32.863)$, which correspond to $\bar{\boldsymbol{\theta}}_i \pm 3\sigma_i$ confidence bounds. Detailed results for all uncertainty sets and designs are provided in Appendix A.1. The optimality of each solution was verified by evaluating feasibility of 10,000 points generated randomly along the optimized uncertainty set boundaries. The stochastic flexibility index was also estimated for each instance by using 1,000,000 MC samples.




(a) Design 1 (b) Design 2 (c) Design 3

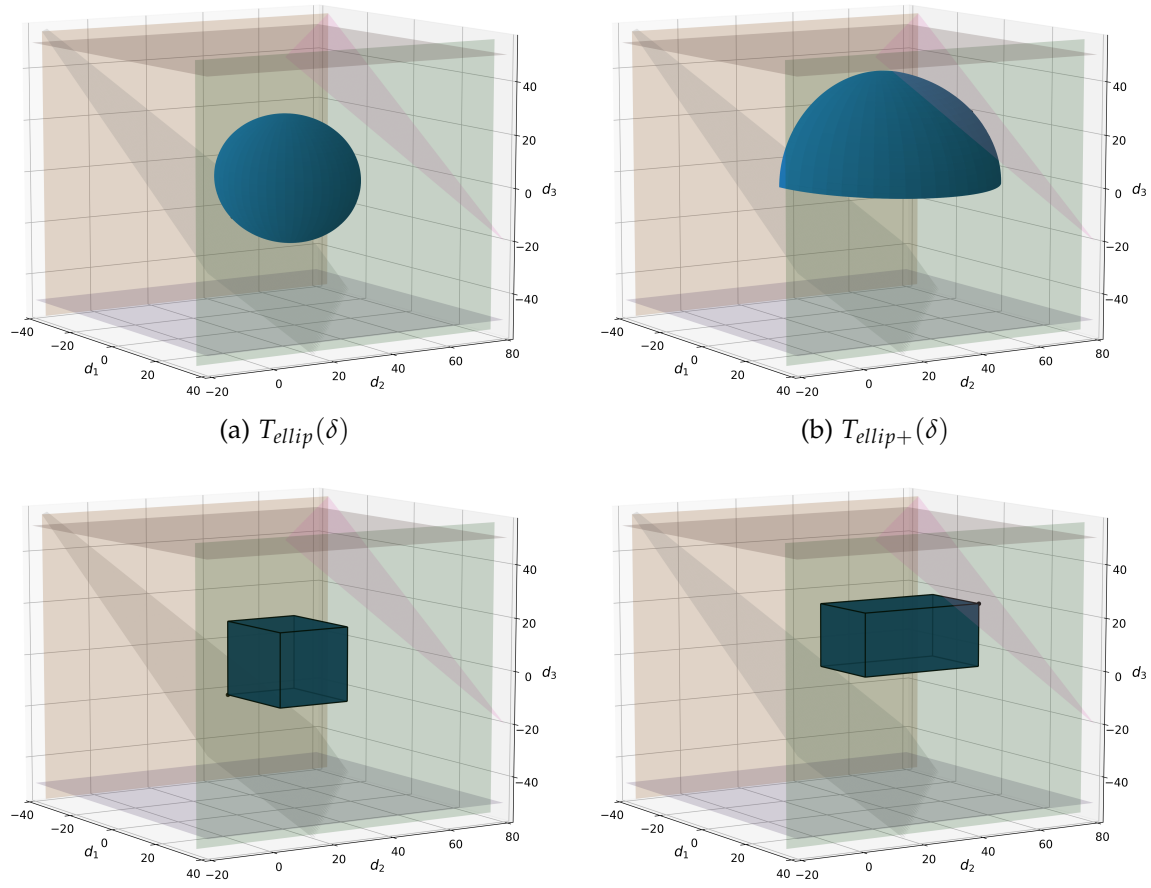
Figure 6.7: Schematics of three candidate 3-node network designs.

***Uncertainty Set and Nominal Point Comparison***

Table 6.5: Flexibility index results for Design 1 of three-node distribution network with $\beta = 0$.

| Center | Set | $F$ | Active Constraint | $SF$-MC (%) | Solution Time (s) |
|---|---|---|---|---|---|
| analytic | $T_{ellip}(\delta)$ | 8.93 | $\gamma_2^L$ | 99.71 | 0.18 |
| | $T_{ellip+}(\delta)$ | 8.93 | $\gamma_2^U$ | 99.44 | 0.13 |
| | $T_{box}(\delta)$ | 0.578 | $\gamma_2^L$ | 99.71 | 0.03 |
| | $T_{box+}(\delta)$ | 0.578 | $\gamma_2^U$ | 99.44 | 0.04 |
| feasible | $T_{ellip}(\delta)$ | 5.00 | $\gamma_2^L$ | 98.71 | 0.15 |
| | $T_{ellip+}(\delta)$ | 14.00 | $\gamma_2^U$ | 99.95 | 0.09 |
| | $T_{box}(\delta)$ | 0.432 | $\gamma_2^L$ | 98.71 | 0.03 |
| | $T_{box+}(\delta)$ | 0.723 | $\gamma_2^U$ | 99.95 | 0.03 |

To illustrate the differences between the various uncertainty sets and center points, we consider those used in conjunction with Design 1. Table 6.5 summarizes these results




(a) $T_{ellip}(\delta)$ (b) $T_{ellip+}(\delta)$

(c) $T_{box}(\delta)$ (d) $T_{box+}(\delta)$

Figure 6.8: A graphical representation of the optimized ellipsoidal and hyperbox uncertainty sets with $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{fc}$ corresponding to Design 1 of the 3-node network.

for the instances with $\beta = 0$. The value of $SF$ corresponding to the feasible center is smaller than that obtained with the analytic center under multivariate Gaussian uncertainty; whereas the converse is true under the truncated multivariate Gaussian. Furthermore, we observe that the behavior of all the indexes tested are in agreement with the trend observed in the stochastic index $SF$. The interpretation of the flexibility index is fundamentally different for the ellipsoidal and hyperbox sets; however, we observe that both follow the same trends in this instance because they have the same set of limiting constraints. We also note that computing each of these indexes required less than 0.2 seconds in contrast to the MC-generated index $SF$ (which required around 5,200 seconds

on average for each case). Also, the hyperbox sets required less computing time than the ellipsoidal sets (this is expected since Problem (6.18) is linear with the hyperbox sets and conic with the ellipsoidal sets).

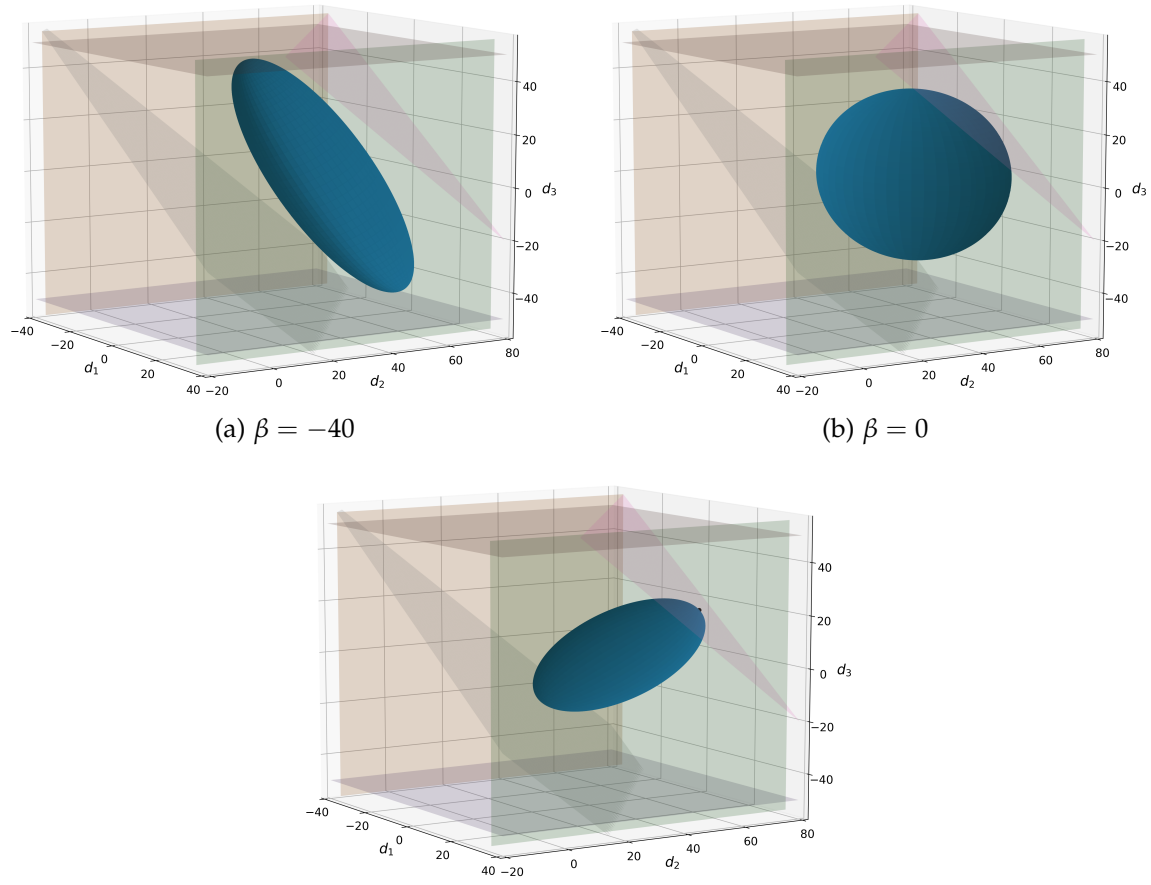
We substituted the node balances into the capacity constraints to obtain a system of inequalities that are expressed solely in terms of $\boldsymbol{\theta}$. Thus, the uncertainty sets for Design 1 can be visualized in three dimensions. Figure 6.8 depicts the uncertainty sets of Table 6.5 with the center $\bar{\boldsymbol{\theta}}_{fc}$. Here, it is clear how intersecting the sets $T_{ellip}(\delta)$ and $T_{box}(\delta)$ with $\mathbb{R}^{n_\theta}_{+}$ creates more exotic regions and this affects the solution. Specifically, the intersected sets are scaled to a greater extent, and they are limited by different constraints (in comparison to their non-intersected counterparts). This again reflects that the shape of $T(\delta)$ has a significant effect on the behavior of the flexibility index. Also, when Figure 6.8a is juxtaposed with Figure 6.9b, it is apparent how the analytic center is more neutrally positioned between the supplier capacity constraints (slanted planes) relative to the feasible center. This observation explains why the flexibility is identical for the intersected and non-intersected sets.

Table 6.6: Results for Design 1 for various covariance values $\beta$.

| Center | $\beta$ | $F_{ellip}$ | $\alpha^*$ (%) | $SF$-MC (%) | Solution Time (s) |
|---|---|---|---|---|---|
| analytic | -40 | 15.31 | 99.84 | 99.99 | 0.18 |
| | 0 | 8.93 | 96.97 | 99.71 | 0.21 |
| | 50 | 4.31 | 77.02 | 96.22 | 0.18 |
| feasible | -40 | 15.31 | 99.84 | 99.99 | 0.14 |
| | 0 | 5.00 | 82.79 | 98.71 | 0.16 |
| | 50 | 2.41 | 50.85 | 93.49 | 0.15 |

We now show $T_{ellip}(\delta)$ can be used to conservatively approximate $SF$ via the corresponding confidence level $\alpha^*$ and how correlation in $\boldsymbol{\theta}$ affects indexes $SF$ and $F$. Specifically, we consider the combination of instances for $\beta \in \{-40, 0, 50\}$ and $\bar{\boldsymbol{\theta}} \in \{\bar{\boldsymbol{\theta}}_{ac}, \bar{\boldsymbol{\theta}}_{fc}\}$. Table 6.6 summarizes the solutions to these instances. Again, we observe that $F_{ellip}$ (and the corresponding confidence levels) mirror the trends observed with $SF$ at a significantly

reduced computational cost. The correlation between the random variables clearly has an impact on $SF$, which $F_{ellip}$ captures as well. On the other hand, index $F_{box}$ does not capture this behavior since it cannot account for parameter correlation. Also, it is apparent that the confidence levels associated with $\beta = -40$ provide a very tight lower bound for $SF$, whereas the one associated with $\beta = 50$ has a much larger gap. This behavior illustrates how the shape of the feasible region and that of the uncertainty set affect the tightness of the bound on $SF$ achieved by $\alpha^*$.




(a) $\beta = -40$

(b) $\beta = 0$

(c) $\beta = 50$

Figure 6.9: Ellipsoidal sets with $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{ac}$ and different covariance values $\beta$ (Design 1 of the three-node network).

Figure 6.9 depicts the uncertainty sets corresponding to the results of Table 6.6. We note how the ellipsoidal set with $\beta = -40$ is able to scale to a greater extent relative to

the set associated with $\beta = 50$ because of its more favorable shape relative to the feasible region. This helps explain why $\alpha^*$ is able to provide the tightest lower bound for $SF$ with $\beta = -40$.

### *Design Comparison*

To compare the three different designs described in Figure 6.7, we consider the results obtained with $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{ac}$ (the analytic center for Design 1) and $\beta = 50$. Table 6.7 summarizes these results. The results indicate that Design 2 (which features decentralized suppliers) is less flexible than Design 1 (which utilizes a centralized supplier scheme). This is surprising, since intuition suggests that decentralization increases network flexibility. Our framework, however, shows that the converse is true in this particular instance. This behavior results from the correlation structure of the random variables. Furthermore, we observe that the arc added in Design 3 does not increase system flexibility (as one might also intuitively expect). As we discuss next, this behavior results from non-obvious interplays in the limiting constraints. This highlights how flexibility analysis can help guide system design.

Table 6.7: A summary of the flexibility index results using $F_{ellip}$ and $F_{box}$ to compare Designs 1, 2, and 3 of the three-node distribution network.

| | $F_{box}$ | $F_{ellip}$ | $\alpha^*$ (%) | $SF$-MC (%) |
|---|---|---|---|---|
| Design 1 | 0.578 | 4.310 | 77.02 | 96.22 |
| Design 2 | 0.578 | 3.612 | 69.35 | 94.32 |
| Design 3 | 0.578 | 4.310 | 77.02 | 96.20 |

### *Limiting Constraint Ranking*

We use Design 1 to illustrate the limiting constraint ranking methodology described in Section 6.6, letting $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{ac}$ and $\beta = 50$ in conjunction with $T_{ellip}(\delta)$. Table 6.8 summarizes the results. We observe that the supplier capacity limits system flexibility the most. This

explains why adding an arc in Design 3 does not increase the flexibility of the network (i.e., the system is constrained by the inability to supply power and not by the inability to transport power). As a result, one can improve flexibility the most by increasing the supplier capacity (instead of adding an arc). Figure 6.10 depicts these results by shading the limiting design components in accordance with their corresponding index $F_{ellip}$. Here, components are colored according to the value of $F_{ellip}$ corresponding to their rank level (a lower value indicates that the component is more limiting). We thus see that supply is the most limiting component, the left arc is the second most limiting component, and the right arc is the third most limiting component.

Table 6.8: A summary of the constraint ranking results corresponding to Design 1 of the three-node distribution network using the set $T_{ellip}(\delta)$ with $\beta = 50$ and $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{ac}$.

| | Active Constraint Multipliers | $F_{ellip}$ | Flexibility Increase (%) |
|---|---|---|---|
| Rank 1 | $\gamma_2^U$, $\gamma_2^L$ | 4.310 | – |
| Rank 2 | $\lambda_{2:1}^U$, $\lambda_{2:1}^L$ | 15.312 | 255.3 |
| Rank 3 | $\lambda_{2:3}^U$, $\lambda_{2:3}^L$ | 20.833 | 383.4 |

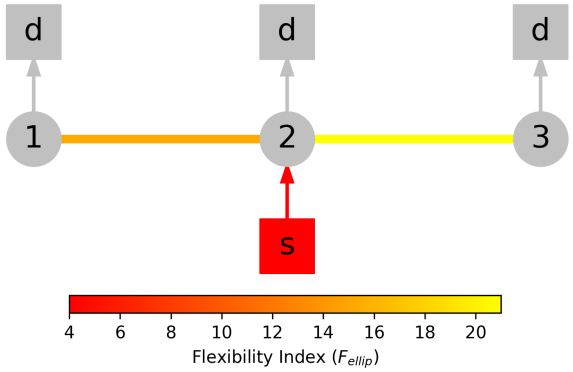
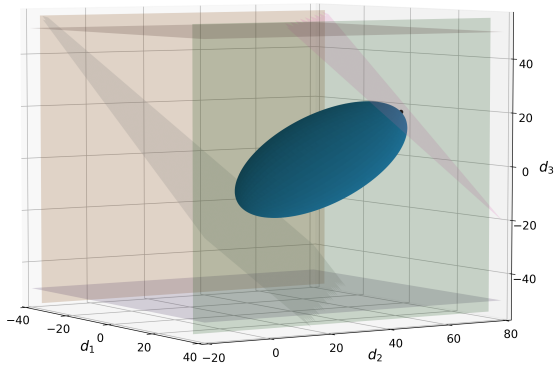


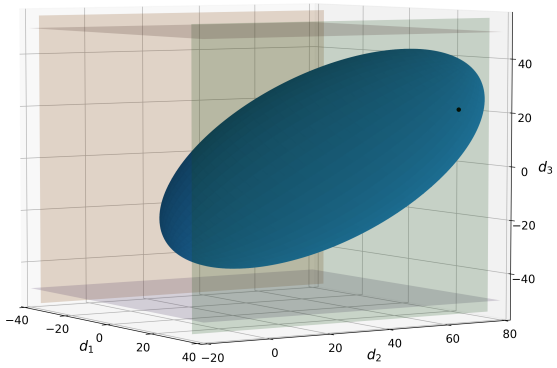

Figure 6.10: Limiting components for three-node network. Colors are proportional to the corresponding indexes $F_{ellip}$ (the smaller the value the most limiting the component is).

Figure 6.11 depicts the solutions corresponding to the results of Table 6.8. We see that the size of $T_{ellip}(\delta)$ significantly increases as limiting constraints are removed. In this study, we found that two constraints simultaneously limit flexibility (corresponding to the maximum and minimum capacities of the limiting component). This indicates that

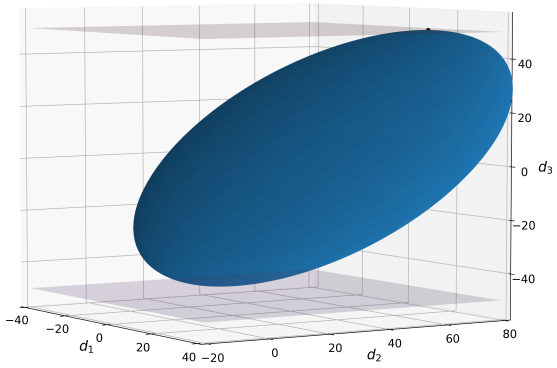
the uncertainty set touches the boundary of the feasible set at two locations.




(a) Rank 1




(b) Rank 2




(c) Rank 3

Figure 6.11: Limiting components for Design 1 of three-node network using $T_{ellip}(\delta)$ with $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}}_{ac}$ and $\beta = 50$.

### 6.7.3 *IEEE-14 Network*

We now consider the IEEE 14-node power network, originally provided in Dabbagchi in [165]. The system data is obtained from `MATPOWER`. This test case does not provide arc capacities, so we enforce $a^C = 100$ for all the arcs. This base design is labeled Design 1 and a schematic of this system is provided in Figure 6.12. Design 2 is obtained by removing the arc that connects node 2 to node 5 and by removing the arc that connects node 9 to node 14. Design 3 uses a centralized power supply scheme, this is accomplished by removing all the suppliers except for the two located on nodes 1 and 2, by increasing

the capacity of each remaining supplier from 332 to 432 and from 140 to 340 respectively, and by increasing the capacity of each arc to 200.

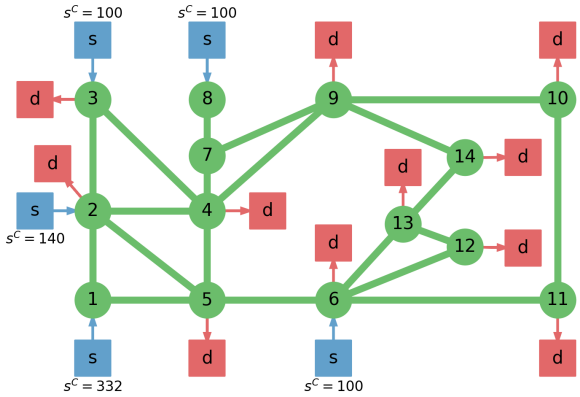



Figure 6.12: Schematic of the IEEE 14-node power system where the values $s^C$ are indicated and all the values $a^C = 100$.

This system is subjected to a total of 10 uncertainty disturbances (the network demands). The demands are assumed to be $\boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$, where the nominal point is defined $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{fc} = (87.3, 50.0, 25.0, 28.8, 50.0, 25.0, 0, 0, 0, 0, 0)$ and $V_{\boldsymbol{\theta}} = 1200\mathbb{I}$. We select the sets $T_{box+}(\delta)$ and $T_{ellip+}(\delta)$ (the demands are nonnegative). Each element of the hyperbox deviations $\Delta\boldsymbol{\theta}^-, \Delta\boldsymbol{\theta}^+$ is set to $3\sigma_i = 103.92$ (corresponding to $\bar{\boldsymbol{\theta}}_i \pm 3\sigma_i$ confidence bounds). Both of these sets are used to determine the flexibility index from Problem (6.18) for each design. The stochastic flexibility index $SF$ is computed using 1,000,000 MC samples.

Table 6.9 summarizes these design comparison results. Index $SF$ shows that the flexibility of Design 1 is worsened with the removal of the two arcs, whereas the centralized supply modification improves flexibility. An observation is that $F_{ellip+}$ mirrors the behavior $SF$ at a much lower computational cost. Specifically, $F_{ellip+}$ required less than 20 seconds to compute while computing $SF$ using MC sampling required 13,030 seconds (3.6 hours). Index $F_{box+}$ is not able to mirror the same behavior (it does not indicate that Design 2 is less flexible than Design 1 as would be expected). This analysis also shows that the higher capacity of the centralized supply scheme (Design 3) increases system flexibility relative to Design 1, which again is counter-intuitive.

Table 6.9: Flexibility index results for various designs of IEEE-14 system.

| | $F_{box+}$ | $F_{ellip+}$ | $SF$-MC (%) |
|---|---|---|---|
| Design 1 | 0.327 | 10.594 | 97.84 |
| Design 2 | 0.327 | 8.333 | 94.89 |
| Design 3 | 0.404 | 12.244 | 99.91 |

The limiting constraints of Design 1 are identified and ranked using $T_{ellip+}(\delta)$. The ranking results are shown in Table 6.10. The capacity constraints corresponding to four of the five suppliers along with three arc capacity constraints are identified as the most limiting ones. Again, this indicates that the uncertainty set touches the boundary of the feasible set at multiple points. We see that the next set of constraints only have an index that is 15.6% larger meaning they are likely to also limit the system flexibility to a certain extent. Subsequent sets have significantly increased flexibility indexes (up to 214% larger) and therefore have little effect on flexibility.

Table 6.10: Ranking of limiting constraints for IEEE-14 system.

| | Active Constraints | $F_{ellip+}$ | Increase (%) | Solution Time (s) |
|---|---|---|---|---|
| Rank 1 | $\lambda^U_{1:2}$, $\lambda^U_{1:5}$, $\lambda^U_{6:11}$, $\gamma^U_2$, $\gamma^U_3$, $\gamma^U_6$, $\gamma^U_8$ | 10.594 | – | 16.61 |
| Rank 2 | $\lambda^L_{2:3}$, $\lambda^L_{2:5}$, $\lambda^U_{12:13}$, $\gamma^L_1$, $\gamma^L_2$, $\gamma^L_3$, $\gamma^L_6$, $\gamma^L_8$ | 12.243 | 15.6 | 3.76 |
| Rank 3 | $\lambda^U_{2:3}$, $\lambda^U_{2:4}$, $\lambda^U_{3:4}$, $\lambda^U_{6:12}$, $\lambda^U_{6:13}$, $\lambda^U_{9:14}$ | 25.000 | 136.0 | 2.72 |
| Rank 4 | $\lambda^U_{2:5}$, $\lambda^L_{5:6}$, $\lambda^U_{9:10}$, $\lambda^L_{9:14}$, $\lambda^L_{10:11}$, $\gamma^U_1$ | 33.333 | 214.6 | 0.14 |

Figure 6.13 highlights the limiting components of Design 1. The limiting constraints corresponding to the active capacity constraints at each rank level are shaded according to the value of $F_{ellip+}$ (the lower the value, the most limiting the component is). We can see that the suppliers along with arcs attached to such suppliers limit flexibility the most. The rest of the components follow non-intuitive patterns, reflecting complex dependencies due to the network topology.

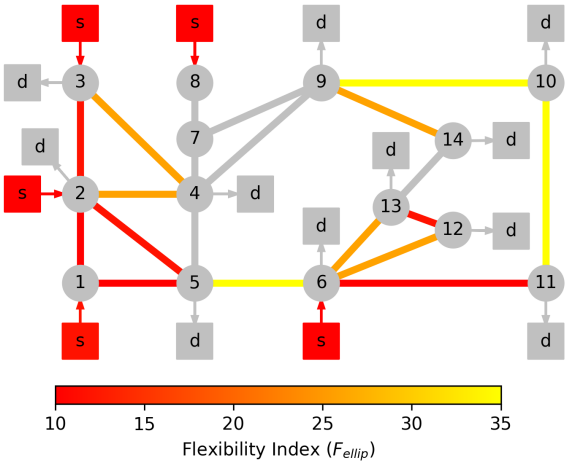



Figure 6.13: Limiting components for IEEE-14 power system. The colors are proportional to the corresponding indexes $F_{ellip+}$ (the smaller the value the most limiting the component is).

### 6.7.4 *141-Node Network*

Finally, we consider a 141-node power distribution network which corresponds to an urban area in Caracas, Venezuela and was originally developed in [166]. The network data is extracted from `MATPOWER`, but again no arc capacities are provided (we assume $a^C = 100$). Figure 6.14 provides a schematic of the network.

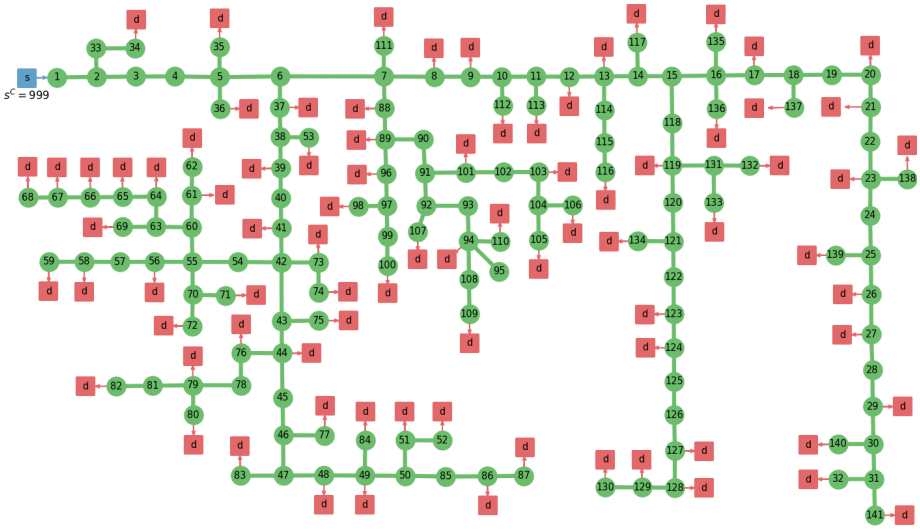


Figure 6.14: Schematic of the 141-node power distribution network.

This system is subjected to 84 uncertain disturbances (corresponding to the demands). The demands are assumed to be $\boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$, where $\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{fc}$ and $V_{\boldsymbol{\theta}} = 100\mathbb{I}$. We use $T_{ellip}(\delta)$ in combination with Problem (6.18) to rank all the inequality constraints. This problem was solved 270 times while iteratively removing the active constraints. We found that several solutions have the same value of $F_{ellip}$. Thus, the active constraints corresponding to solutions with equivalent $F_{ellip}$ values are combined and assigned to the same ranks. A total of 44 rank levels were identified, the ranking data corresponding to the first 30 levels is provided in Appendix A.

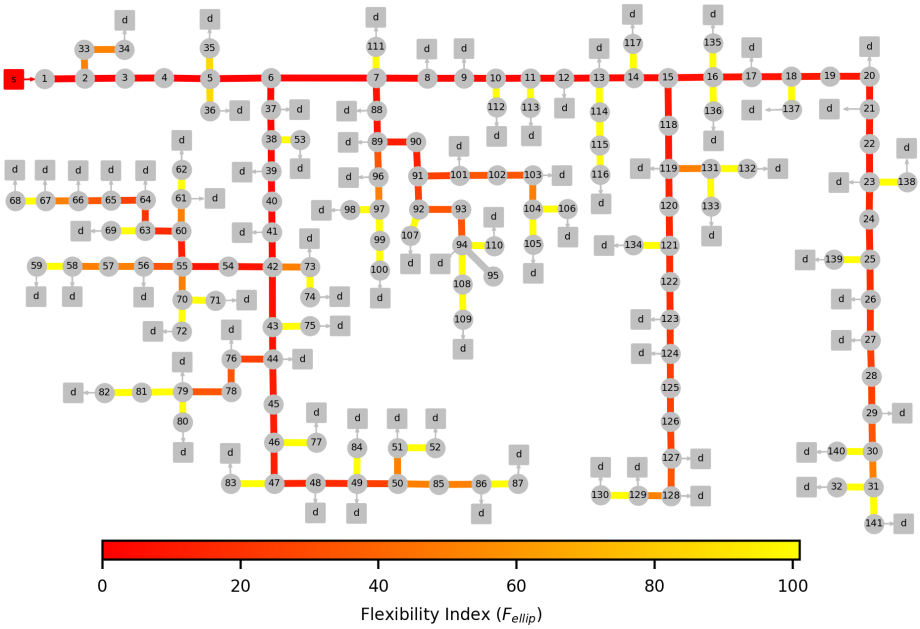



Figure 6.15: Limiting components for the 141-node power system. The colors are proportional to the corresponding indexes $F_{ellip}$ (the smaller the value the most limiting the component is).

Figure 6.15 shows the limiting components corresponding to the 44 rank levels. We observe that the supplier is the most limiting component along with the arcs that form the spanning tree of the network (the path that connects all nodes in the network). The arcs become less relevant as they branch out towards the network boundaries. We thus see that the flexibility framework reveals topological limitations of the network. We also highlight that the flexibility index can be computed for this large system in 20 seconds.

In contrast, given the large number of random parameters, computing the $SF$ index using Monte Carlo sampling is impractical.

# Chapter 7

SCALABLE DESIGN OF FLEXIBLE SYSTEMS

The content of this chapter is published in [1].

## 7.1 Introduction

The flexibility index problem has given rise to the most common class of design problems in flexibility analysis where design variables $\mathbf{d}$ are selected to minimize cost and either ensure the feasibility at a fixed $F$ index or maximize the flexibility index problem [146, 160, 161, 167]. However, studies reported in the literature find the latter problem to be largely intractable due to its multi-objective complexity [46, 168]. Thus, the design problem has traditionally been formulated such that it enforces the feasibility of an uncertainty set $T(\hat{F})$ fixed at a particular flexibility index $\hat{F}$. This optimal design problem is formalized

$$\begin{aligned} \underset{\mathbf{d},\boldsymbol{\theta}}{\text{argmin}} \quad & c(\mathbf{d}) \\ \text{s.t.} \quad & \max_{\boldsymbol{\theta}\in T(\hat{F})} \psi(\mathbf{d},\boldsymbol{\theta}) \leq 0 \\ & \mathbf{d} \in D \end{aligned} \tag{7.1}$$

where $c(\mathbf{d})$ is the cost function and $D$ denotes the set of all feasible $\mathbf{d}$ values. This is a straightforward approach for selecting $\mathbf{d}$, however the use of uncertainty sets conser-

vatively models the flexibility of a given system design [52]. This means that solutions to the design problem with the use of uncertainty sets might be overly conservative and thus choose a greater cost than is necessary to achieve a certain degree of flexibility.

Straub and Grossmann proposed a probabilistic flexibility measure that they called the stochastic flexibility ($SF$) index. This index quantifies the probability of finding feasible operation [47]. Here, the uncertain parameters $\boldsymbol{\theta}$ are modeled as random variables with associated probability density function $p : \mathbb{R}^{n_\theta} \to \mathbb{R}$. The stochastic flexibility index is defined:

$$SF(\mathbf{d}) := \int_{\boldsymbol{\theta} \in \Theta(\mathbf{d})} p(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{7.2}$$

where $\Theta(\mathbf{d}) := \{\boldsymbol{\theta} : \psi(\mathbf{d}, \boldsymbol{\theta}) \leq 0\}$ is the feasible set (projected onto the space of the uncertain parameters). Pistikopoulos and Mazzuchi proposed a similar definition in [149]. In Chapter 6, we noted that the SF index can also be expressed as:

$$\begin{aligned} SF(\mathbf{d}) &= \mathbb{P}\left(\psi(\mathbf{d}, \boldsymbol{\theta}) \leq 0\right) \\ &= \mathbb{P}\left(\exists\, \mathbf{z} : f_j(\mathbf{d}, \mathbf{z}, \boldsymbol{\theta}) \leq 0,\ h_i(\mathbf{d}, \mathbf{z}, \boldsymbol{\theta}) = 0,\ j \in J, i \in I\right). \end{aligned} \tag{7.3}$$

The $SF$ index can be computed rigorously by evaluating (7.2) or (7.3) via Monte Carlo (MC) sampling. Here, the feasibility of each realization of $\boldsymbol{\theta}$ is assessed using the feasibility function $\psi(\mathbf{d}, \boldsymbol{\theta})$. Such an approach converges asymptotically but can require a large number of samples [48, 49]. This is a common drawback of sample average approximation approaches for stochastic programming but is especially important in the context of flexibility analysis because limiting behavior is often found near the boundary of $\Theta(\mathbf{d})$. Interestingly, high-dimensional multivariate Gaussian random variables have been shown to contain most probability mass in a thin ellipsoidal shell from which the MC samples are generated [169]. Consequently, in some applications a moderate number of samples might be needed. Important sampling and sparse grid techniques have also been proposed recently to cover the uncertainty space more effectively and with this reduce the number of samples [88, 170].

The $SF$ index can be used as a metric to guide the design of flexible systems [11]. Typically, one aims to find a design $\mathbf{d}$ that minimizes a design cost while ensuring that the system remains feasible with a given probability level $\alpha \in [0, 1]$ (typically close to one):

$$\begin{aligned} &\min_{\mathbf{d}\in\mathcal{D}} c(\mathbf{d}) \\ &\text{s.t. } SF(\mathbf{d}) \geq \alpha. \end{aligned} \tag{7.4}$$

Here, $c : \mathbb{R}^{n_d} \to \mathbb{R}$ is the cost design function and $\mathcal{D} \subseteq \mathbb{R}^{n_d}$ is the design space. Note that this is an optimization problem with *joint chance constraints* that is computationally challenging to solve. In particular, the joint chance constraint often needs to be reformulated by using binary variables [57]. In this work, we provide a more scalable approach to address the design problem. The approach relies on the observation that the above problem provides a Pareto solution (for a specific value of $\alpha$) for the conflict resolution (multi-objective) problem:

$$\min_{\mathbf{d}\in\mathcal{D}} \left\{ c(\mathbf{d}), -SF(\mathbf{d}) \right\}. \tag{7.5}$$

We will demonstrate that we can recover the Pareto set for this problem to high accuracy (which includes solutions of the design problem (7.4)) by solving a *continuous* formulation. This thus provides a scalable approach to solve large-scale design problems.

## 7.2 Stochastic Flexibility Design

This section outlines standard approaches to solve the conflict resolution problem using sample average approximations and mixed-integer reformulations. Here, we also discuss the continuous formulation used to approximate the solution of the design problem.

### 7.2.1 *Sample Average Approximation of the Stochastic Flexibility Index*

The $SF$ index (shown in (7.2) and (7.3)) involves a high-dimensional integral and a projection into the feasible space of the system. This index can be approximated via Monte Carlo (MC) sampling as:

$$SF(\mathbf{d}) = \mathbb{E}[\mathbb{1}_{\boldsymbol{\theta}\in\Theta(\mathbf{d})}] \approx SF_K(\mathbf{d}) := \frac{1}{|K|}\sum_{k\in K} \mathbb{1}_{\boldsymbol{\theta}^k\in\Theta(\mathbf{d})} \tag{7.6}$$

where $K$ is the number of MC samples, $\boldsymbol{\theta}^k$ is a random sample drawn from $p(\boldsymbol{\theta})$, and $\mathbb{1}_{\boldsymbol{\theta}^k\in\Theta(\mathbf{d})}$ is the indicator function. This function takes a value of one if $\boldsymbol{\theta}^k$ is feasible (i.e., $\psi(\mathbf{d},\boldsymbol{\theta}^k) \leq 0$) or takes a value of zero otherwise. Note that $\mathbb{1}_{\boldsymbol{\theta}^k\in\Theta(\mathbf{d})} = \mathbb{1}_{\psi(\mathbf{d},\boldsymbol{\theta}^k)\leq 0}$. From the law of large numbers we have that this approximation becomes exact as $|K| \to \infty$ [49].

For a given sample $\boldsymbol{\theta}^k$, Problem (6.3) can be written as:

$$\begin{aligned} \psi(\mathbf{d},\boldsymbol{\theta}^k) = \quad & \min_{\mathbf{z}^k, y^k} \quad y^k U \\ & \text{s.t.} \quad f_j(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) \leq y^k U, \quad j\in J \\ & \qquad\; h_i(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) = 0, \qquad i\in I \\ & \qquad\; y^k \in \{0,1\} \end{aligned} \tag{7.7}$$

where $U \in \mathbb{R}_+$ is a sufficiently large constant. Note that the continuous variable for a particular sample $u^k$ is equivalently replaced with $y^k U$. The binary variable $y^k$ takes a value of zero if the realization $\boldsymbol{\theta}^k$ is feasible (corresponding to $\psi(\mathbf{d},\boldsymbol{\theta}^k) = 0$) or takes a value of one if it is infeasible (corresponding to $\psi(\mathbf{d},\boldsymbol{\theta}^k) > 0$).

By combining all realizations $k \in K$, we obtain the aggregated problem:

$$\begin{aligned} \min_{\mathbf{z}^k, y^k} \quad & \frac{1}{|K|} \sum_{k \in K} y^k U \\ \text{s.t.} \quad & f_j(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) \leq y^k U, \quad j \in J,\ k \in K \\ & h_i(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) = 0, \qquad i \in I,\ k \in K \\ & y^k \in \{0,1\}, \qquad\qquad k \in K. \end{aligned} \tag{7.8}$$

Since $\mathbf{d}$ is fixed, this problem is fully decoupled in the set $K$ (i.e., solving Problem (7.8) is equivalent to solving Problem (6.3) for each $k \in K$) and the optimal values of the binary variables $y^k$ satisfy $SF_K(\mathbf{d}) = \frac{1}{|K|} \sum_{k \in K} (1 - y^k)$. Consequently, (7.8) can be used to compute the sample average approximation of the $SF$ index and the approximation becomes exact as $|K| \to \infty$.

The constant $U$ in the objective can be eliminated without affecting the solution. Moreover, we obtain an equivalent problem by maximizing $SF_K(\mathbf{d}) = \frac{1}{|K|} \sum_{k \in K} (1 - y^k)$ directly. These modifications give the problem:

$$\begin{aligned} \max_{\mathbf{z}^k, y^k} \quad & \frac{1}{|K|} \sum_{k \in K} (1 - y^k) \\ \text{s.t.} \quad & f_j(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) \leq y^k U, \quad j \in J,\ k \in K \\ & h_i(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) = 0, \qquad i \in I,\ k \in K \\ & y^k \in \{0,1\}, \qquad\qquad k \in K. \end{aligned} \tag{7.9}$$

An important observation is that a continuous relaxation of problem (7.8) is given by:

$$\begin{aligned} \min_{\mathbf{z}^k, u^k} \quad & \frac{1}{|K|} \sum_{k \in K} u^k \\ \text{s.t.} \quad & f_j(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) \leq u^k, \quad j \in J,\ k \in K \\ & h_i(\mathbf{d}, \mathbf{z}^k, \boldsymbol{\theta}^k) = 0, \qquad i \in I,\ k \in K \\ & u^k \geq 0, \qquad\qquad k \in K. \end{aligned} \tag{7.10}$$

This provides a relaxation because $u^k$ can be modeled as $u^k = y^k U$ with $0 \leq y^k \leq 1$ and sufficiently large $U \in \mathbb{R}_+$. In summary, the continuous problem (7.10) is a *relaxation* of the mixed-integer problem (7.8) (or, equivalently, (7.9)). We also note that $u_k = \psi(\mathbf{d}, \boldsymbol{\theta}^k)$ is a measure of infeasibility for sample $\boldsymbol{\theta}^k$ and thus the continuous problem (7.10) minimizes the *mean (expected) infeasibility*. In other words, the expected infeasibility is a surrogate measure of flexibility.

### 7.2.2 *Mixed-Integer Optimal Design Problem*

We now proceed to find a design $\mathbf{d}$ that minimizes the design cost and that maximizes the stochastic flexibility index. A sample average approximation of this problem is given by:

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D},\mathbf{z}^k,y^k} \quad & \left\{ c(\mathbf{d}), -\frac{1}{|K|}\sum_{k\in K}(1-y^k) \right\} \\ \text{s.t.} \quad & f_j(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) \leq y^k U, && j\in J,\ \ k\in K \\ & h_i(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) = 0, && i\in I,\ \ k\in K \\ & y^k \in \{0,1\}, && k\in K \end{aligned} \tag{7.11}$$

A Pareto solution of this problem can be obtained by solving an $\epsilon$-constrained problem of the form:

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D},\mathbf{z}^k,y^k} \quad & c(\mathbf{d}) \\ \text{s.t.} \quad & f_j(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) \leq y^k U, && j\in J,\ \ k\in K \\ & h_i(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) = 0, && i\in I,\ \ k\in K \\ & \frac{1}{|K|}\sum_{k\in K}(1-y^k) \geq \epsilon_s \\ & y^k \in \{0,1\} && k\in K \end{aligned} \tag{7.12}$$

where $\epsilon_s$ is the $\epsilon$-parameter. This formulation is precisely a sample average approximation of the joint chance-constrained problem (7.4). This problem is computationally expensive to solve due to its mixed-integer nature and due to the large number of potential MC

samples.

### 7.2.3 *Continuous Optimal Design Problem*

A key observation is that Pareto solutions for the conflict resolution problem can also be obtained by solving the mixed-integer problem:

$$\begin{aligned} \max_{\mathbf{d}\in\mathcal{D},\mathbf{z}^k,y^k} \quad & \frac{1}{|K|}\sum_{k\in K}(1-y^k) \\ \text{s.t.} \quad & f_j(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) \leq y^k U, \quad j\in J, \quad k\in K \\ & h_i(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) = 0, \qquad i\in I, \quad k\in K \\ & c(\mathbf{d}) \leq \epsilon_c \\ & y^k \in \{0,1\}, \qquad k\in K. \end{aligned} \tag{7.13}$$

Note that this is simply the counterpart to (7.12). We denote the solution of this problem as $\mathbf{d}^*, y^{k^*}$ and we have that $c(\mathbf{d}^*) = \epsilon_f$ holds at the solution (because the objective is conflicting) and the Pareto pair is given by $(c(\mathbf{d}^*), SF_K) = (\epsilon_f, SF_K)$, where $SF_K = \frac{1}{|K|}\sum_{k\in K}(1-y^{k^*})$.

A continuous relaxation of this problem is given by:

$$\begin{aligned} \max_{\mathbf{d}\in\mathcal{D},\mathbf{z}^k,y^k} \quad & \frac{1}{|K|}\sum_{k\in K}(1-y^k) \\ \text{s.t.} \quad & f_j(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) \leq y^k U, \quad j\in J, \quad k\in K \\ & h_i(\mathbf{d},\mathbf{z}^k,\boldsymbol{\theta}^k) = 0, \qquad i\in I, \quad k\in K \\ & c(\mathbf{d}) \leq \epsilon_c \\ & 0 \leq y^k \leq 1, \qquad k\in K. \end{aligned} \tag{7.14}$$

The solution of this problem delivers a continuous solution $\bar{y}^k$ from which we recover integer values as $\bar{y}^k \leftarrow \mathbb{1}_{\bar{y}^k}$ (i.e., we recover $\bar{y}^k = 0$ if $k$ is feasible or $\bar{y}^k = 1$ otherwise). With this, we compute an estimate of the SF index as $\bar{SF}_K = \frac{1}{|K|}\sum_{k\in K}(1-\mathbb{1}_{\bar{y}^k})$. We solve

(7.14) again (by adjusting $\mathbf{d} \in \mathcal{D}$ and $\mathbf{z}^k$) by fixing the binary variables to ensure that the selection of *active* constraints provides a feasible solution. Note that the objective function $\bar{SF}_K$ is fixed since the binaries are fixed. The solution of this feasibility problem is given by $\bar{\mathbf{d}}, \bar{\mathbf{z}}^k$. We highlight that problem (7.14) can also be solved by using the continuous variables $u^k$. We will see that this approach delivers Pareto pairs $(c(\bar{\mathbf{d}}), \bar{SF}_K)$ that closely match the optimal Pareto pairs $(c(\mathbf{d}^*), SF_K)$ of the mixed-integer formulation. We also note an approximation of comparable quality is achieved when the counterpart problem of (7.14) is solved.

A precise theoretical justification for this continuous relaxation providing such a high quality approximation is the subject of ongoing research and is beyond the scope of this work. We hypothesize that the high quality of the solutions is due to the degenerate nature of the SF index. Specifically, different combinations of active constraints give the same or a very similar SF index. This behavior has been recently reported in [171], where the authors note that chance constraints give rise to a wide range of local minima with similar values. This behavior becomes more evident when one moves the SF index into the objective (as opposed to imposing it in the constraints). This is precisely why it is important to think about the design problem as a conflict resolution problem. Recent work has also provided evidence that continuous (Lagrangian) relaxations of chance-constrained problems provide high quality approximations [172].

## 7.3 Case Studies

We analyze the behavior of the proposed formulations by applying them to distribution networks. We consider the same simple three-node, IEEE-14, and 141-node networks featured in Chapter 6. All formulations are implemented in `JuMP 0.18.5` [163] and are solved using `CPLEX 12.6.3` for mixed-integer problems and `Gurobi 7.5.1` for linear problems on a dual Intel® Xeon® ES-2698 v3 machine running at 2.30 GHz with 64 hardware threads and 198 GB of RAM. All results can be reproduced using the scripts provided in

https://github.com/zavalab/JuliaBox/tree/master/FlexDesign.

### 7.3.1 *Case Study Models*

Each network is modeled following the model presented in Section 2.5.2 by performing balances at each node $n \in \mathcal{C}$ and enforcing capacity constraints on the arcs $a_l, l \in \mathcal{A}$, and on the suppliers $s_b, b \in \mathcal{S}$. The demands $r_m, m \in \mathcal{R}$, are assumed to be the uncertain parameters which are given by the MC samples. The deterministic network model is given by:

$$\sum_{l \in \mathcal{A}_n^{rec}} a_l - \sum_{l \in \mathcal{A}_n^{snd}} a_l + \sum_{b \in \mathcal{S}_n} s_b - \sum_{m \in \mathcal{R}_n} r_m = 0, \quad n \in \mathcal{C} \tag{7.15a}$$

$$-a_l^C - d_l^a \leq a_l \leq a_l^C + d_l^a, \quad l \in \mathcal{A} \tag{7.15b}$$

$$0 \leq s_b \leq s_b^C + d_b^s, \quad b \in \mathcal{S} \tag{7.15c}$$

where $\mathcal{A}_n^{rec}$ denotes the set of receiving arcs at node $n$, $\mathcal{A}_n^{snd}$ denotes the set of sending arcs at $n$, $\mathcal{S}_n$ denotes the set of suppliers at $n$, $\mathcal{R}_n$ denotes the set of demands at $n$, $a_l^C$ are the arc capacities, $d_l^a \geq 0$ are design variables that increase arc capacity, $s_b^C$ are the supplier capacities, and $d_b^s \geq 0$ are design variables that increase supplier capacity.

The three-node distribution network features a centralized supplier configuration. Figure 7.1 details this network and provides the arc and supplier capacities. The network is subjected to multivariate Gaussian demands $\boldsymbol{\theta} = (r_1, r_2, r_3) \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$. The mean $\bar{\boldsymbol{\theta}}$ is taken to be $\bar{\boldsymbol{\theta}} = (0.0, 60.0, 10.0)$, and the covariance matrix $V_{\boldsymbol{\theta}}$ is assumed to be:

$$V_{\boldsymbol{\theta}} = \begin{bmatrix} 80 & 0 & 0 \\ 0 & 80 & 0 \\ 0 & 0 & 120 \end{bmatrix}. \tag{7.16}$$

We also set the parameter $U = 10000$.

The IEEE 14-node network was originally provided in Dabbagchi in [165]. The system data is obtained from MATPOWER. This test case does not provide arc capacities so we

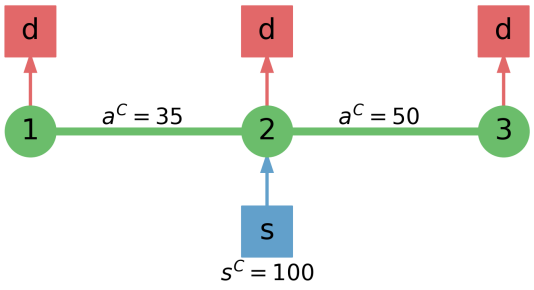



Figure 7.1: The three-node distribution network

enforce $a^C = 100$ for all the arcs. A schematic of this system is provided in Figure 7.2. This system is subjected to a total of 11 uncertain parameters (the network demands). The demands are assumed to be $\boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$, where $\bar{\boldsymbol{\theta}} = (87.3, 50.0, 25.0, 28.8, 50.0, 25.0, 0, 0, 0, 0, 0)$ and $V_{\boldsymbol{\theta}}$ is symmetric matrix with $(V_{\boldsymbol{\theta}})_{ii} = 1200$ and $(V_{\boldsymbol{\theta}})_{ij} = 240$, $\forall i \neq j$. Also, we set $U = 10000$.

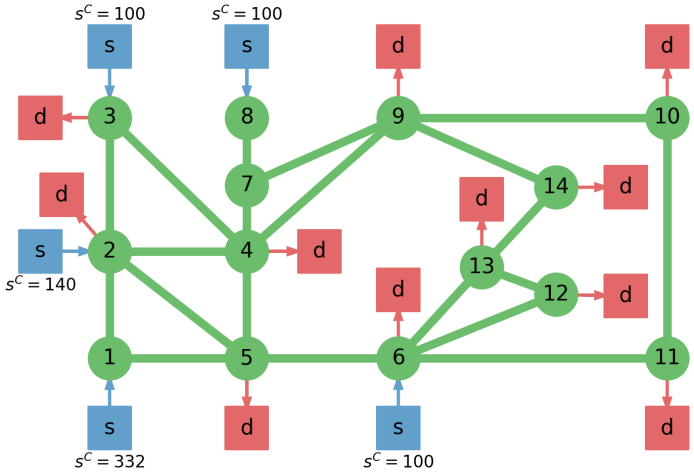



Figure 7.2: Schematic of the IEEE 14-node power system where the values $s^C$ are indicated and all the values $a^C = 100$.

The 141-node power distribution network corresponds to an urban area in Caracas, Venezuela and was originally developed in [166]. The network data is extracted from `MATPOWER`, but again no arc capacities are provided (we assume $a^C = 100$). Figure 7.3 provides a schematic of the network. This system is subjected to 84 uncertain disturbances corresponding to the demands. The demands are assumed to be $\boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}, V_{\boldsymbol{\theta}})$, where

$\bar{\boldsymbol{\theta}} = \bar{\boldsymbol{\theta}}_{fc}$ and $V_{\boldsymbol{\theta}} = 100\mathbb{I}$. The point $\boldsymbol{\theta}_{fc}$ is the feasible center and is the instance of $\boldsymbol{\theta} \in \Theta(\mathbf{d})$ that minimizes the feasibility function $\psi(\mathbf{d}, \boldsymbol{\theta})$ [144]. We set the upper bounding constant to $U = 10000$.

For each network, the cost function $c(\mathbf{d})$ is defined to be linear of the form:

$$c(\mathbf{d}^s, \mathbf{d}^a) = \frac{1}{\sqrt{n_d}} \left( \sum_{b \in \mathcal{S}} d_b^s + \sum_{l \in \mathcal{A}} d_l^a \right). \tag{7.17}$$

Here the unit cost for each design variable is taken to be $1/\sqrt{n_d}$, where $n_d = |\mathcal{S}| + |\mathcal{A}|$. Furthermore, a relatively large value of $U$ is selected in the above problems to ensure a sufficient amount of slack is provided for the inequality constraints. This is all done for convenience in conducting the analysis below.

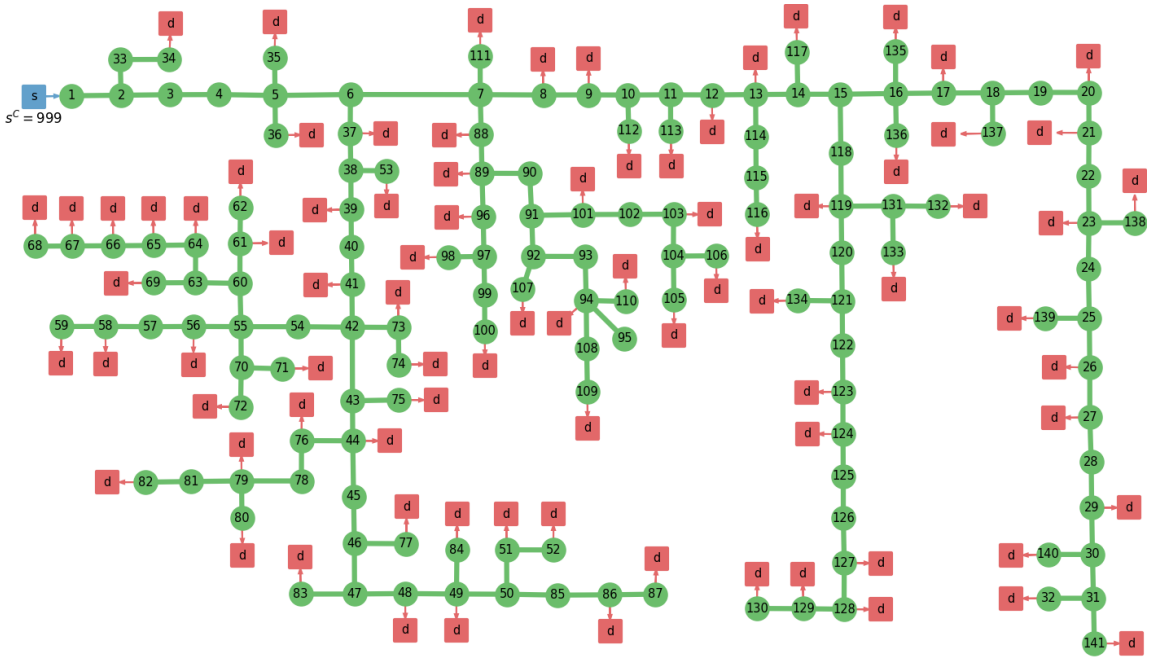



Figure 7.3: Schematic of the 141-node power distribution network.

### 7.3.2 *Mixed-Integer Formulation*

Combining (7.13) with the model equations in (7.15), we obtain:

$$
\begin{aligned}
\max_{d_b^s, d_l^a, a_l^k, s_b^k, y^k} \quad & \frac{1}{|K|} \sum_{k \in K} (1 - y^k) \\
\text{s.t.} \quad & -a_l^C - d_l^a - a_l^k \leq y^k U, && l \in \mathcal{A}, \;\; k \in K \\
& -a_l^C - d_l^a + a_l^k \leq y^k U, && l \in \mathcal{A}, \;\; k \in K \\
& -s_b^k \leq y^k U, && b \in \mathcal{S}, \;\; k \in K \\
& -s_b^C - d_b^s + s_b^k \leq y^k U, && b \in \mathcal{S}, \;\; k \in K \\
& \sum_{l \in \mathcal{A}_n^{rec}} a_l^k - \sum_{l \in \mathcal{A}_n^{snd}} a_l^k + \sum_{b \in \mathcal{S}_n} s_b^k - \sum_{m \in \mathcal{R}_n} r_m^k = 0, && n \in \mathcal{C}, \;\; k \in K \\
& \frac{1}{\sqrt{n_d}} \left( \sum_{b \in \mathcal{S}} d_b^s + \sum_{l \in \mathcal{A}} d_l^a \right) \leq \epsilon_c \\
& y^k \in \{0, 1\} && k \in K \\
& d_b^s \geq 0, \;\; d_l^a \geq 0, && b \in \mathcal{S}, \;\; l \in \mathcal{A}.
\end{aligned}
\tag{7.18}
$$

We extract elements of the Pareto set by varying the cost threshold $\epsilon_c$. This analysis is done using the three-node network with 1,000 MC samples that are generated from the underlying distribution. The Pareto set is obtained from Problem (7.18) for 43 values of $\epsilon_c$ set from 0 to 10.5 in 0.25 increments. A subset of the numerical results is presented in Table 7.1 and the complete set of results is given in Tables A.4 and A.5 in the Appendix. The minimum possible $SF_K$ index associated with the base case is 96.7%. Initially, as we increase $\epsilon_c$, we are able to increase the $SF_K$ index at a relatively small cost. For instance, a 2.1% improvement of the $SF_K$ index only incurs a design cost of 2.75 (1.31 per % increase). In contrast, the final 0.4% increase in the $SF_K$ index, making the network perfectly flexible relative to the 1,000 sampled instances, incurs a 4.75 increase in design cost (11.88 per % increase). This trend can be visualized in Figure 7.4.

We substitute the node balances into the capacity constraints to obtain a system of in-

Table 7.1: A reduced subset of results obtained for the 3-node network with Problem (7.18) using 1,000 MC samples.

| $\epsilon_c$ | Design Cost | $SF_K$ (%) | Solution Time (s) |
|---|---|---|---|
| 0 | 0 | 96.7 | 0.0155 |
| 1.25 | 1.25 | 97.8 | 0.0190 |
| 2.75 | 2.75 | 98.8 | 0.0175 |
| 4.25 | 4.25 | 99.1 | 0.0200 |
| 5.75 | 5.75 | 99.6 | 0.0193 |
| 7.25 | 7.25 | 99.7 | 0.0207 |
| 8.75 | 8.75 | 99.8 | 0.0192 |
| 10.5 | 10.5 | 100 | 0.0195 |

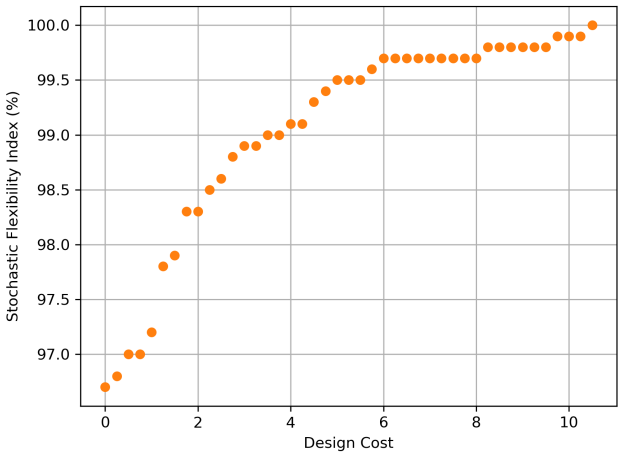
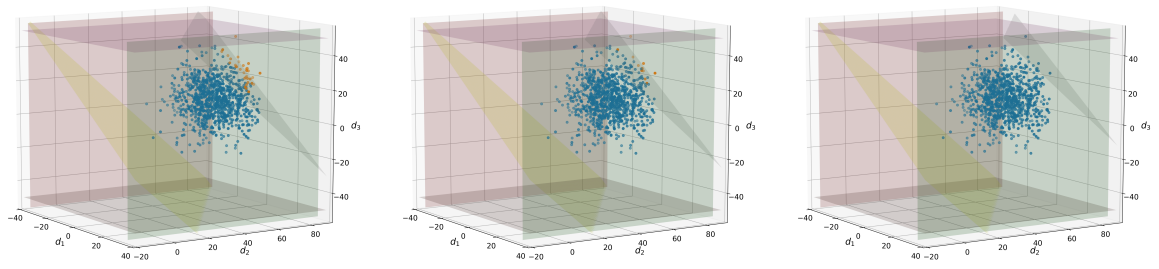



Figure 7.4: The Pareto set for the three-node network obtained with the mixed-integer optimal design formulation.

equalities that are expressed solely in terms of $\boldsymbol{\theta}$. Thus, the three-node network solutions can be visualized in three dimensions. Figure 7.5 shows three solutions corresponding to $\epsilon_c = \{0, 2.75, 10.5\}$. Here, we observe how the diagonal plane (on the right side of the figures), which corresponds the supplier capacity constraint, is shifted as the value of $\epsilon_c$ is increased to minimize the number of infeasible instances and thus maximize the $SF_K$ index.

The same analysis is done using the IEEE-14 network with 2,000 MC samples. This



(a) $\epsilon_c = 0$ (b) $\epsilon_c = 2.75$ (c) $\epsilon_c = 10.5$

Figure 7.5: The optimized solutions to Problem (7.18) for the three-node network, showing how the design constraints are shifted to minimize the number of infeasible (orange) instances as the value of $\epsilon_c$ is increased.

creates a MILP with 50,025 continuous variables, 2,000 binary variables, and 128,001 constraints. The Pareto set is created by varying the value of $\epsilon_c$ from 0 to 65 in increments of 2.5. All the corresponding numerical results are given in Table A.6 in the appendix. Figure 7.6 shows the Pareto set corresponding to these results. Here, the minimum $SF_K$ index associated with the base case is 88.10% and the maximum $SF_K$ index is 91.50%. This means that, regardless of how much the line and supplier capacities are increased, we can only improve $SF_K$ by 3.4% (relative to this MC sample set). This behavior occurs because in certain sampled instances a number of demands are sufficiently negative such that this is a surplus in the system that cannot be rectified since there are no sinks in the network. Such behavior is not readily obvious and thus highlights the utility of this optimal design framework in understanding how to promote the flexibility of a system and in determining to what extent its flexibility can be improved. We also observe that the average solution time for the MILP problems is 40.47s.

Problem (7.18) is also applied to the 141-node power distribution network using 10,000 MC samples and is evaluated for each value of $\epsilon_c$ from 0 to 300 in increments of 10. Each MILP contains 1,410,141 continuous variables, 10,000 binary variables, and 4,230,001 constraints. A time limit of 3,600s is also imposed. All 30 results are summarized in Table A.7 in the Appendix and Figure 7.7 shows these results. Here, a much more appreciable range of $SF_K$ index values is obtained where the minimum $SF$ index of the system is

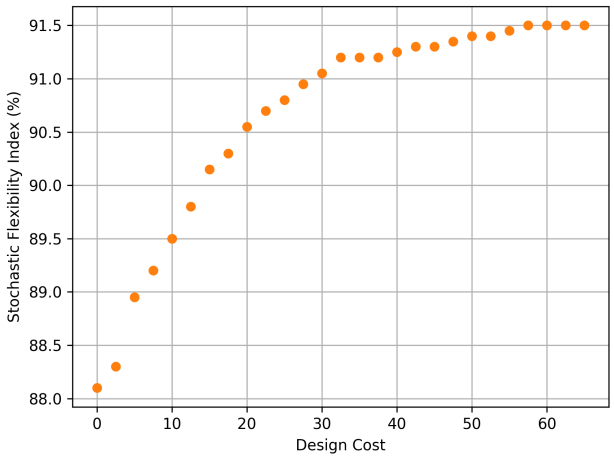



Figure 7.6: The Pareto set for the IEEE-14 power network obtained with the mixed-integer design formulation.

35.60% and the maximum $SF$ index is 70.64%. This limited maximum $SF$ index can be attributed to topological limitations in the network. Specifically, the sampled demands are Gaussian random variables and thus can be negative which in certain cases means the network is confronted with excess power production that it cannot shed. Thus, we observe that in this case increasing the network capacities alone cannot provide a $SF$ index near 100% due to topographical limitations. Instead, the system topology would need to be modified to enhance its flexibility (e.g., add a sink to shed excess power production). This highlights how this framework is useful in guiding system design by demonstrating to what degree the system flexibility can be bettered via continuous design variables such as capacity.

A key observation is that only 12 of the 30 problems converged within the time limit. This unfavorable behavior is likely due to the large-U constraints which can lead to weak linear program (LP) relaxations. This scalability limitation clearly demonstrates that Problem (7.13) can become impractical for moderate and large sized systems. We note some advanced methods such as those proposed in [173, 174] can be used to select smaller $U$ values that might strengthen the LP relaxations. However, our proposed continuous re-

laxation entails a more efficient strategy since it does not require the solution of a MIP. Another observation drawn from the solution times in Table A.7 is that Pareto pairs near minimum or maximum design cost require significantly less computational time relative to the intermediate pairs. This trend is also prevalent in the computational results of the IEEE-14 network and the three-node network.

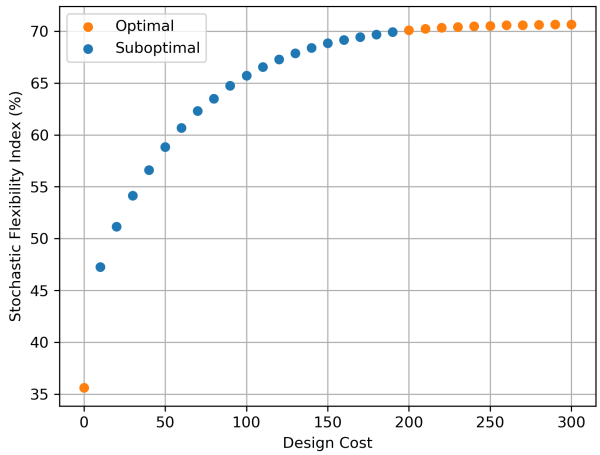



Figure 7.7: The Pareto set for the 141-node power network obtained with the mixed-integer design formulation. Here, the optimal points are those that solved within the time limit of 3,600s.

### 7.3.3 *Continuous Formulation*

We compare the Pareto pairs of the mixed-integer formulation to those obtained with the continuous formulation:

$$
\begin{aligned}
\max_{d_b^s, d_l^a, a_l^k, s_b^k, y^k} \quad & \frac{1}{|K|} \sum_{k \in K} (1 - y^k) \\
\text{s.t.} \quad & -a_l^C - d_l^a - a_l^k \leq y^k U, && l \in \mathcal{A}, \;\; k \in K \\
& -a_l^C - d_l^a + a_l^k \leq y^k U, && l \in \mathcal{A}, \;\; k \in K \\
& -s_b^k \leq y^k U, && b \in \mathcal{S}, \;\; k \in K \\
& -s_b^C - d_b^s + s_b^k \leq y^k U, && b \in \mathcal{S}, \;\; k \in K \\
& \sum_{l \in \mathcal{A}_n^{rec}} a_l^k - \sum_{l \in \mathcal{A}_n^{snd}} a_l^k + \sum_{b \in \mathcal{S}_n} s_b^k - \sum_{m \in \mathcal{R}_n} r_m^k = 0, && n \in \mathcal{C}, \;\; k \in K \\
& \frac{1}{\sqrt{n_d}} \left( \sum_{b \in \mathcal{S}} d_b^s + \sum_{l \in \mathcal{A}} d_l^a \right) \leq \epsilon_c \\
& 0 \leq y^k \leq 1, && k \in K \\
& d_b^s \geq 0, \;\; d_l^a \geq 0, && b \in \mathcal{S}, \;\; l \in \mathcal{A}.
\end{aligned}
\tag{7.19}
$$

The $\bar{SF}_K$ index is determined after solving this problem via the indicator function $\mathbb{1}_{y^k}$ as described in Section 7.2.3. The three-node network is again analyzed under the same conditions described in Section 7.3.2 (i.e., the same samples and $\epsilon_c$ values). For the small 3-node network, the problem is an LP with 4,003 variables and 9,001 constraints. Each solution is also verified to be integer feasible by ensuring the formulation is feasible with the values of the $y^k$ variables fixed to their indicated values as described in Section 7.2.3. The numerical results are provided in Tables A.4 and A.5 in the Appendix and Figure 7.8 juxtaposes these results with those obtained with the mixed-integer formulation. Here, we note that the Pareto pairs are equivalent and that the mixed-integer $y^k$ values exactly match the indicated $y^k$ values obtained with the continuous formulation (i.e., the same active constraints are identified). Also, each continuous formulation solution only required

0.0049s on average to converge which equates to a 74% reduction in computational cost.

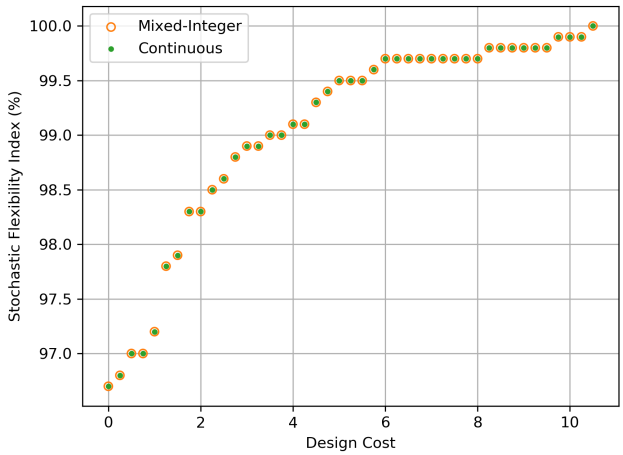



Figure 7.8: The Pareto sets for three-node network obtained with mixed-integer and continuous formulations.

We also applied Problem (7.19) to the IEEE-14 node power network with the same samples and conditions used in Section 7.3.2. This is an LP with 52,025 variables and 128,001 constraints. The indicated values of $y^k$ are all verified to be integer feasible. These results are detailed in Table A.6 in the Appendix and Figure 7.9 compares the results with those obtained with the mixed-integer formulation. Again, we note that the continuous formulation is able to recover the same Pareto set as the mixed-integer counterpart. Only the last Pareto pair differs (by one $y^k$ out of 2,000). All other pairs are identical and identify exactly the same sets of active constraints. Furthermore, the continuous formulation only requires 1.74s on average to solve, which corresponds to a 96% reduction in computational cost relative to the mixed-integer formulation.

Finally, we applied Problem (7.19) to the 141-node power network under the same conditions detailed in Section 7.3.2. This creates a large-scale LP with 1,420,141 variables and 4,230,001 constraints. The indicated values of $y^k$ are again all verified to be integer feasible. These results are provided in Table A.7 in the Appendix, and Figure 7.10 shows the Pareto pairs obtained from both formulations. In this case, the Pareto solutions are

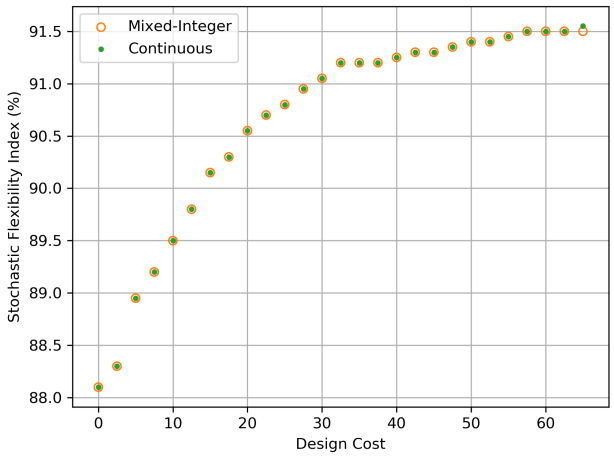



Figure 7.9: Pareto sets for IEEE-14 node power network obtained with mixed-integer and continuous formulations.

very similar, but differences are more perceptible but rather small. Specifically, the identified values of $y^k$ differ by 0.49% on average relative to the optimal values obtained via the mixed-integer formulation (i.e., 49 differences in the set of $y^k$ values out of the 10,000). The continuous Pareto pairs only required 13.41s on average to solve (in contrast to the mixed-integer results for which the majority were unable to converge within the 3,600s time limit).

Our results highlight that the continuous optimal design formulation provides high quality solutions with significant reductions in computational time. As mentioned in Section 7.2.3, the high quality of the solutions can likely be attributed to the degenerate nature of the SF index.

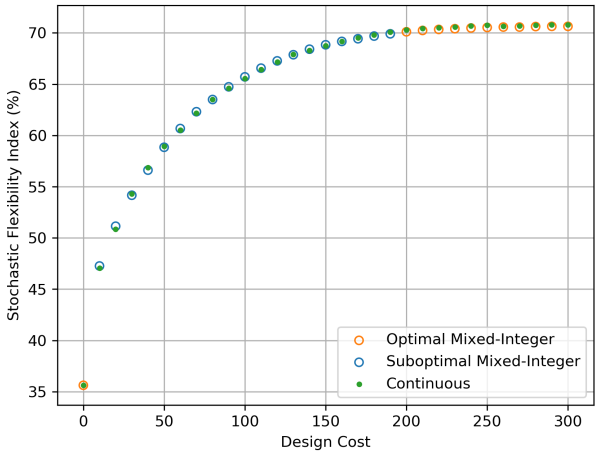



Figure 7.10: Pareto sets for 141-node power network obtained with the mixed-integer and continuous formulations.

# Chapter 8

RELIABILITY MEASURES

The content of this chapter is published in [16].

## 8.1 Introduction

In this chapter, we investigate the problem of quantifying the reliability of complex systems and of designing systems of maximum reliability. Such problems have a wide range of applications such as supply chains, transportation networks, energy networks, process networks, sensor networks, and control networks [175]. In these applications, it is vital to design systems that maintain functionality in the face of natural and man-made events (e.g., mechanical failures, power outages, weather, and cyber-attacks) [176]. Despite its practical importance, quantifying the reliability of complex systems remains a technical challenge.

We show that reliability can be computed by solving a stochastic mixed-integer program. This framework allows us to handle arbitrary system topologies, probability distributions to characterize different types of failures, and system constraints. Moreover, the stochastic program can be easily incorporated within optimal design formulations. We also provide evidence that accurate solutions for large systems can be obtained by solving purely continuous relaxations.

## 8.2 Graph Abstraction and Model

We model a system as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$ with components $\mathcal{N}$ (nodes) and $\mathcal{E}$ (edges). We use $n \in \mathcal{N}$ and $e \in \mathcal{E}$ to represent specific nodes and edges in the graph, respectively. The set of edges originating at node $n$ is denoted as $\mathcal{E}_{in}(n) \subseteq \mathcal{E}$ and the set of edges ending a node $n$ is denoted as $\mathcal{E}_{out}(n) \subseteq \mathcal{E}$. The set of supporting nodes for an edge $e$ (the pair of nodes connected by the edge) is denoted $\mathcal{N}(e) \subseteq \mathcal{N}$. A schematic representation of the graph notation is provided in Figure 8.1.

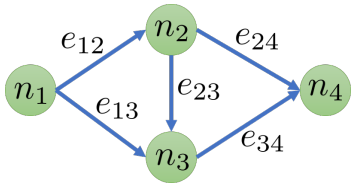



Figure 8.1: Representation of a system as a directed graph with node set $\mathcal{N} = \{n_1, n_2, n_3, n_4\}$ and edge set $\mathcal{E} = \{e_{12}, e_{13}, e_{23}, e_{24}, e_{34}\}$.

The topology of the system $\mathcal{G}(\mathcal{N}, \mathcal{E})$ is encoded in the incidence matrix $A \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{E}|}$ where $A_{ne} = 1$ if $e \in \mathcal{E}_{in}(n)$, $A_{ne} = -1$ if $e \in \mathcal{E}_{out}(n)$, or $A_{ne} = 0$ otherwise. The nominal topology $A$ is subject to *failures* of its components (nodes and edges); as such, we define the *perturbed* incidence matrix as a random matrix $A(\xi_{\mathcal{N}}, \xi_{\mathcal{E}})$. Here, $\xi_{\mathcal{N}} \in \mathbb{R}^{|\mathcal{N}|}$ is the realization of a discrete (binary) random vector that indicates the set of nodes that function ($\xi_{\mathcal{N},n} = 1$ if node $n$ functions) or do not function ($\xi_{\mathcal{N},n} = 0$ if $n$ does not function). Similarly, $\xi_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$ denotes the realization of a binary random vector that indicates the set of nodes that function ($\xi_{\mathcal{E},e} = 1$) or do not function ($\xi_{\mathcal{E},e} = 0$). Under these definitions, the perturbed incidence matrix under realization $\xi := (\xi_{\mathcal{N}}, \xi_{\mathcal{E}})$ can be computed as:

$$A(\xi) := \Xi_{\mathcal{N}} A \Xi_{\mathcal{E}} \tag{8.1}$$

where $\Xi_{\mathcal{N}} \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$, $\Xi_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ are diagonal matrices of the form $\Xi_{\mathcal{N}} = \text{diag}(\xi_{\mathcal{N}})$ and $\Xi_{\mathcal{E}} = \text{diag}(\xi_{\mathcal{E}})$, respectively. In a stochastic programming context, one can interpret

$A(\xi)$ as a random technology matrix [6]. The elements of the perturbed incidence matrix can also be written as:

$$A_{ne}(\xi) = A_{ne} \cdot \xi_{\mathcal{N},n} \cdot \xi_{\mathcal{E},e},\ n \in \mathcal{N},\ e \in \mathcal{E}. \tag{8.2}$$

In other words, $A_{ne}(\xi) = 0$ (entry does not exist) if either node $n$ or edge $e$ fails (do not exist) in scenario $\xi$.

We use a network flow model to represent paths between nodes. Specifically, we define a set of *source* nodes as $\mathcal{N}_{so} \subseteq \mathcal{N}$ with associated source flows $d_n > 0$, a set of *sink* nodes as $\mathcal{N}_{si} \subseteq \mathcal{N}$ with associated sink flows $d_n < 0$, and a set of *relay* nodes as $\mathcal{N}_{re} \subseteq \mathcal{N}$ with associated flows $d_n = 0$. We observe that the source and sink flows are *fixed*. Under these definitions, the network flow representation can be expressed as:

$$\sum_{e \in \mathcal{E}} A_{ne}(\xi) z_e + d_n = 0,\ n \in \mathcal{N} \tag{8.3}$$

where $z_e \in \mathbb{R}_+$ is the flow along edge $e \in \mathcal{E}$. The network flow model can also be expressed in compact form as:

$$A(\xi)z + d = 0. \tag{8.4}$$

In our framework, we expand this basic network flow model to capture the possibility of readjusting flows in order to maintain system functionality. This can be done by allowing some nodes $\mathcal{N}_u \subseteq \mathcal{N}$ to have controllable flows $u_n \in \mathbb{R}_+$. Moreover, in many applications, the edge flows $z$ and the controls $u$ have physical meaning and are thus subject to constraints; we capture such constraints by using feasible sets $\mathcal{Z} \subseteq \mathbb{R}^{|\mathcal{E}|}$ and $\mathcal{U} \subseteq \mathbb{R}^{|\mathcal{N}_u|}$. With this, we define the extended network flow model as:

$$A(\xi)z + u + d = 0 \tag{8.5a}$$

$$u \in \mathcal{U},\ z \in \mathcal{Z}. \tag{8.5b}$$

In this representation, the set $\mathcal{U}$ is constructed in a way that it restricts control at certain nodes. For instance, we consider the box control set:

$$\mathcal{U} = \{u \,:\, u_n = 0,\ n \notin \mathcal{N}_u \ \&\ \underline{u}_n \leq u_n \leq \overline{u}_n,\ n \in \mathcal{N}_u\}. \tag{8.6}$$

For simplicity, we assume that the feasible set for flows is also a box set of the form:

$$\mathcal{Z} = \{z \,:\, \underline{z}_e \leq z_e \leq \overline{z}_e,\ e \in \mathcal{E}\}. \tag{8.7}$$

## 8.3 Reliability Measures

In this section, we propose a new measure for system reliability and discuss effect methods to compute it.

### 8.3.1 *Definition*

A *reliability measure* seeks to quantify the probability that a system remains functional under random component failures. Under a graph representation, the system is said to be functional if there exists at least one path that connects each sink node to a source node. For a particular realization $\xi$ (with associated topology $A(\xi)$) and in the absence of controls and constraints, the functionality of a system can be checked by using the *reliability function*:

$$\psi(A, \xi) := \begin{cases} 1 & \text{if } \exists\, z : A(\xi)z + d = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8.8}$$

This function uses the network flow representation to check if there exist a set of flows $z$ that connect sinks and sources. This is based on the observation that, if a path does not exist between a sink and at least one source node (e.g., the network becomes disconnected in a given failure scenario), then there is no set of flows $z$ that satisfies the flow constraint $A(\xi)z + d = 0$.

The traditional definition of reliability does not account for constraints and does not account for the possibility to control flows. To account for these features, we extend the *reliability function* as:

$$\psi(A, \xi, \mathcal{Z}, \mathcal{U}) := \begin{cases} 1 & \text{if } \exists\, z \in \mathcal{Z}, u \in \mathcal{U} : A(\xi)z + u + d = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8.9}$$

We use this extended function to define the *reliability measure*:

$$R(A, \mathcal{Z}, \mathcal{U}) := \mathbb{P}(\psi(A(\xi), \mathcal{Z}, \mathcal{U}) = 1). \tag{8.10}$$

This measure function is the probability that the system remains functional. A similar measure has been proposed to measure system *flexibility* which, in our setting, would represent the ability of a system to withstand perturbations in the source and sink flows $d$ (exogenous disturbances) [11, 52, 177, 178]. Therefore, we highlight that a key distinction between flexibility and reliability is that the former deals with continuous perturbations while the later deals with discrete perturbations.

### 8.3.2 *Evaluation*

We motivate the discussion by considering a simple setting with a single-input and single-output graph. Under this setting, the sets $\mathcal{N}_{so}$ and $\mathcal{N}_{si}$ are singletons and thus the system is said to remain functional if there exists at least one path between the source and the sink node. Equivalently, given a fixed source flow, the system is functional if we can find a set of edge flows that satisfy the fixed sink flow. Under this logic, we can compute $\psi(A, \xi)$ by finding a feasible solution for a network flow problem and this problem can

be cast as a mixed-integer linear program (MILP) of the form:

$$\begin{aligned}
\psi(A,\xi) = \quad & \max_{y,z} \quad (1-y) \\
& \text{s.t.} \quad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e = 0, && n \in \mathcal{N}_{re} \\
& \qquad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e + d_n \cdot (1-y) = 0, && n \in \mathcal{N}_{so} \\
& \qquad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e + d_n \cdot (1-y) = 0, && n \in \mathcal{N}_{si} \\
& \qquad z_e \geq 0, && e \in \mathcal{E} \\
& \qquad y \in \{0,1\}.
\end{aligned} \tag{8.11}$$

Here, we arbitrarily set the source and sink flows to $d_n = 1$ and $d_n = -1$, respectively. This is done without loss of generality because the flows do not necessarily have physical meaning (in more general settings they might have meaning). We use the binary variable $y \in \{0,1\}$ to relax the balances at the source and sink nodes (i.e., if $y = 0$ then the network flow system has a feasible solution and if $y = 1$ then it does not). If the network flow system does not have a solution then we obtain the trivial flow solution $z_e = 0$ for all $e \in \mathcal{E}$. We thus have that the reliability measure is given by $\psi(A,\xi) = 1 - y^*$ and we note that the maximization problem is equivalent to minimize $y$. The MILP can be relaxed by setting $0 \leq y \leq 1$; interestingly, this LP is guaranteed to deliver an optimal (binary) solution for the MILP (see Appendix).

Problem (8.11) can be easily generalized to compute the reliability measure for graphs with multiple sources and sinks and with controllable flows. This can be done by solving

the MILP:

$$\begin{aligned}
\psi(A,\xi,\mathcal{Z},\mathcal{U}) = \quad & \max_{y,z,u} \quad (1-y) \\
& \text{s.t.} \quad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e = 0, && n \in \mathcal{N}_r \\
& \qquad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e + d_n \cdot (1-y) + u_n = 0, && n \in \mathcal{N}_{so} \\
& \qquad \sum_{e\in\mathcal{E}} A_{ne}(\xi) z_e + d_n \cdot (1-y) + u_n = 0, && n \in \mathcal{N}_{si} \\
& \qquad u \in \mathcal{U},\ z \in \mathcal{Z} \\
& \qquad y \in \{0,1\}.
\end{aligned} \tag{8.12}$$

This MILP determines if all the sink flows can be satisfied via the source flows (i.e., each sink has at least one path to a source); this is true whenever $y = 0$ (which indicates that none of the source and/or sink nodes needs to be relaxed to achieve a feasible solution).

The MILP representation of the reliability function reveals that the measure $R(A,\mathcal{Z},\mathcal{U})$ is a *joint chance constraint*. This chance constraint can be approximated using MC samples $\xi^k,\ k \in \mathcal{K}$ as [179]:

$$R(A,\mathcal{Z},\mathcal{U}) \approx \frac{1}{|\mathcal{K}|} \sum_{k\in\mathcal{K}} \psi(A,\xi^k,\mathcal{Z},\mathcal{U}). \tag{8.13}$$

By the law of large numbers, this sample average approximation becomes asymptotically exact as the number of samples increases [180]; moreover, the approximation converges exponentially [26]. Combining problems (8.13) and (8.12), we obtain the following ap-

proximation of the reliability measure:

$$
\begin{aligned}
R(A,\mathcal{Z},\mathcal{U}) \approx \quad \max_{y^k,z^k,u^k} \quad & \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}(1-y^k) \\
\text{s.t.} \quad & \sum_{e\in\mathcal{E}} A_{ne}(\xi^k)z_e^k = 0, && n\in\mathcal{N}_{re},\ k\in\mathcal{K} \\
& \sum_{e\in\mathcal{E}} A_{ne}(\xi^k)z_e^k + d_n\cdot(1-y^k) + u_n^k = 0, && n\in\mathcal{N}_{so},\ k\in\mathcal{K} \\
& \sum_{e\in\mathcal{E}} A_{ne}(\xi^k)z_e^k + d_n\cdot(1-y^k) + u_n^k = 0, && n\in\mathcal{N}_{si},\ k\in\mathcal{K} \\
& z^k\in\mathcal{Z},\ u^k\in\mathcal{U}, && k\in\mathcal{K} \\
& y^k\in\{0,1\}, && k\in\mathcal{K}.
\end{aligned}
\tag{8.14}
$$

This problem is *fully decoupled* in the MC samples $k \in \mathcal{K}$ and thus can be trivially parallelized. It has been recently reported that a continuous relaxation of this problem (in combination with an appropriate rounding strategy) provides high quality approximations of the exact solution [1]. Specifically, we can relax $y^k \in \{0,1\}$ to $0 \leq y^k \leq 1$ and then round the optimized relaxed $y^{k*}$ values to 1 if they are nonzero. This approach is analogous to employing slack variables to identify active and inactive sets of constraints. In the following section we provide numerical evidence that this relaxation approach is effective. The exact relaxation result for the simple reliability problem (8.11) provides some intuition as to why this happens. However, establishing a theoretical justification in a more complex setting with constraints and controllable flows is difficult and is left as a topic of future work.

The MILP representation can be extended in a number of ways to capture desirable decision-making logic. For instance, one might want to relax the requirement that paths must exist to all sink nodes and instead require that only a subset of nodes are reachable. This can be done by introducing binary variables for all sink nodes $y_n^k$ and by solving the

problem:

$$
\begin{aligned}
R(A,\mathcal{Z},\mathcal{U}) \approx \quad & \max_{y^k,z^k,u^k} \quad \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}} L(y^k) \\
& \text{s.t.} \quad \sum_{e\in\mathcal{E}} A_{ne}(\xi^k) z_e^k = 0, && n\in\mathcal{N}_{re},\ k\in\mathcal{K} \\
& \quad \sum_{e\in\mathcal{E}} A_{ne}(\xi^k) z_e^k + d_n\cdot(1-y_n^k) + u_n^k = 0, && n\in\mathcal{N}_{so},\ k\in\mathcal{K} \\
& \quad \sum_{e\in\mathcal{E}} A_{ne}(\xi^k) z_e^k + d_n\cdot(1-y_n^k) + u_n^k = 0, && n\in\mathcal{N}_{si},\ k\in\mathcal{K} \\
& \quad z^k\in\mathcal{Z},\ u^k\in\mathcal{U}, && k\in\mathcal{K} \\
& \quad y_n^k\in\{0,1\}, && k\in\mathcal{K},\ n\in\mathcal{N}_{si}\cup\mathcal{N}_{so}.
\end{aligned}
\tag{8.15}
$$

Here, $L(y^k)$ is a logic function which is set to one if a subset of sinks of interest are reachable (or is set to zero otherwise).

## 8.4 Reliability Design

In this section, we propose optimal design problems that seek promote system reliability while minimizing the design cost. We also show how these can be efficiently implemented.

### 8.4.1 *Definition*

We are interested in using the reliability measure to find system designs that maximize reliability. In this task, one often needs to trade-off cost $c(A,\mathcal{Z},\mathcal{U})$ and reliability, giving rise to the abstract problem:

$$
\begin{aligned}
\max_{A,\mathcal{Z},\mathcal{U}} \quad & R(A,\mathcal{Z},\mathcal{U}) \\
\text{s.t.} \quad & c(A,\mathcal{Z},\mathcal{U}) \le \epsilon
\end{aligned}
\tag{8.16}
$$

where $\epsilon\in\mathbb{R}$ is a cost budget that is spanned to find Pareto pairs $(c^*,R^*)$. We highlight the dependence of the cost measure and reliability measure on the topological design

(given by the incidence matrix $A$) and on the operational design (given by the sets $\mathcal{Z}, \mathcal{U}$).

### 8.4.2 *Evaluation*

The design problem (8.16) aims to make topological and capacity changes to a nominal network in order to maximize reliability (under a given cost budget). To formulate this problem, we recall that the base topology of the system is given by the graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$ with associated incidence matrix $A$, nodes $\mathcal{N}$, and $\mathcal{E}$. Our goal is this formulation to expand the number of edges in order to maximize reliability. This is done by defining an expanded set of edges $\bar{\mathcal{E}}$ such that $\mathcal{E} \subset \bar{\mathcal{E}}$. The expanded set of edges has an associated incidence matrix $\bar{A}$. In other words, the new incidence matrix has an expanded set of connections between the nodes. We represent the added set of edges as $\hat{\mathcal{E}} := \bar{\mathcal{E}} \setminus \mathcal{E}$. In our design problem, we also seek to expand the set of feasible edge flows and control flows (to model capacity expansions). The design problem is cast as the following MILP:

$$
\begin{aligned}
\max_{v, \underline{z}, \overline{z}, \underline{u}, \overline{u}, z^k, y^k, u^k} \quad & \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} L(y^k) \\
\text{s.t.} \quad & c(v, \underline{z}, \overline{z}, \underline{u}, \overline{u}) \leq \epsilon \\
& \sum_{e \in \bar{\mathcal{E}}} A^k_{ne} z^k_e = 0, && n \in \mathcal{N}_{re},\ k \in \mathcal{K} \\
& \sum_{e \in \bar{\mathcal{E}}} A^k_{ne} z^k_e + d_n \cdot (1 - y^k_n) + u^k_n = 0, && n \in \mathcal{N}_{so},\ k \in \mathcal{K} \\
& \sum_{e \in \bar{\mathcal{E}}} A^k_{ne} z^k_e + d_n \cdot (1 - y^k_n) + u^k_n = 0, && n \in \mathcal{N}_{si},\ k \in \mathcal{K} \\
& A^k_{ne} = \bar{A}_{ne} \cdot \xi^k_{\mathcal{N},n} \cdot \xi^k_{\mathcal{E},e}, && n \in \mathcal{N},\ e \in \bar{\mathcal{E}},\ k \in \mathcal{K} \\
& A^k_{ne} = \bar{A}_{ne} \cdot \xi^k_{\mathcal{N},n} \cdot \xi^k_{\mathcal{E},e} \cdot v_e, && n \in \mathcal{N},\ e \in \hat{\mathcal{E}},\ k \in \mathcal{K} \\
& \underline{z} \leq z^k \leq \overline{z},\ \underline{u} \leq u^k \leq \overline{u}, && k \in \mathcal{K} \\
& y^k_n \in \{0, 1\}, && n \in \mathcal{N}_{si} \cup \mathcal{N}_{so},\ k \in \mathcal{K} \\
& v_e \in \{0, 1\}, && e \in \hat{\mathcal{E}} \\
& \underline{z}, \overline{z} \in \overline{\mathcal{Z}},\ \underline{u}, \overline{u} \in \overline{\mathcal{U}}.
\end{aligned}
\tag{8.17}
$$

Here, the sets $\overline{\mathcal{Z}}$ and $\overline{\mathcal{U}}$ include possible design values for flow and control bounds. Also, $v \in \{0,1\}^{|\hat{\mathcal{E}}|}$ denote topological design variables for selecting which of the candidate edges are included in the new design (if none are added then $v_e = 0$ for all $e \in \hat{\mathcal{E}}$ and the network retains its nominal topology). We note that the abstract design cost function $c(A, \mathcal{Z}, \mathcal{U})$ can now be expressed in the parametric form $c(v, \underline{z}, \overline{z}, \underline{u}, \overline{u})$. The proposed design formulation seeks to highlight the modeling flexibility provided by the proposed stochastic programming framework.

## 8.5 Case Studies

We analyze the behavior of the proposed framework by applying it to distribution networks. We consider a simple three-node network and the IEEE-14 power distribution network. We also consider a simple parallel-series RBD system to illustrate how the proposed stochastic programming framework is consistent with the analytical RBD solution. All formulations are implemented in `JuMP v0.18.5` [68] and are solved using `Gurobi v7.5.1` on a Intel® Core™ i7-7500U machine running at 2.90 GHz with 4 hardware threads and 16 GB of RAM. All results can be reproduced using the scripts provided in https://github.com/zavalab/JuliaBox/tree/master/ReliableDesign.

### 8.5.1 *Simple Parallel-Series System*

We consider a simple parallel-series system to highlight that the stochastic programming approach is consistent. The system of interest is represented by the reliability block diagram shown in Figure 8.2. This system seeks to pump a flow stream using two pumps and valves in parallel. The parallel design topology enhances the reliability of the system (compared to a topology with a single pump and valve).

Traditional RBD methods can be leveraged to obtain an analytic representation for the overall reliability of the system since this system features a single source and sink [55]. In particular, the analytic reliability measure is computed by aggregating the component

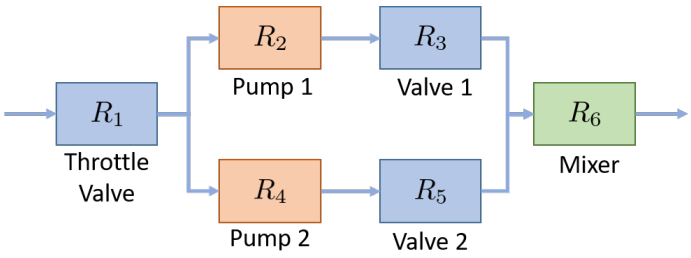



Figure 8.2: Reliability block diagram for a pump system.

reliabilites according to their respective connectivities. Specifically, the reliability of $m$ components in series configurations are given by:

$$R_s = \prod_i^m R_i. \tag{8.18}$$

The reliability of a parallel configuration is given by:

$$R_p = 1 - \prod_i^m (1 - R_i). \tag{8.19}$$

Following these basic rules, the reliability of the system of interest can be computed as:

$$R_{overall} = R_1(1 - (1 - R_2R_3)(1 - R_4R_5))R_6. \tag{8.20}$$

For simplicity, we let each component lifetime be described by an exponential distribution with a mean lifetime of 100 years and we evaluate reliability after 5 years of operation. The reliability for each component is given by the exponential cumulative distribution function evaluated at 5 years (i.e., $R_i = \exp(-5/100)$). From Equation (8.20) we thus obtain the overall reliability $R_{overall} = 89.66\%$.

To demonstrate the equivalence of the proposed stochastic programming setting, we use MC samples drawn from the component distribution functions (a component fails if the lifetime is above the desired threshold of 5 years). We use a total of 10,000 MC samples and solve Problem (8.14). Here, we let the throttle valve be the source node and the mixer be the sink node with $d_n = 1$ and $d_n = -1$, respectively. Furthermore, we

set $\mathcal{U} = \emptyset$ (no controls) and $\mathcal{Z} = \mathbb{R}_+^{|\mathcal{E}|}$. Using this approach, the reliability measure is $R(A, \mathcal{Z}, \mathcal{U}) = 89.69\%$, which is close to the analytical solution.

### 8.5.2 *Network Models*

The systems under study are illustrated in Figures 8.3 and 8.4. We consider a simple 3-node distribution network and the IEEE 14-node power network benchmark problem. In these cases, the sink nodes $n \in \mathcal{N}_{si}$ have a fixed flow $d_n$ and the source nodes $n \in \mathcal{N}_{so}$ are controllable with capacity $\bar{u}$. Furthermore, the edges have a finite capacity $\bar{z}$. The 3-node network features a single source (a power plant) and three sink nodes (power consumers). The IEEE 14-node network exhibits a more complex topology with multiple sinks and sources. The data for this problem is obtained from `MATPOWER` [181].

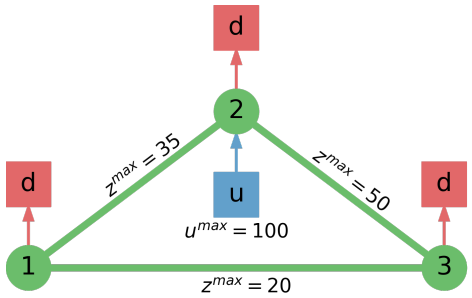



Figure 8.3: Schematic of 3-node distribution network.

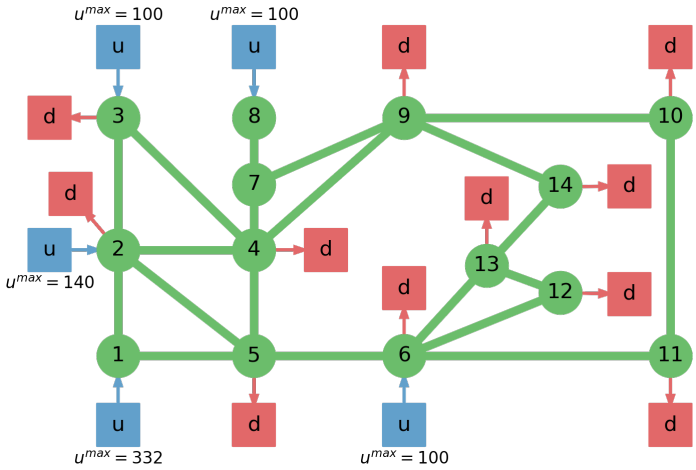



Figure 8.4: Schematic of IEEE 14-node distribution network.

For our design studies, we consider a cost function of the form:

$$c(v, \overline{z}, \overline{u}) := \sum_{e \in \mathcal{E}} (100 \cdot v_e + \overline{z}_e) + \sum_{n \in \mathcal{N}_u} \overline{u}_n \tag{8.21}$$

We consider random failures for relay nodes, source nodes, and sink nodes. Here, we model the lifetimes of the components as exponential random variables with mean lifetimes of 100, 80, and 50 years, respectively. MC failure scenarios $\xi_{\mathcal{N}}^k$, $\xi_{\mathcal{E}}^k$ are generated by sampling the exponential distributions and the components are set to failure mode if their lifetime is above a certain threshold value. The thresholds for the 3-node and IEEE 14-node networks were set to 5 and 2 years, respectively.

### 8.5.3 *Design for Maximum Reliability (Capacity Expansion)*

We first consider a design problem in which capacity is expanded (i.e., topological expansion variables $v$ are omitted). For the 3-node power network, we solve the MILP formulation to obtain 6 Pareto pairs and we use 1,000 MC samples. The Pareto pairs are plotted in Figure 8.5. Here, we note that the Pareto frontier shows abrupt changes; this is because this system is simple, and thus the solution space is small. A manifestation of this limited spaces is that the maximum possible reliability for this system is just 51.4%. This indicates that, regardless of how much capacity is provided (unlimited budget), this network will never achieve a higher reliability because of its limiting topology. In other words, the only way to increase reliability is to add edges.

We explore the Pareto solutions obtained with $\epsilon = 30$ and $\epsilon = 45$. The optimized capacities for these solutions are shown in Figure 8.6. Here, the increased capacities relative to the base design are highlighted in red. In the first case, enough capacity is added to the edges connecting nodes 2, 3, and 1 to permit the network to function in the event that the edges connecting nodes 1 and 2 fails. In the other case, enough capacity is added to the edges to permit feasible operation if any one edge fails.

We apply the same design formulation to the IEEE-14 power network problem. We

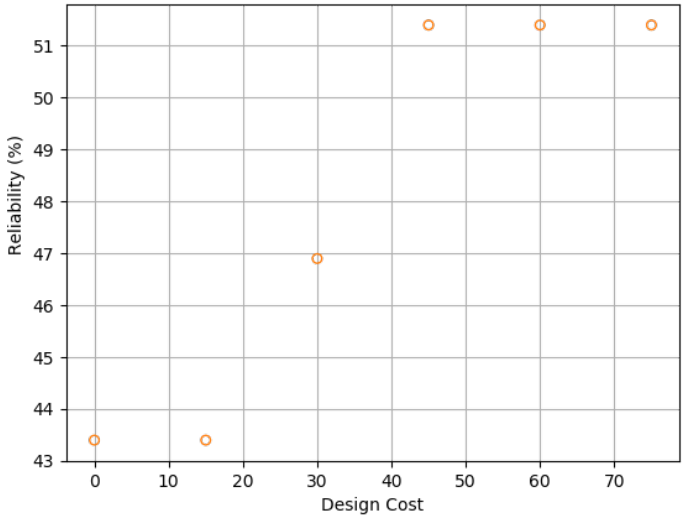



Figure 8.5: Pareto frontier for optimal capacity design of 3-node network using 1,000 MC samples.

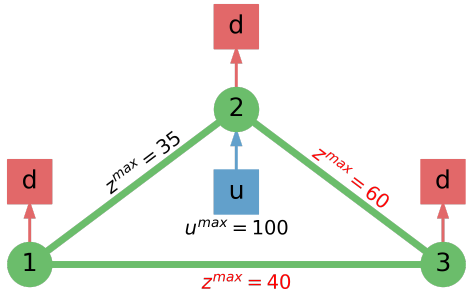



Figure 8.6: Schematic of the 3-node network corresponding to Pareto pair shown in Figure 8.5 with a design cost of 30.

compute a total of 13 Pareto pairs by varying the budget $\epsilon$ from 0 to 1800, and we use 2,000 MC samples. The solutions obtained with the MILP formulation are presented in Figure 8.7. We see that this system shows a smoother Pareto frontier because the solution space for this more complex system is larger. For this system the largest possible reliability is 78.7% (this system has more degrees of freedom).

The design obtained with a budget of $\epsilon = 400$ is shown in Figure 8.8. The expanded capacities are highlighted in red. The capacity of the supplier attached to relay node 6 is significantly expanded (which occurs because it is the only supplier that serves the right side of the network). The capacities of two edges attached to node 6 are also increased

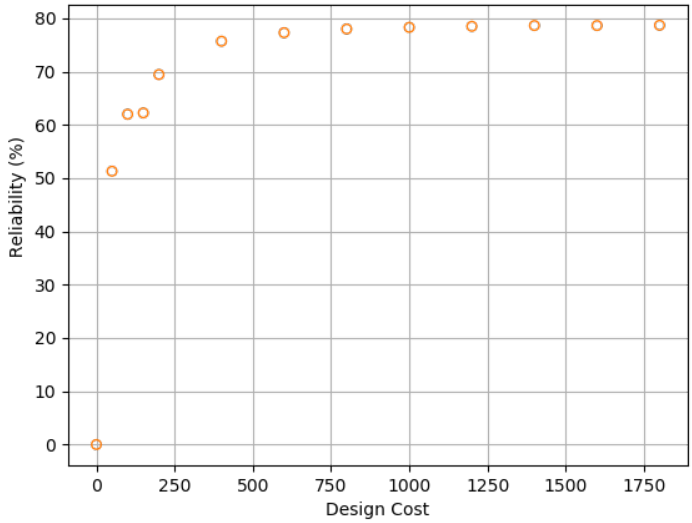



Figure 8.7: Pareto frontier for optimal design problem of IEEE 14-node network.

such that the internal demands can be satisfied if either edge fails. It is interesting to note that these 3 simple changes to the network design significantly increase the overall reliability of the system (they increase it by 24.4%).

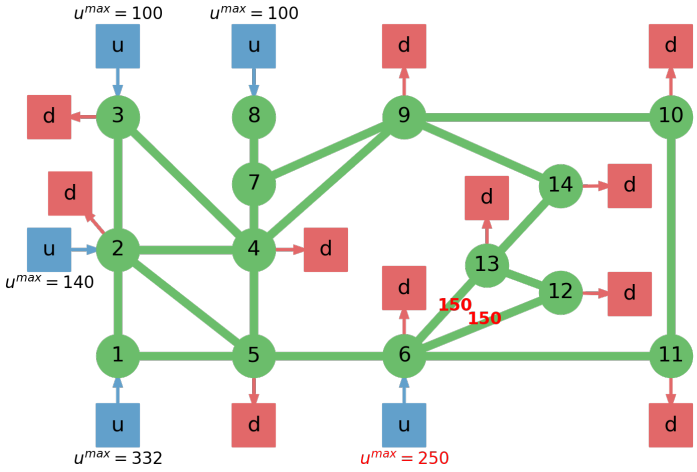



Figure 8.8: Schematic of the IEEE 14-node network corresponding to Pareto pair shown in Figure 8.7 with a cost of 400.

### 8.5.4 *Continuous Approximation for Design Problem*

We consider a continuous relaxation of the design problem; here, integer solutions are obtained by solving the relaxation and then using simple rounding. This technique is first applied to the 3-node power network using the same samples and $\epsilon$ values considered above in Section 8.5.3. In Figure 8.9, we juxtapose the resulting Pareto pairs. We observe that 5 out of 6 pairs are exactly recovered, and a pair is underestimated. In Chapter 7, we hypothesized that the quality of the approximations is the result of degeneracy associated with the joint chance constraint (i.e., multiple solutions yield the same optimal value). This simple network exhibits little degeneracy at that solution because its solution space is small. We provide an additional result that in part explains the quality of this approximation in Theorem 5.

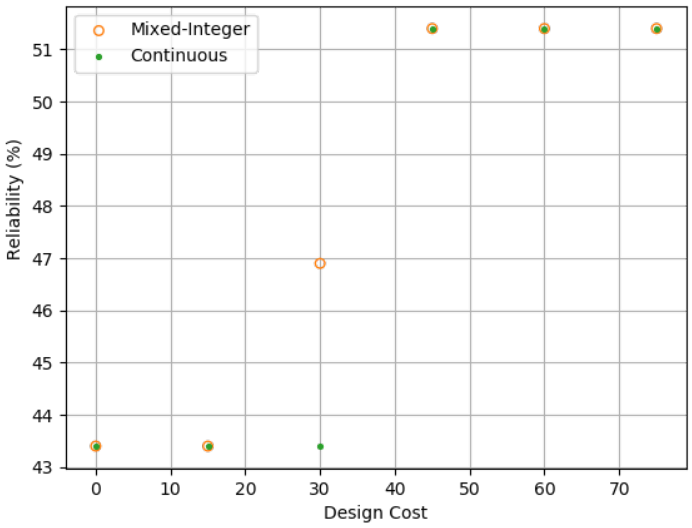



Figure 8.9: Pareto frontier for optimal capacity design of the 3-node network juxtaposing the pairs obtained from the full MILP formulation and its continuous relaxation.

Table 8.1 summarizes the performance of the MILP formulation and the continuous relaxation for the 3-node network. We observe that 5 of the 6 pairs are exactly equivalent since they have no differences in the active constraints. Also, the third pair only differs by 3.5% (which is a small gap). For this small network, the solution times are negligible,

so the benefits of the relaxation are not obvious.

Table 8.1: Performance results obtained for 3-node network using the MILP design formulation and continuous relaxation.

| Cost | MILP R (%) | LP R (%) | MILP Time (s) | LP Time (s) | Active Difference (%) |
|---|---|---|---|---|---|
| 0 | 43.4 | 43.4 | 0.0467 | 0.0349 | 0 |
| 15 | 43.4 | 43.4 | 0.0400 | 0.0468 | 0 |
| 30 | 46.9 | 43.4 | 0.0407 | 0.0478 | 3.5 |
| 45 | 51.4 | 51.4 | 0.0562 | 0.0478 | 0 |
| 60 | 51.4 | 51.4 | 0.0526 | 0.0478 | 0 |
| 75 | 51.4 | 51.4 | 0.0443 | 0.0458 | 0 |

The relaxation strategy was also applied to the IEEE 14-node power network using the same conditions specified above in Section 8.5.3. A juxtaposition of the Pareto pairs is shown in Figure 8.10. We observe that the frontier is approximated well (the majority of the pairs being exactly reproduced). Some minor discrepancies are attributed to numerical precision. A summary of the results is shown in Table 8.2. Except the third pair, the Pareto pairs only exhibit differences in the active constraints of less than 1%. We can thus see that the relaxation indeed delivers high quality approximation. Importantly, we observe that the computational time is reduced by 96%. This enables us to handle much larger networks than would be possible using the full MILP formulation.

### 8.5.5 *Design for Maximum Reliability (Topological Expansion)*

We apply the MILP formulation to the 3-node power network including the use of the topological design variables $v$. We recall that these design variables determine if a particular edge is added to the system. In other words, this more complex formulation chooses an optimal design configuration of edges and capacities where it enforces a fixed upfront cost for the use of each edge. A total of 7 Pareto solutions were computed by varying the budget $\epsilon$ from 0 to 750 and the same samples mentioned above are used. These solutions are presented in Figure 8.11. A nonzero $R$ index is not obtained until a budget of 600 is

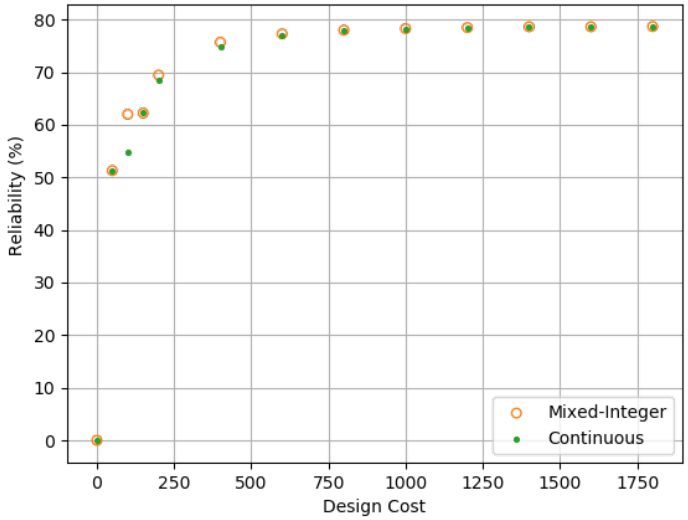



Figure 8.10: Pareto frontier for optimal capacity design of the IEEE 14-node network juxtaposing the pairs obtained from the full MILP formulation and its continuous relaxation.

Table 8.2: The performance results obtained for the IEEE 14-node network using the mixed-integer and continuous capacity design formulations.

| Design Cost (-) | MILP R (%) | LP R (%) | MILP Time (s) | LP Time (s) | Active Difference (%) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 5.42 | 1.23 | 0 |
| 50 | 51.3 | 51.3 | 58.84 | 12.84 | 0 |
| 100 | 62 | 54.9 | 78.62 | 3.68 | 7.1 |
| 150 | 62.25 | 62.25 | 95.09 | 5.75 | 0.5 |
| 200 | 69.45 | 68.4 | 83.35 | 4.75 | 1.05 |
| 400 | 75.7 | 74.85 | 124.46 | 4.84 | 0.85 |
| 600 | 77.3 | 77.1 | 168.61 | 7.39 | 0.8 |
| 800 | 78 | 77.9 | 185.11 | 3.97 | 0.2 |
| 1000 | 78.3 | 78.1 | 205.75 | 7.2 | 0.5 |
| 1200 | 78.5 | 78.45 | 186.34 | 3.27 | 0.05 |
| 1400 | 78.65 | 78.55 | 125.08 | 4.14 | 0.1 |
| 1600 | 78.65 | 78.65 | 151.48 | 2.57 | 0.1 |
| 1800 | 78.7 | 78.7 | 108.57 | 1.72 | 0 |

employed since at least 6 edges are required to allow the network to function and the capital cost of each line is 100. After this, capacity increases help improve the network until the $R$ index is maximized by adding all the edges and adding extra capacity resulting in the same best possible optimal design considered shown in Figure 8.6.

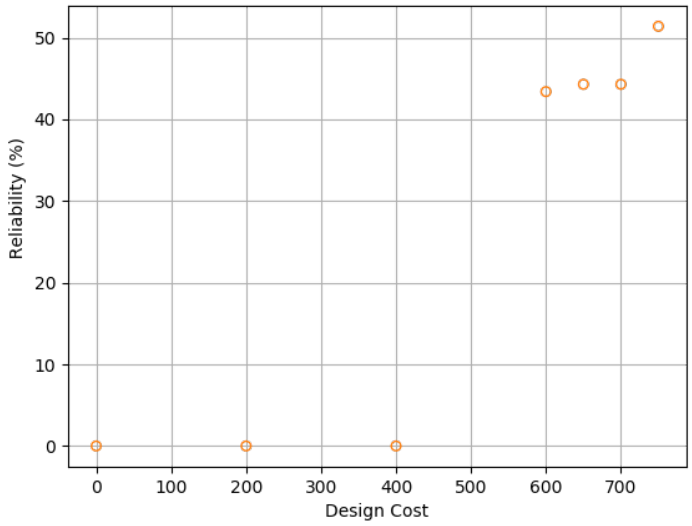



Figure 8.11: Pareto frontier for optimal topological and capacity design of the 3-node network obtained from the full MILP formulation.

The optimal design for a budget of 650 is depicted in Figure 8.12 (left) and this is compared against the design with maximum budget (right). The edges that are switched off are plotted in gray. Here, we observe that the trade-off design is able to effectively operate relative to the sample set without one of the relay edges with the addition of some capacity. The design of maximum budget employs all the edges with enough capacity to operate if any relay edge fails (making it the most robust design).

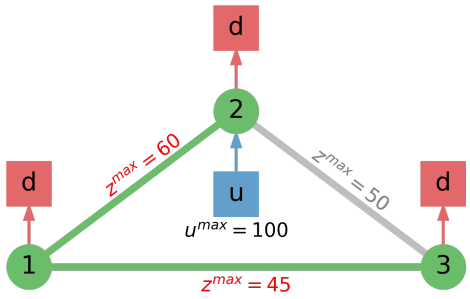



Figure 8.12: Schematic of the 3-node network corresponding to the Pareto pair shown in Figure 8.11 with a design cost of 650.

We also considered topology and capacity design decisions for the IEEE 14-node power network. A total of 27 Pareto pairs were obtained by varying the budget $\epsilon$ from 0 to 4700. The solutions are shown in Figure 8.13. The Pareto frontier was obtained

using the continuous relaxation. An average of 56 seconds were needed to compute the frontier. The reduced solution times allow us to explore more designs and to explore the reliability limits of the system. In other words, even if the relaxation cannot be guaranteed to provide an exact solution, it captures general behavior and thus can be used as an exploratory tool.

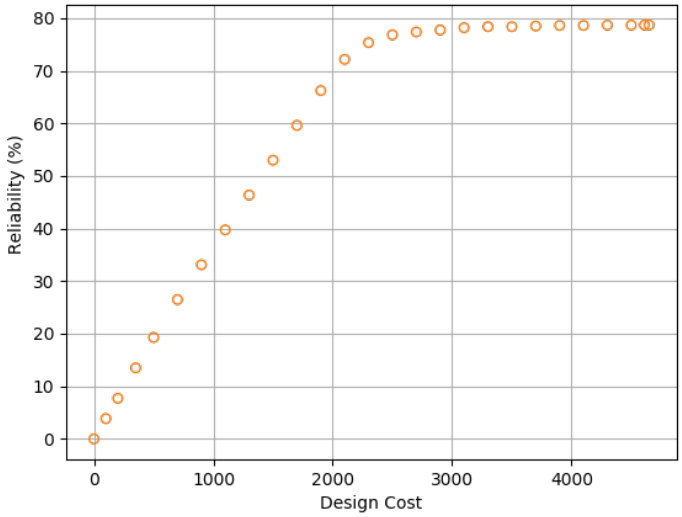



Figure 8.13: Pareto frontier for optimal topological and capacity design for IEEE 14-node network (using continuous relaxation).

The Pareto pair with a budget of 2500 is shown in Figure 8.14. The modified capacities are highlighted in red and the edges not in use are colored gray. Here we observe that reliable performance can be obtained simply by adding capacity to the right supplier and most of the edges, except the edges connecting relay nodes 1 to 2 and 4 to 7. Interestingly, this analysis shows that these edges do not impact reliability and can be eliminated (this elimination is not obvious). This highlights that the use of systematic reliability analysis techniques in being can help determine what components are truly needed and avoids over-engineering.

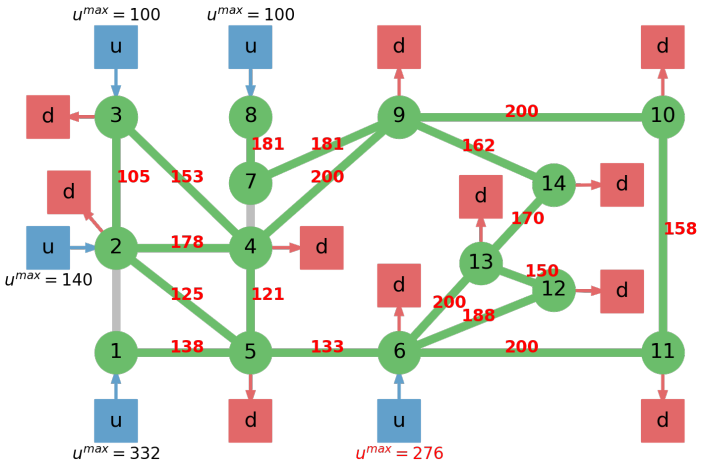



Figure 8.14: Schematic of the IEEE 14-node network corresponding to Pareto pair shown in Figure 8.13 with a cost of 2500.

# Part III

FINAL THOUGHTS

# Chapter 9

## CONCLUSIONS AND FUTURE DIRECTIONS

We now conclude this dissertation by summarizing our key findings/contributions made through the work presented in Chapters 2 - 8. We also identify how these contributions can be enriched further with future research.

## 9.1 Contributions

**Unifying Abstraction for InfiniteOpt Problems**

Chapter 2 presents a unifying abstraction for representing infinite-dimensional optimization (InfiniteOpt) problems that appear across diverse disciplines. This unifying abstraction introduces the notions of infinite variables (variables parameterized over infinite-dimensional domains). The abstraction also uses measure and differential operators that facilitate the construction of objectives and constraints. The proposed abstraction facilitates knowledge transfer; specifically, it helps identify and transfer modeling constructs across disciplines (as shown in Chapters 3 - 5). The proposed modeling abstraction aims also to decouple the formulation from transformation schemes (e.g., direct transcription), as we believe that this facilitates analysis and implementation.

**Generalized Risk Measures**

In Chapter 3, we show that risk measures used in stochastic optimization (SO) can be

transferred (along with their mathematical properties) to dynamic optimization (DO) and other fields such as PDE-constrained optimization. This enables a new class of DO formulations that shape time trajectories in interesting and useful ways. The transfer of insights across the SO and DO disciplines is facilitated by our unifying infinite-dimensional abstraction.

**Random Field Optimization**

Following our unifying abstraction, we present (in Chapter 4) a general framework for leveraging random field uncertainty in an optimization context that allows us to conduct uncertainty propagation over general domains (e.g., space and/or time) in a new area that we call random field optimization. This provides us with a unified approach to model and optimize engineering systems that are inherently random, since random field theory provides a general framework for uncertainty (producing stochastic processes as a special case). Moreover, our proposed framework for modeling and optimizing RFO problems is general, capturing existing approaches such as multi-stage stochastic optimization as special cases. We hope that this framework that forms the basis of random field optimization will serve as a foundation for further advancement and innovation for tackling this challenging problem class.

**Event Constraints**

In part, Chapter 5 introduces a new family of constraints that we call event constraints. These constraints generalize the notion of chance constraints, excursion set conditions, and exceedance probabilities that are used in different scientific disciplines (e.g., stochastic optimization, reliability engineering, and random fields). This affords us greater flexibility in posing measured constraint functions over logical events of arbitrary complexity. Moreover, this approach is generic and can be applied in diverse optimization disciplines.

**Generalized Uncertainty Sets in Flexibility Analysis**

In Chapter 6, we present a number of fundamental extensions to traditional flexibility analysis. One key discovery was that any compact uncertainty set such as those featured in Table 6.1 can readily be incorporated using existing solution techniques for assessing

the flexibility index. This insight enables us to select from a wide breadth of uncertainty sets to more accurately capture the underlying behavior of uncertainties that a system is subjected to; for instance, we can use ellipsoidal sets to capture the correlation exhibited between uncertainty parameters.

**Limiting Constraint Ranking**

Moreover, in Chapter 6, we establish a new approach for identifying and quantifying the vulnerabilities of a system design. This entails iterative solution of the flexibility index problem that identifies limiting constraints (with their associated flexibility indices). This straightforward approach gives us a systematic way to interrogate the limiting behavior of designs that exhibit arbitrary complexity. We can employ this analysis procedure to better inform design retrofits and/or identify components that warrant close monitoring.

**Abstraction for Reliability Analysis**

Chapter 8 proposes stochastic programming formulations to compute the reliability of complex systems. Specifically, the proposed reliability measure uses a graph representation of the system and aims to identify the probability that sink nodes are reachable by source nodes. This measure can be computed by solving a network flow problem, can be easily extended to incorporate constraints, and can be easily embedded in design formulations. We also show that the reliability measure can be computed by solving a stochastic mixed-integer program and that a continuous relaxation of this problem provides high-quality solutions (as discussed more in the following highlight).

**Scalable Optimal Design of Flexibility/Reliability**

In Chapters 7 and 8, we propose tractable stochastic programming formulations for optimally investigating the tradeoff between system flexibility/reliability and design cost. In particular, we are able to reproduce the results of the mixed-integer joint chance constraint based formulations using a continuous conflict resolution program at a significantly reduced computational cost. This greatly enhances the tractability of these optimal design problems and has the potential to analogously enhance other joint chance constrained problems. This approach has already enabled a number of advancements in the litera-

ture; for instance, the hybrid pharmaceutical process design approach proposed in [182] leverages this relaxation approach to significantly increase the scalability of their design tool.

**Software**

Our proposed innovations are based on modeling abstractions that capture general problem classes. This workflow has readily enabled the development of software tools such as `InfiniteOpt.jl` and `FlexibilityAnalysis.jl`. These promote innovation by providing general tools that inspire novel formulations (e.g., those arising from transferring modeling objects between fields). Most of the innovations presented in this dissertation were inspired from software-enabled experimentation. Such tools also lower the barrier to implementing advanced modeling techniques (e.g., orthogonal collocation and event constraints) and thus encourage researchers to quickly test a variety of approaches for a candidate research problem. Finally, such tools greatly increase the accessibility of these advanced modeling approaches to a non-expert audience (e.g., undergraduate engineers).

## 9.2 Future Research Directions

**Random Field Optimization Transformations**

The simple direct transcription transformations that we use for solving random field optimization problems in Chapter 4 quickly become intractable due to considering the necessary combinatorics. This means that only a few random field samples can be used to represent the uncertainty as shown in Section 4.5 which limits the solution accuracy. To address this shortcoming, we plan to develop more advanced transformation approaches to achieve higher fidelity solutions in a more tractable manner. Generalizing the decomposition techniques proposed in [14] and the order-reduction techniques common to PDE-constrained optimization provide promising avenues of research in this respect. This effort will be better enabled in part by our planned development of `InfiniteOpt.jl` to facilitate method of weighted residual transformations as discussed further below.

**Event-Constrained Optimization**

Event constraints present an intriguing modeling approach that warrants further application in general InfiniteOpt problems. We theorize that there exists a connection between posing the logical condition reformulations shown in Section 5.2.1 and disjunctive programming theory. A number of scalable solution approaches have been proposed for disjunctive programming as reviewed in [183]. We plan to explore this connection to model event constraints as disjunctive programs to enhance their intrinsic scalability.

**Generalized Risk Measures**

We have shown that risk measures from SO can be applied to InfiniteOpt problems over general domains. In future work, it will be interesting to investigate the analogy between SO and DO further in transferring more amenable measure operators and establishing their properties in a DO context (e.g., dominance and stability). Moreover, the establishment of time-valued pdfs provides a foundation from which the utility of transferring distributionally robust measure functions to DO can be investigated; which would potentially allow us to consider multiple weighting functions over the time horizon.

**Gray-Box Optimization**

Our proposed random field optimization approaches consider random field uncertainty (e.g., a known Gaussian process) that a system is subjected to. The recent work of Paulson and colleagues [184], shows that gray-box Bayesian optimization techniques can exploit random fields to model a process and accelerate the solution time. This presents an interesting application area for our unifying modeling abstraction that will allow us to embed problems that embed random field relations to model portions of a system's behavior. Another intriguing corollary will be to investigate InfiniteOpt problems that directly optimize the moments of random field models which might enable a new class of advanced experimental design approaches.

**Software-Enabled MWR Transformations**

In Chapter 2, we discussed how MWR transformations present a powerful framework for solving InfiniteOpt problems. However, such approaches have seen little adoption in

practice since implementing MWR transformations is very difficult and there is a lack of general software tools. Our general modeling approach in `InfiniteOpt.jl` coupled with the rich capabilities of the `Julia` ecosystem provide us with the means to implement a general MWR transformation tool. Such a tool will enable a number of interesting research directions and make these challenging approaches readily available for engineering applications.

**Software: Random Field Optimization**

The random field optimization formulations of Chapter 4 were modeled semi-manually using `InfiniteOpt.jl`, since it does not currently provide modeling objects to represent random field uncertainty and general stochastic differential equations. Thus, we plan to add infinite parameter function objects that can readily model random fields as described in [30]. Moreover, we plan to support stochastic integrals to characterize modeling equations that treat domains in a nonanticipative manner.

# Appendix A

SUPPLEMENTARY INFORMATION

This appendix presents supplementary information from select sections of this dissertation.

## A.1 Flexibility Measure Results

The results for all the instances discussed in Section 6.7.2 are summarized in Tables A.1 and A.2. Table A.1 features the use of sets $T_{box}(\delta)$ and $T_{ellip}(\delta)$ for all designs, covariances, and centers considered in the three-node network example. Table A.2 features the use of the sets $T_{box+}(\delta)$ and $T_{ellip+}(\delta)$ for the same system. Table A.3 provides the first 30 rank levels corresponding to the results discussed in Section 6.7.4 for the 141-node network.

Table A.1: Results for three-node distribution network using sets $T_{box}(\delta)$ and $T_{ellip}(\delta)$.

| Design | Center | $\beta$ | $F_{box}$ | Time (s) | $F_{ellip}$ | Time (s) | $\alpha^*$ (%) | $SF$-MC (%) |
|---|---|---|---|---|---|---|---|---|
| 1 | analytic | -40 | 0.578 | 0.03 | 15.31 | 0.18 | 99.84 | 99.99 |
| | | 0 | 0.578 | 0.03 | 8.93 | 0.21 | 96.97 | 99.71 |
| | | 50 | 0.578 | 0.03 | 4.31 | 0.18 | 77.02 | 96.22 |
| | feasible | -40 | 0.432 | 0.03 | 15.31 | 0.14 | 99.84 | 99.99 |
| | | 0 | 0.432 | 0.03 | 5.00 | 0.16 | 82.79 | 98.71 |
| | | 50 | 0.432 | 0.03 | 2.41 | 0.15 | 50.85 | 93.49 |
| 2 | analytic | -40 | 0.578 | 0.08 | 3.61 | 0.21 | 69.35 | 96.82 |
| | | 0 | 0.578 | 0.08 | 3.61 | 0.23 | 69.35 | 96.56 |
| | | 50 | 0.578 | 0.08 | 3.61 | 0.19 | 69.35 | 94.32 |
| | feasible | -40 | 0.432 | 0.03 | 3.61 | 0.15 | 69.35 | 96.82 |
| | | 0 | 0.432 | 0.03 | 3.61 | 0.21 | 69.35 | 96.00 |
| | | 50 | 0.432 | 0.03 | 2.41 | 0.20 | 50.85 | 92.67 |
| 3 | analytic | -40 | 0.578 | 0.04 | 37.81 | 0.20 | 100.00 | 100.00 |
| | | 0 | 0.578 | 0.04 | 8.93 | 0.16 | 96.97 | 99.72 |
| | | 50 | 0.578 | 0.04 | 4.31 | 0.16 | 77.02 | 96.20 |
| | feasible | -40 | 0.432 | 0.03 | 34.97 | 0.17 | 100.00 | 100.00 |
| | | 0 | 0.432 | 0.03 | 5.00 | 0.16 | 82.79 | 98.72 |
| | | 50 | 0.432 | 0.03 | 2.41 | 0.17 | 50.85 | 93.51 |

Table A.2: Results for three-node distribution network using sets $T_{box+}(\delta)$ and $T_{ellip+}(\delta)$.

| Design | Center | $\beta$ | $F_{box+}$ | Time (s) | $F_{ellip+}$ | Time (s) | $SF$-MC (%) |
|---|---|---|---|---|---|---|---|
| 1 | analytic | -40 | 0.578 | 0.04 | 18.38 | 0.14 | 100.00 |
| | | 0 | 0.578 | 0.04 | 8.93 | 0.10 | 99.44 |
| | | 50 | 0.578 | 0.04 | 4.31 | 0.13 | 94.33 |
| | feasible | -40 | 0.723 | 0.03 | 18.38 | 0.11 | 100.00 |
| | | 0 | 0.723 | 0.03 | 14.00 | 0.09 | 99.95 |
| | | 50 | 0.273 | 0.03 | 6.76 | 0.09 | 98.60 |
| 2 | analytic | -40 | 0.578 | 0.04 | 26.46 | 0.18 | 100.00 |
| | | 0 | 0.578 | 0.04 | 8.82 | 0.16 | 99.34 |
| | | 50 | 0.578 | 0.04 | 4.31 | 0.13 | 94.23 |
| | feasible | -40 | 0.723 | 0.03 | 26.46 | 0.12 | 100.00 |
| | | 0 | 0.723 | 0.03 | 14.00 | 0.15 | 99.96 |
| | | 50 | 0.723 | 0.03 | 6.76 | 0.12 | 98.60 |
| 3 | analytic | -40 | 0.578 | 0.03 | 45.38 | 0.20 | 100.00 |
| | | 0 | 0.578 | 0.03 | 8.93 | 0.11 | 99.45 |
| | | 50 | 0.578 | 0.03 | 4.31 | 0.06 | 94.34 |
| | feasible | -40 | 0.723 | 0.03 | 47.19 | 0.09 | 100.00 |
| | | 0 | 0.723 | 0.03 | 14.00 | 0.12 | 99.96 |
| | | 50 | 0.723 | 0.03 | 6.76 | 0.09 | 98.60 |

Table A.3: Limiting constraints for 141-node network (determined using $T_{ellip}(\delta)$).

| | Active Constraint Multipliers | $F_{ellip}$ | Increase (%) | Solution Time (s) |
|---|---|---|---|---|
| Rank 1 | $\lambda^{U}_{1:2}$, $\gamma^{L}_{1}$ | 0.298 | – | 0.19 |
| Rank 2 | $\lambda^{U}_{2:3}$, $\lambda^{U}_{4:5}$, $\lambda^{U}_{3:4}$ | 0.705 | 151.9 | 2.43 |
| Rank 3 | $\lambda^{U}_{5:6}$ | 1.110 | 273.0 | 4.35 |
| Rank 4 | $\lambda^{L}_{5:6}$ | 1.365 | 358.8 | 4.98 |
| Rank 5 | $\lambda^{L}_{4:5}$, $\lambda^{L}_{2:3}$, $\lambda^{L}_{3:4}$ | 1.767 | 493.8 | 6.00 |
| Rank 6 | $\lambda^{U}_{6:7}$ | 2.034 | 583.5 | 1.62 |
| Rank 7 | $\lambda^{L}_{6:7}$ | 2.223 | 646.9 | 2.20 |
| Rank 8 | $\lambda^{L}_{1:2}$ | 2.679 | 800.0 | 4.13 |
| Rank 9 | $\lambda^{U}_{6:37}$ | 2.770 | 830.8 | 5.89 |
| Rank 10 | $\lambda^{U}_{7:8}$ | 2.986 | 903.2 | 7.99 |
| Rank 11 | $\lambda^{L,U}_{37:38}$ | 3.030 | 918.1 | 11.45 |
| Rank 12 | $\lambda^{L}_{7:8}$ | 3.075 | 933.2 | 11.82 |
| Rank 13 | $\lambda^{L}_{6:37}$ | 3.117 | 947.3 | 13.04 |
| Rank 14 | $\lambda^{L,U}_{38:39}$, $\lambda^{L,U}_{8:9}$ | 3.125 | 949.9 | 25.67 |
| Rank 15 | $\lambda^{L,U}_{40:41}$, $\lambda^{L,U}_{39:40}$, $\lambda^{L,U}_{9:10}$ | 3.226 | 983.9 | 44.29 |
| Rank 16 | $\lambda^{L,U}_{41:42}$, $\lambda^{L,U}_{10:11}$ | 3.333 | 1020 | 18.95 |
| Rank 17 | $\lambda^{L,U}_{11:12}$ | 3.448 | 1058.6 | 3.86 |
| Rank 18 | $\lambda^{L,U}_{12:13}$ | 3.571 | 1099.9 | 4.64 |
| Rank 19 | $\lambda^{L,U}_{13:14}$ | 3.846 | 1192.3 | 4.95 |
| Rank 20 | $\lambda^{L,U}_{14:15}$ | 4.000 | 1243.9 | 5.08 |
| Rank 21 | $\lambda^{L,U}_{15:16}$, $\lambda^{L,U}_{42:43}$ | 6.666 | 2139.9 | 14.41 |
| Rank 22 | $\lambda^{L,U}_{43:44}$ | 7.143 | 2299.9 | 6.78 |
| Rank 23 | $\lambda^{U}_{7:88}$ | 7.579 | 2446.5 | 6.10 |
| Rank 24 | $\lambda^{L,U}_{16:17}$, $\lambda^{L,U}_{42:54}$, $\lambda^{L,U}_{54:55}$ | 7.692 | 2484.5 | 11.51 |
| Rank 25 | $\lambda^{L}_{7:88}$ | 7.806 | 2522.9 | 4.06 |
| Rank 26 | $\lambda^{L,U}_{17:18}$, $\lambda^{L,U}_{88:89}$ | 8.333 | 2699.9 | 9.83 |
| Rank 27 | $\lambda^{L,U}_{19:20}$, $\lambda^{L,U}_{18:19}$ | 9.091 | 2954.5 | 3.78 |
| Rank 28 | $\lambda^{L,U}_{20:21}$, $\lambda^{L,U}_{15:118}$, $\lambda^{L,U}_{118:119}$ | 10.000 | 3259.9 | 8.52 |
| Rank 29 | $\lambda^{L,U}_{21:22}$, $\lambda^{L,U}_{22:23}$, $\lambda^{L,U}_{44:45}$, $\lambda^{L,U}_{45:46}$ | 11.111 | 3633.3 | 10.65 |
| Rank 30 | $\lambda^{L,U}_{46:47}$, $\lambda^{L,U}_{55:60}$, $\lambda^{L,U}_{89:90}$, $\lambda^{L,U}_{90:91}$ | 12.500 | 4099.9 | 16.38 |

## A.2 Flexibility Design Numerical Results

The numerical results from Section 7.3 are provided below in Tables A.4, A.5, A.6, and A.7. Tables A.4 and A.5 feature the Pareto set data for the three-node network obtained with Problems (7.18) and (7.19). Table A.6 features the Pareto set data for the IEEE-14 node power network obtained with Problems (7.18) and (7.19). Table A.7 features the Pareto set data for the 141-node power network with obtained with Problems (7.18) and (7.19).

Table A.4: The first 22 results obtained for the three-node network using the mixed-integer and continuous formulations.

| $\epsilon_c$ | Design Cost | $SF_K$ (%) | $\bar{SF}_K$ (%) | Mixed-Integer Time (s) | Continuous Time (s) | Differences in $y^k$ (%) |
|---|---|---|---|---|---|---|
| 0 | 0 | 96.7 | 96.7 | 0.0155 | 0.0156 | 0 |
| 0.25 | 0.25 | 96.8 | 96.8 | 0.0170 | 0.0156 | 0 |
| 0.5 | 0.5 | 97.0 | 97.0 | 0.0171 | 0.0000 | 0 |
| 0.75 | 0.75 | 97.0 | 97.0 | 0.0179 | 0.0040 | 0 |
| 1 | 1 | 97.2 | 97.2 | 0.0189 | 0.0157 | 0 |
| 1.25 | 1.25 | 97.8 | 97.8 | 0.0190 | 0.0156 | 0 |
| 1.5 | 1.5 | 97.9 | 97.9 | 0.0173 | 0.0000 | 0 |
| 1.75 | 1.75 | 98.3 | 98.3 | 0.0195 | 0.0178 | 0 |
| 2 | 2 | 98.3 | 98.3 | 0.0187 | 0.0040 | 0 |
| 2.25 | 2.25 | 98.5 | 98.5 | 0.0168 | 0.0156 | 0 |
| 2.5 | 2.5 | 98.6 | 98.6 | 0.0191 | 0.0000 | 0 |
| 2.75 | 2.75 | 98.8 | 98.8 | 0.0175 | 0.0050 | 0 |
| 3 | 3 | 98.9 | 98.9 | 0.0208 | 0.0000 | 0 |
| 3.25 | 3.25 | 98.9 | 98.9 | 0.0203 | 0.0000 | 0 |
| 3.5 | 3.5 | 99.0 | 99.0 | 0.0193 | 0.0000 | 0 |
| 3.75 | 3.75 | 99.0 | 99.0 | 0.0180 | 0.0000 | 0 |
| 4 | 4 | 99.1 | 99.1 | 0.0172 | 0.0000 | 0 |
| 4.25 | 4.25 | 99.1 | 99.1 | 0.0200 | 0.0000 | 0 |
| 4.5 | 4.5 | 99.3 | 99.3 | 0.0186 | 0.0156 | 0 |
| 4.75 | 4.75 | 99.4 | 99.4 | 0.0192 | 0.0198 | 0 |
| 5 | 5 | 99.5 | 99.5 | 0.0186 | 0.0050 | 0 |
| 5.25 | 5.25 | 99.5 | 99.5 | 0.0182 | 0.0000 | 0 |

Table A.5: The last 21 results obtained for the three-node network using the mixed-integer and continuous formulations.

| $\epsilon_c$ | Design Cost | $SF_K$ (%) | $\bar{SF}_K$ (%) | Mixed-Integer Time (s) | Continuous Time (s) | Differences in $y^k$ (%) |
|---|---|---|---|---|---|---|
| 5.5 | 5.5 | 99.5 | 99.5 | 0.0183 | 0.0000 | 0 |
| 5.75 | 5.75 | 99.6 | 99.6 | 0.0193 | 0.0157 | 0 |
| 6 | 6 | 99.7 | 99.7 | 0.0177 | 0.0000 | 0 |
| 6.25 | 6.25 | 99.7 | 99.7 | 0.0187 | 0.0000 | 0 |
| 6.5 | 6.5 | 99.7 | 99.7 | 0.0202 | 0.0000 | 0 |
| 6.75 | 6.75 | 99.7 | 99.7 | 0.0179 | 0.0050 | 0 |
| 7 | 7 | 99.7 | 99.7 | 0.0185 | 0.0015 | 0 |
| 7.25 | 7.25 | 99.7 | 99.7 | 0.0207 | 0.0040 | 0 |
| 7.5 | 7.5 | 99.7 | 99.7 | 0.0196 | 0.0000 | 0 |
| 7.75 | 7.75 | 99.7 | 99.7 | 0.0182 | 0.0000 | 0 |
| 8 | 8 | 99.7 | 99.7 | 0.0183 | 0.0158 | 0 |
| 8.25 | 8.25 | 99.8 | 99.8 | 0.0195 | 0.0000 | 0 |
| 8.5 | 8.5 | 99.8 | 99.8 | 0.0192 | 0.0000 | 0 |
| 8.75 | 8.75 | 99.8 | 99.8 | 0.0192 | 0.0000 | 0 |
| 9 | 9 | 99.8 | 99.8 | 0.0207 | 0.0000 | 0 |
| 9.25 | 9.25 | 99.8 | 99.8 | 0.0192 | 0.0156 | 0 |
| 9.5 | 9.5 | 99.8 | 99.8 | 0.0182 | 0.0000 | 0 |
| 9.75 | 9.75 | 99.9 | 99.9 | 0.0180 | 0.0000 | 0 |
| 10 | 10 | 99.9 | 99.9 | 0.0202 | 0.0000 | 0 |
| 10.25 | 10.25 | 99.9 | 99.9 | 0.0201 | 0.0040 | 0 |
| 10.5 | 10.5 | 100 | 100 | 0.0195 | 0.0000 | 0 |

Table A.6: The results obtained for the IEEE-14 power network using the mixed-integer and continuous formulations.

| $\epsilon_c$ | Design Cost | $SF_K$ (%) | $\bar{SF}_K$ (%) | Mixed-Integer Time (s) | Continuous Time (s) | Differences in $y^k$ (%) |
|---|---|---|---|---|---|---|
| 0 | 0 | 88.10 | 88.10 | 16.20639 | 0.788219 | 0 |
| 2.5 | 2.5 | 88.30 | 88.30 | 47.31307 | 1.19029 | 0 |
| 5 | 5 | 88.95 | 88.95 | 36.28867 | 1.56477 | 0 |
| 7.5 | 7.5 | 89.20 | 89.20 | 51.17903 | 2.046263 | 0 |
| 10 | 10 | 89.50 | 89.50 | 54.72244 | 1.790163 | 0 |
| 12.5 | 12.5 | 89.80 | 89.80 | 84.38467 | 1.499423 | 0 |
| 15 | 15 | 90.15 | 90.15 | 94.45383 | 1.705751 | 0 |
| 17.5 | 17.5 | 90.30 | 90.30 | 71.09405 | 1.671522 | 0 |
| 20 | 20 | 90.55 | 90.55 | 49.37532 | 1.978597 | 0 |
| 22.5 | 22.5 | 90.70 | 90.70 | 72.26784 | 1.67794 | 0 |
| 25 | 25 | 90.80 | 90.80 | 36.79027 | 1.598156 | 0 |
| 27.5 | 27.5 | 90.95 | 90.95 | 34.87547 | 1.908839 | 0 |
| 30 | 30 | 91.05 | 91.05 | 45.3939 | 1.970099 | 0 |
| 32.5 | 32.5 | 91.20 | 91.20 | 57.60199 | 1.913504 | 0 |
| 35 | 35 | 91.20 | 91.20 | 50.41451 | 1.501976 | 0 |
| 37.5 | 37.5 | 91.20 | 91.20 | 25.38126 | 1.753275 | 0 |
| 40 | 40 | 91.25 | 91.25 | 20.42641 | 2.240765 | 0 |
| 42.5 | 42.5 | 91.30 | 91.30 | 19.73717 | 1.493643 | 0 |
| 45 | 45 | 91.30 | 91.30 | 22.286 | 1.579459 | 0 |
| 47.5 | 47.5 | 91.35 | 91.35 | 22.73862 | 1.853226 | 0 |
| 50 | 50 | 91.40 | 91.40 | 31.93482 | 2.758479 | 0 |
| 52.5 | 52.5 | 91.40 | 91.40 | 29.63871 | 1.616043 | 0 |
| 55 | 55 | 91.45 | 91.45 | 26.9819 | 1.736844 | 0 |
| 57.5 | 57.5 | 91.50 | 91.50 | 34.11424 | 1.60253 | 0 |
| 60 | 60 | 91.50 | 91.50 | 19.58831 | 2.56982 | 0 |
| 62.5 | 62.5 | 91.50 | 91.50 | 16.80612 | 1.569985 | 0 |
| 65 | 65 | 91.50 | 91.55 | 20.63473 | 1.492194 | 0.05 |

Table A.7: The results obtained for the 141-node power network using the mixed-integer and continuous formulations.

| $\epsilon_c$ | Design Cost | $SF_K$ (%) | $\bar{SF}_K$ (%) | MIP Time (s) | Continuous Time (s) | Differences in $y^k$ (%) | Optimal MIP |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 35.62 | 35.62 | 7.09 | 12.52 | 0 | Yes |
| 10 | 10 | 47.26 | 47.07 | 3605.73 | 13.66 | 0.95 | No |
| 20 | 20 | 51.14 | 50.84 | 3608.59 | 14.18 | 2.02 | No |
| 30 | 30 | 54.16 | 54.29 | 3614.13 | 13.77 | 1.95 | No |
| 40 | 40 | 56.61 | 56.86 | 3617.04 | 11.57 | 2.17 | No |
| 50 | 50 | 58.84 | 58.93 | 3622.68 | 13.46 | 1.79 | No |
| 60 | 60 | 60.67 | 60.53 | 3625.90 | 15.61 | 2.24 | No |
| 70 | 70 | 62.31 | 62.19 | 3625.88 | 15.21 | 2.12 | No |
| 80 | 80 | 63.50 | 63.50 | 3641.28 | 12.29 | 1.68 | No |
| 90 | 90 | 64.74 | 64.60 | 3618.84 | 15.34 | 1.42 | No |
| 100 | 100 | 65.71 | 65.54 | 3641.04 | 15.49 | 1.97 | No |
| 110 | 110 | 66.56 | 66.42 | 3624.48 | 12.30 | 2.1 | No |
| 120 | 120 | 67.27 | 67.18 | 3620.28 | 13.01 | 1.79 | No |
| 130 | 130 | 67.87 | 67.89 | 3628.07 | 15.14 | 1.3 | No |
| 140 | 140 | 68.41 | 68.29 | 3619.19 | 12.63 | 1.24 | No |
| 150 | 150 | 68.84 | 68.76 | 3619.67 | 15.29 | 1.2 | No |
| 160 | 160 | 69.17 | 69.18 | 3635.72 | 12.64 | 1.13 | No |
| 170 | 170 | 69.43 | 69.52 | 3624.00 | 13.38 | 1.01 | No |
| 180 | 180 | 69.69 | 69.81 | 3623.32 | 14.87 | 0.76 | No |
| 190 | 190 | 69.91 | 70.10 | 3623.43 | 11.80 | 0.59 | No |
| 200 | 200 | 70.11 | 70.28 | 804.34 | 11.51 | 0.49 | Yes |
| 210 | 210 | 70.24 | 70.43 | 193.87 | 13.10 | 0.47 | Yes |
| 220 | 220 | 70.34 | 70.50 | 160.40 | 12.64 | 0.5 | Yes |
| 230 | 230 | 70.41 | 70.59 | 165.22 | 12.67 | 0.5 | Yes |
| 240 | 240 | 70.48 | 70.67 | 124.44 | 12.42 | 0.49 | Yes |
| 250 | 250 | 70.53 | 70.76 | 111.72 | 12.45 | 0.47 | Yes |
| 260 | 260 | 70.57 | 70.64 | 108.32 | 14.97 | 0.63 | Yes |
| 270 | 270 | 70.57 | 70.69 | 103.03 | 13.21 | 0.64 | Yes |
| 280 | 280 | 70.62 | 70.76 | 101.47 | 13.25 | 0.56 | Yes |
| 290 | 290 | 70.64 | 70.78 | 101.27 | 13.55 | 0.54 | Yes |
| 300 | 300 | 70.64 | 70.78 | 99.66 | 11.79 | 0.54 | Yes |

## A.3 Quality of Relaxation for Simple Setting

**Theorem 5.** *The relaxation of problem* (8.11) *is exact.*

*Proof.* We express problem (8.11) (denoted as $P$) in vector form as:

$$\begin{aligned} \psi(A,\xi) = \quad & \max_{y,z} \quad (1-y) \\ & \text{s.t.} \quad A(\xi)z = \hat{d}\cdot(1-y) \\ & \qquad\quad z \geq 0 \\ & \qquad\quad y \in \{0,1\} \end{aligned}$$

where we define $\hat{d} := -d$ to express the constraints in a standard linear form. The relaxed problem (denoted as $\bar{P}$) is obtained by replacing $y \in \{0,1\}$ with $\bar{y} \in [0,1]$. We denote optimal solutions for $\bar{P}$ and $P$ as $\bar{y}^*$ and $y^*$, respectively. We show that $\bar{P}$ delivers optimal solutions for $P$ by analyzing three possible cases. First consider the case in which $\hat{d} \in \mathcal{R}(A(\xi))$ (where $\mathcal{R}(\cdot)$ denotes the range of the input matrix) and there exists a nontrivial flow solution ($z_j^* > 0$ for some $j$) such that $A(\xi)z^* = \hat{d}$. This implies that all values of $y$ are feasible since $(1-y)\hat{d} \in \mathcal{R}(A(\xi))$, $\forall y \in \mathbb{R}$. Thus, $y^* = \bar{y}^* = 0$ must be optimal solutions (yielding the largest possible objective), since any other feasible value of $y$ would have a lower objective. The second case is that in which $\hat{d} \in \mathcal{R}(A(\xi))$ and there does not exist a nontrivial solution ($z_j^* > 0$ for some $j$) such that $A(\xi)z^* = \hat{d}$. In this case the only feasible solution is the trivial solution $z^* = 0$ and thus $y^* = \bar{y}^* = 1$. The third and final case corresponds to $\hat{d} \notin \mathcal{R}(A(\xi))$; it follows that $(1-y)\hat{d} \in \mathcal{R}(A(\xi))$ if and only if $y = 1$ since any scalar multiple of $\hat{d}$ will lie outside of $\mathcal{R}(A(\xi))$ except for the trivial case that $\hat{d} = 0$. Thus, the only feasible (and therefore optimal) solution to both problems is $y^* = \bar{y}^* = 1$. □

# BIBLIOGRAPHY

[1] Joshua L Pulsipher and Victor M Zavala. A scalable stochastic programming approach for the design of flexible systems. *Computers & Chemical Engineering*, 128:69–76, 2019.

[2] Olivier Devolder, François Glineur, and Yurii Nesterov. Solving infinite-dimensional optimization problems by polynomial approximation. In *Recent Advances in Optimization and its Applications in Engineering*, pages 31–40. Springer, 2010.

[3] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[4] Lorenz T Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46(11):1043–1053, 2007.

[5] Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media, 2008.

[6] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[7] Oliver Stein and Georg Still. Solving semi-infinite optimization problems with interior point techniques. *SIAM Journal on Control and Optimization*, 42(3):769–788, 2003.

[8] James B Rawlings. Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52, 2000.

[9] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[10] Andrzej I Stankiewicz, Jacob A Moulijn, et al. Process intensification: transforming chemical engineering. *Chemical engineering progress*, 96(1):22–34, 2000.

[11] David A Straub and Ignacio E Grossmann. Design optimization of stochastic flexibility. *Computers & Chemical Engineering*, 17(4):339–354, 1993.

[12] Ulas Çakmak and Süleyman Özekici. Portfolio optimization in stochastic markets. *Mathematical Methods of Operations Research*, 63(1):151–168, 2006.

[13] Darinka Dentcheva and Andrzej Ruszczyński. Portfolio optimization with stochastic dominance constraints. *Journal of Banking & Finance*, 30(2):433–451, 2006.

[14] Sungho Shin, Ophelia S Venturelli, and Victor M Zavala. Scalable nonlinear programming framework for parameter estimation in dynamic biological system models. *PLoS computational biology*, 15(3):e1006828, 2019.

[15] Lorenz T Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders. Large-scale pde-constrained optimization: an introduction. In *Large-Scale PDE-Constrained Optimization*, pages 3–13. Springer, 2003.

[16] Joshua L Pulsipher and Victor M Zavala. Measuring and optimizing system reliability: a stochastic programming approach. *TOP*, pages 1–20, 2020.

[17] Arthriya Suksuwan and Seymour MJ Spence. Optimization of uncertain structures subject to stochastic wind loads under system-level first excursion constraints: A data-driven approach. *Computers & Structures*, 210:58–68, 2018.

[18] Line Roald, Sidhant Misra, Michael Chertkov, and Göran Andersson. Optimal power flow with weighted chance constraints and general policies for generation

control. In *2015 54th IEEE conference on decision and control (CDC)*, pages 6927–6933. IEEE, 2015.

[19] Tian Lan, Zhangxin Zhou, and Garng M Huang. Modeling and numerical analysis of stochastic optimal transmission switching with dcopf and acopf. *IFAC-PapersOnLine*, 51(28):126–131, 2018.

[20] Christos Georgakis. Design of dynamic experiments: A data-driven methodology for the optimization of time-varying processes. *Industrial & Engineering Chemistry Research*, 52(35):12369–12382, 2013.

[21] SP Asprey and S Macchietto. Designing robust optimal dynamic experiments. *Journal of Process Control*, 12(4):545–556, 2002.

[22] Bethany Nicholson, John D Siirola, Jean-Paul Watson, Victor M Zavala, and Lorenz T Biegler. pyomo. dae: A modeling and automatic discretization framework for optimization with differential and algebraic equations. *Mathematical Programming Computation*, 10(2):187–223, 2018.

[23] Robert J Vanderbei. *Linear programming: foundations and extensions*, volume 285. Springer Nature, 2020.

[24] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[25] Michael Chen, Sanjay Mehrotra, and Dávid Papp. Scenario generation for stochastic optimization problems via the sparse grid method. *Computational Optimization and applications*, 62(3):669–692, 2015.

[26] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[27] Matti Koivu and Teemu Pennanen. Galerkin methods in dynamic stochastic programming. *Optimization*, 59(3):339–354, 2010.

[28] TH Tsang, DM Himmelblau, and Thomas F Edgar. Optimal control via collocation and non-linear programming. *International Journal of Control*, 21(5):763–768, 1975.

[29] Timm Faulwasser and Lars Grüne. Turnpike properties in optimal control: An overview of discrete-time and continuous-time results. *arXiv preprint arXiv:2011.13670*, 2020.

[30] Joshua L Pulsipher, Weiqi Zhang, Tyler J Hongisto, and Victor M Zavala. A unifying modeling abstraction for infinite-dimensional optimization. *Computers & Chemical Engineering*, 156, 2022.

[31] Logan DR Beal, Daniel C Hill, R Abraham Martin, and John D Hedengren. Gekko optimization suite. *Processes*, 6(8):106, 2018.

[32] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit - an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[33] Mariano Asteasuain, Stella Maris Tonelli, Adriana Brandolin, and Jose Alberto Bandoni. Dynamic simulation and optimisation of tubular polymerisation reactors in gproms. *Computers & Chemical Engineering*, 25(4-6):509–515, 2001.

[34] Jordan Jalving, Yankai Cao, and Victor M Zavala. Graph-based modeling and simulation of complex systems. *Computers & Chemical Engineering*, 125:134–154, 2019.

[35] Jordan Jalving, Sungho Shin, and Victor M Zavala. A graph-based modeling abstraction for optimization: Concepts and implementation in plasmo. jl. *arXiv preprint arXiv:2006.05378*, 2020.

[36] Sungho Shin, Mihai Anitescu, and Victor M Zavala. Exponential decay of sensitivity in graph-structured nonlinear programs. *arXiv preprint arXiv:2101.03067*, 2021.

[37] Ross Edward Swaney and Ignacio E Grossmann. An index for operational flexibility in chemical process design. part i: Formulation and theory. *AIChE Journal*, 31(4):621–630, 1985.

[38] Babatunde A Ogunnaike. *Random phenomena: fundamentals of probability and statistics for engineers*. CRC Press, 2009.

[39] EN Pistikopoulos and MG Ierapetritou. Novel approach for optimal process design under uncertainty. *Computers & Chemical Engineering*, 19(10):1089–1110, 1995.

[40] Robin Smith. *Chemical process: design and integration*. John Wiley & Sons, 2005.

[41] George Papaefthymiou and Dorota Kurowicka. Using copulas for modeling stochastic dependence in power system uncertainty analysis. *IEEE Transactions on Power Systems*, 24(1):40–49, 2009.

[42] Andreas Ulbig and Göran Andersson. Analyzing operational flexibility of electric power systems. *International Journal of Electrical Power & Energy Systems*, 72:155–164, 2015.

[43] Myungsoo Jun and Raffaello D'Andrea. Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In *Cooperative control: models, applications and algorithms*, pages 95–110. Springer, 2003.

[44] Junmin Wang, Joe Steiber, and Bapiraju Surampudi. Autonomous ground vehicle control system for high-speed and safe operation. In *2008 American Control Conference*, pages 218–223. IEEE, 2008.

[45] Daniel A Crowl and Joseph F Louvar. *Chemical process safety: fundamentals with applications*. Pearson Education, 2001.

[46] Ignacio E Grossmann, Bruno A Calfa, and Pablo Garcia-Herreros. Evolution of concepts and models for quantifying resiliency and flexibility of chemical processes. *Computers & Chemical Engineering*, 70:22–34, 2014.

[47] David A Straub and Ignacio E Grossmann. Integrated stochastic metric of flexibility for systems with discrete state and continuous parameter uncertainties. *Computers & Chemical Engineering*, 14(9):967–985, 1990.

[48] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[49] Alexander Shapiro. Sample average approximation. In *Encyclopedia of Operations Research and Management Science*, pages 1350–1355. Springer, 2013.

[50] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical algorithms*, 18(3-4):209, 1998.

[51] Qi Zhang, Ignacio E Grossmann, and Ricardo M Lima. On the relation between flexibility analysis and robust optimization for linear systems. *AIChE Journal*, 62(9):3109–3123, 2016.

[52] Joshua L Pulsipher and Victor M Zavala. A mixed-integer conic programming formulation for computing the flexibility index under multivariate gaussian uncertainty. *Computers & Chemical Engineering*, 119:302–308, 2018.

[53] TV Thomaidis and EN Pistikopoulos. Integration of flexibility, reliability and maintenance in process synthesis and design. *Computers & chemical engineering*, 18:S259–S263, 1994.

[54] Yixin Ye, Ignacio E Grossmann, and Jose M Pinto. Mixed-integer nonlinear programming models for optimal design of reliable chemical plants. *Computers & Chemical Engineering*, 116:3–16, 2018.

[55] Fathollah Bistouni and Mohsen Jahanshahi. Analyzing the reliability of shuffle-exchange networks using reliability block diagrams. *Reliability Engineering & System Safety*, 132:97–106, 2014.

[56] Wenyuan Li et al. *Reliability assessment of electric power systems using Monte Carlo methods*. Springer Science & Business Media, 2013.

[57] James Luedtke and Shabbir Ahmed. A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699, 2008.

[58] Sungho Shin and Victor M Zavala. Diffusing-horizon model predictive control. *arXiv preprint arXiv:2002.08556*, 2020.

[59] Sen Na, Sungho Shin, Mihai Anitescu, and Victor M Zavala. Overlapping schwarz decomposition for nonlinear optimal control. *arXiv preprint arXiv:2005.06674*, 2020.

[60] Dongbin Xiu. *Numerical methods for stochastic computations: a spectral method approach*. Princeton university press, 2010.

[61] Antonios Armaou and Panagiotis D Christofides. Dynamic optimization of dissipative pde systems using nonlinear order reduction. *Chemical engineering science*, 57(24):5083–5114, 2002.

[62] Fabian Gnegel, Armin Fügenschuh, Michael Hagel, Sven Leyffer, and Marcus Stiemer. A solution framework for linear pde-constrained mixed-integer problems. *Mathematical Programming*, pages 1–34, 2021.

[63] Michael D Graham and James Blake Rawlings. *Modeling and analysis principles for chemical and biological engineers*. Nob Hill Pub., 2013.

[64] Steve Zymler, Daniel Kuhn, and Berç Rustem. Distributionally robust joint chance constraints with second-order moment information. *Mathematical Programming*, 137(1-2):167–198, 2013.

[65] Bruce A Finlayson. *The method of weighted residuals and variational principles*. SIAM, 2013.

[66] Tillmann Mühlpfordt, Line Roald, Veit Hagenmeyer, Timm Faulwasser, and Sidhant Misra. Chance-constrained ac optimal power flow: A polynomial chaos approach. *IEEE Transactions on Power Systems*, 34(6):4806–4816, 2019.

[67] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[68] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[69] Jorge Nocedal. Knitro: an integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization*, pages 35–60. Springer, 2006.

[70] Mathieu Besançon, David Anthoff, Alex Arslan, Simon Byrne, Dahua Lin, Theodore Papamarkou, and John Pearson. Distributions.jl: Definition and modeling of probability distributions in the juliastats ecosystem. *arXiv preprint arXiv:1907.08611*, 2019.

[71] Benoit Legat, Oscar Dowson, Joaquim Dias Garcia, and Miles Lubin. Mathoptinterface: a data structure for mathematical optimization problems. *arXiv preprint arXiv:2002.03447*, 2020.

[72] Thomas Keuter, Norbert Heribert Menzler, Georg Mauer, Frank Vondahlen, Robert Vaßen, and Hans Peter Buchkremer. Modeling precursor diffusion and reaction of atomic layer deposition in porous structures. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 33(1):01A104, 2015.

[73] Legesse Lemecha Obsu and Shiferaw Feyissa Balcha. Optimal control strategies for the transmission risk of covid-19. *Journal of biological dynamics*, 14(1):590–607, 2020.

[74] Ivan Area, FaÏÇal NdaÏrou, Juan J Nieto, Cristiana J Silva, and Delfim FM Torres. Ebola model and optimal control with vaccination constraints. *arXiv preprint arXiv:1703.01368*, 2017.

[75] Calvin Tsay, Fernando Lejarza, Mark A Stadtherr, and Michael Baldea. Modeling, state estimation, and optimal control for the us covid-19 outbreak. *arXiv preprint arXiv:2004.06291*, 2020.

[76] Joan L Aron and Ira B Schwartz. Seasonality and period-doubling bifurcations in an epidemic model. *Journal of theoretical biology*, 110(4):665–679, 1984.

[77] Joshua L Pulsipher, Benjamin R Davidson, and Victor M Zavala. New measures for shaping trajectories in dynamic optimization. *arXiv preprint arXiv:2110.07041*, 2021.

[78] Michael J Risbeck and James B Rawlings. Economic model predictive control for time-varying cost and peak demand charge optimization. *IEEE Transactions on Automatic Control*, 2019.

[79] Andrzej Ruszczyński and Alexander Shapiro. Optimization of risk measures. In *Probabilistic and randomized methods for design under uncertainty*, pages 119–157. Springer, 2006.

[80] James Stewart. *Calculus: Concepts and contexts*. Cengage Learning, 2009.

[81] Marek Petrik and Bruno Scherrer. Biasing approximate dynamic programming with a lower discount factor. In *Twenty-Second Annual Conference on Neural Information Processing Systems-NIPS 2008*, 2008.

[82] Pavlo Krokhmal, Michael Zabarankin, and Stan Uryasev. Modeling and optimization of risk. *Handbook of the fundamentals of financial decision making: Part II*, pages 555–600, 2013.

[83] Hayne E Leland. Beyond mean–variance: Performance measurement in a nonsymmetrical world (corrected). *Financial analysts journal*, 55(1):27–36, 1999.

[84] Sergey Sarykalin, Gaia Serraino, and Stan Uryasev. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In *State-of-the-art decision-making tools in the information-intensive age*, pages 270–294. Informs, 2008.

[85] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[86] Cristinca Fulga. Portfolio optimization with disutility-based risk measure. *European Journal of Operational Research*, 251(2):541–553, 2016.

[87] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[88] Jose A Renteria, Yankai Cao, Alexander W Dowling, and Victor M Zavala. Optimal pid controller tuning using stochastic programming techniques. *AIChE Journal*, 64(8):2997–3010, 2018.

[89] James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[90] Alexander W Dowling, Gerardo Ruiz-Mercado, and Victor M Zavala. A framework for multi-stakeholder decision-making and conflict resolution. *Computers & Chemical Engineering*, 90:136–150, 2016.

[91] Joshua L Pulsipher, Benjamin R Davidson, and Victor M Zavala. Random field optimization. Under review. 2022.

[92] Michael Baldea and Thomas F Edgar. Dynamic process intensification. *Current opinion in chemical engineering*, 22:48–53, 2018.

[93] Ali Ahmadzadeh, Nader Motee, Ali Jadbabaie, and George Pappas. Multi-vehicle path planning in dynamically changing environments. In *2009 IEEE international conference on Robotics and Automation*, pages 2449–2454. IEEE, 2009.

[94] Navid Dadkhah and Berenice Mettler. Survey of motion planning literature in the presence of uncertainty: Considerations for uav guidance. *Journal of Intelligent & Robotic Systems*, 65(1):233–246, 2012.

[95] Veniamin D Dimitriadis and Efstratios N Pistikopoulos. Flexibility analysis of dynamic systems. *Industrial & Engineering Chemistry Research*, 34(12):4451–4462, 1995.

[96] Qiugang Lu and Victor M Zavala. Image-based model predictive control via dynamic mode decomposition. *Journal of Process Control*, 104:146–157, 2021.

[97] Shikui Chen, Wei Chen, and Sanghoon Lee. Level set based robust shape and topology optimization under random field uncertainties. *Structural and Multidisciplinary Optimization*, 41(4):507–524, 2010.

[98] Dennis C Rapaport and Dennis C Rapaport Rapaport. *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[99] Tobias Neckel and Florian Rupp. *Random differential equations in scientific computing*. De Gruyter Open Poland, 2013.

[100] Sheldon M Ross, John J Kelly, Roger J Sullivan, William James Perry, Donald Mercer, Ruth M Davis, Thomas Dell Washburn, Earl V Sager, Joseph B Boyce, and Vincent L Bristow. *Stochastic processes*, volume 2. Wiley New York, 1996.

[101] Nicolaas Godfried Van Kampen. *Stochastic processes in physics and chemistry*, volume 1. Elsevier, 1992.

[102] Linda JS Allen. *An introduction to stochastic processes with applications to biology*. CRC press, 2010.

[103] Carlo Laing and Gabriel J Lord. *Stochastic methods in neuroscience*. Oxford University Press, 2010.

[104] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-6(6):721–741, 1984.

[105] Tsu T Soong and SOONG TT. *Random differential equations in science and engineering*. Academic Press, Inc., 1973.

[106] Philip E Protter. Stochastic differential equations. In *Stochastic integration and differential equations*, pages 249–361. Springer, 2005.

[107] Christopher Rackauckas and Qing Nie. Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, 5(1), 2017.

[108] Umberto Picchini. Sde toolbox: Simulation and estimation of stochastic differential equations with matlab., 2007.

[109] Mikhail D Kuznetsov and Dmitriy F Kuznetsov. Sde-math: A software package for the implementation of strong high-order numerical methods for itô sdes with multidimensional non-commutative noise based on multiple fourier legendre series. *Differential Equations & Control Processes*, (1), 2021.

[110] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

[111] Alexander Shapiro and Arkadi Nemirovski. On complexity of stochastic programming problems. In *Continuous optimization*, pages 111–146. Springer, 2005.

[112] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[113] Oscar Dowson and Lea Kapelevich. Sddp. jl: a julia package for stochastic dual dynamic programming. *INFORMS Journal on Computing*, 33(1):27–33, 2021.

[114] Robert J Adler, Jonathan E Taylor, et al. *Random fields and geometry*, volume 80. Springer, 2007.

[115] Matthew Brett, Will Penny, and Stefan Kiebel. Introduction to random field theory. *Human brain function*, 2:867–879, 2003.

[116] Chaohui Wang, Nikos Komodakis, and Nikos Paragios. Markov random field modeling, inference & learning in computer vision & image understanding: A survey. *Computer Vision and Image Understanding*, 117(11):1610–1627, 2013.

[117] George Christakos. *Random field models in earth sciences*. Courier Corporation, 2012.

[118] JW Harbaugh and FW Preston. Fourier analysis in geology. *Spatial analysis: A reader in statistical geography*, pages 218–38, 1968.

[119] Robert J Adler. Some new random field tools for spatial analysis. *Stochastic Environmental Research and Risk Assessment*, 22(6):809–822, 2008.

[120] George Christakos. *Spatiotemporal random fields: theory and applications*. Elsevier, 2017.

[121] Moo K Chung. Introduction to random fields. *arXiv preprint arXiv:2007.09660*, 2020.

[122] Fei Kang, Bin Xu, Junjie Li, and Sizeng Zhao. Slope stability evaluation using gaussian processes with various covariance functions. *Applied Soft Computing*, 60:387–396, 2017.

[123] Yang Liu, Jingfa Li, Shuyu Sun, and Bo Yu. Advances in gaussian random field generation: a review. *Computational Geosciences*, 23(5):1011–1047, 2019.

[124] Keith J Worsley. Local maxima and the expected euler characteristic of excursion sets of $\chi$ 2, f and t fields. *Advances in Applied Probability*, 26(1):13–42, 1994.

[125] Yuri Rozanov. *Random fields and stochastic partial differential equations*, volume 438. Springer Science & Business Media, 2013.

[126] Julie Fournier. *Geometric characteristics of smooth anisotropic random fields*. PhD thesis, Sorbonne Université; Université Paris 5 René Descartes, 2018.

[127] Malcolm R Leadbetter, Georg Lindgren, and Holger Rootzén. *Extremes and related properties of random sequences and processes*. Springer Science & Business Media, 2012.

[128] Alexander Smith and Victor M. Zavala. The euler characteristic: A general topological descriptor for complex data. *Computers & Chemical Engineering*, 154:107463, 2021.

[129] Robert J Adler, Jose H Blanchet, and Jingchen Liu. Efficient monte carlo for high excursions of gaussian random fields. *The Annals of Applied Probability*, 22(3):1167–1214, 2012.

[130] Erhan Cinlar. *Introduction to stochastic processes*. Courier Corporation, 2013.

[131] Renate Winkler. Stochastic differential algebraic equations in transient noise analysis. In *Scientific computing in electrical engineering*, pages 151–156. Springer, 2006.

[132] Sumit Suthar and Soumyendu Raha. On explicit stochastic differential algebraic equations. *arXiv preprint arXiv:2102.00605*, 2021.

[133] Kiyosi Ito. *Foundations of stochastic differential equations in infinite dimensional spaces*. SIAM, 1984.

[134] RL196814 Stratonovich. A new representation for stochastic integrals and equations. *SIAM Journal on Control*, 4(2):362–371, 1966.

[135] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[136] Zhongqiang Zhang and George Karniadakis. *Numerical methods for stochastic partial differential equations with white noise*. Springer, 2017.

[137] Maha Youssef and Roland Pulch. Poly-sinc solution of stochastic elliptic differential equations. *Journal of Scientific Computing*, 87(3):1–19, 2021.

[138] Sungho Shin, Victor M Zavala, and Mihai Anitescu. Decentralized schemes with overlap for solving graph-structured optimization problems. *IEEE Transactions on Control of Network Systems*, 7(3):1225–1236, 2020.

[139] SK Au and James L Beck. First excursion probabilities for linear systems by very efficient importance sampling. *Probabilistic engineering mechanics*, 16(3):193–207, 2001.

[140] Tillmann Mühlpfordt, Timm Faulwasser, and Veit Hagenmeyer. A generalized framework for chance-constrained optimal power flow. *Sustainable Energy, Grids and Networks*, 16:231–242, 2018.

[141] Ophelia S Venturelli, Alex V Carr, Garth Fisher, Ryan H Hsu, Rebecca Lau, Benjamin P Bowen, Susan Hromada, Trent Northen, and Adam P Arkin. Deciphering microbial interactions in synthetic human gut microbiome communities. *Molecular systems biology*, 14(6):e8157, 2018.

[142] Iauw Bhieng Tjoa and Lorenz T Biegler. Simultaneous solution and optimization strategies for parameter estimation of differential-algebraic equation systems. *Industrial & Engineering Chemistry Research*, 30(2):376–385, 1991.

[143] Jim O Ramsay, Giles Hooker, David Campbell, and Jiguo Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[144] Joshua L Pulsipher, Daniel Rios, and Victor M Zavala. A computational framework for quantifying and analyzing system flexibility. *Computers & Chemical Engineering*, 2019.

[145] Ross E Swaney and Ignacio E Grossmann. An index for operational flexibility in chemical process design. part ii: Computational algorithms. *AIChE Journal*, 31(4):631–641, 1985.

[146] Ignacio E Grossmann, Keshava Prasad Halemane, and Ross E Swaney. Optimization strategies for flexible chemical processes. *Computers & Chemical Engineering*, 7(4):439–462, 1983.

[147] Ignacio E Grossmann and Christodoulos A Floudas. Active constraint strategy for flexibility analysis in chemical processes. *Computers & Chemical Engineering*, 11(6):675–693, 1987.

[148] William C Rooney and Lorenz T Biegler. Incorporating joint confidence regions into design under uncertainty. *Computers & Chemical Engineering*, 23(10):1563–1575, 1999.

[149] EN Pistikopoulos and TA Mazzuchi. A novel flexibility analysis approach for processes with stochastic parameters. *Computers & Chemical Engineering*, 14(9):991–1000, 1990.

[150] J Ma, V Rokhlin, and Stephen Wandzura. Generalized gaussian quadrature rules for systems of arbitrary functions. *SIAM Journal on Numerical Analysis*, 33(3):971–996, 1996.

[151] Elijah Polak and Alberto Sangiovanni-Vincentelli. Theoretical and computational aspects of the optimal design centering, tolerancing, and tuning problem. *IEEE Transactions on Circuits and Systems*, 26(9):795–813, 1979.

[152] Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a cip framework to solve miqcps. In *Mixed integer nonlinear programming*, pages 427–444. Springer, 2012.

[153] Chi-Squared Distribution. *National Institute of Standards and Technology (NIST) Engineering Statistics Handbook available at* https://www.itl.nist.gov/div898/handbook/eda/section3/eda3666.htm. [Online; accessed 31-May-2018].

[154] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[155] Zukui Li, Ran Ding, and Christodoulos A Floudas. A comparative theoretical and computational study on robust counterpart optimization: I. robust linear optimiza-

tion and robust mixed integer linear optimization. *Industrial & engineering chemistry research*, 50(18):10567–10603, 2011.

[156] Konstantin Pavlikov and Stan Uryasev. Cvar norm and applications in optimization. *Optimization Letters*, 8(7):1999–2020, 2014.

[157] Jun-ya Gotoh and Stan Uryasev. Two pairs of families of polyhedral norms versus $\ell_p$-norms: proximity and applications in optimization. *Mathematical Programming*, 156(1-2):391–431, 2016.

[158] Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015.

[159] Vikrant Bansal, John D Perkins, and Efstratios N Pistikopoulos. Flexibility analysis and design of linear systems by parametric programming. *AIChE Journal*, 46(2):335–354, 2000.

[160] Ignacio E Grossmann and Roger WH Sargent. Optimum design of multipurpose chemical plants. *Industrial & Engineering Chemistry Process Design and Development*, 18(2):343–348, 1979.

[161] Efstratios N Pistikopoulos and Ignacio E Grossmann. Optimal retrofit design for improving process flexibility in linear systems. *Computers & Chemical Engineering*, 12(7):719–731, 1988.

[162] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, Robert John Thomas, et al. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2011.

[163] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[164] Victor M Zavala, Kibaek Kim, Mihai Anitescu, and John Birge. A stochastic electricity market clearing formulation with consistent pricing properties. *Operations Research*, 65(3):557–576, 2017.

[165] Iraj Dabbagchi. Ieee 14 bus power flow test case. *American Electric Power System Golden CO*, pages 557–576, 1962.

[166] HM Khodr, FG Olsina, PM De Oliveira-De Jesus, and JM Yusta. Maximum savings approach for location and sizing of capacitors in distribution systems. *Electric Power Systems Research*, 78(7):1192–1203, 2008.

[167] Dimitrios K Varvarezos, Lorenz T Biegler, and Ignacio E Grossmann. Multiperiod design optimization with sqp decomposition. *Computers & chemical engineering*, 18(7):579–595, 1994.

[168] Efstratios N Pistikopoulos and Ignacio E Grossmann. Optimal retrofit design for improving process flexibility in nonlinear systems—i. fixed degree of flexibility. *Computers & chemical engineering*, 13(9):1003–1016, 1989.

[169] David W Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[170] Javiera Barrera, Tito Homem-de Mello, Eduardo Moreno, Bernardo K Pagnoncelli, and Gianpiero Canessa. Chance-constrained problems and rare events: an importance sampling approach. *Mathematical Programming*, 157(1):153–189, 2016.

[171] Frank E Curtis, Andreas Wachter, and Victor M Zavala. A sequential algorithm for solving nonlinear optimization problems with chance constraints. *SIAM Journal on Optimization*, 28(1):930–958, 2018.

[172] Shabbir Ahmed, James Luedtke, Yongjia Song, and Weijun Xie. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. *Mathematical Programming*, 162(1-2):51–81, 2017.

[173] Yongjia Song, James R Luedtke, and Simge Küçükyavuz. Chance-constrained binary packing problems. *INFORMS Journal on Computing*, 26(4):735–747, 2014.

[174] Feng Qiu, Shabbir Ahmed, Santanu S Dey, and Laurence A Wolsey. Covering linear programming with violations. *INFORMS Journal on Computing*, 26(3):531–546, 2014.

[175] Youngsuk Kim and Won-Hee Kang. Network reliability analysis of complex systems using a non-simulation-based method. *Reliability engineering & system safety*, 110:80–88, 2013.

[176] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on cyber security for smart grid communications. *IEEE Communications Surveys & Tutorials*, 14(4):998–1010, 2012.

[177] V Bansal, JD Perkins, and EN Pistikopoulos. Flexibility analysis and design of dynamic processes with stochastic parameters. *Computers & chemical engineering*, 22:S817–S820, 1998.

[178] Ross Edward Swaney and Ignacio E Grossmann. An index for operational flexibility in chemical process design. part i: Formulation and theory. *AIChE Journal*, 31(4):621–630, 1985.

[179] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. A guide to sample average approximation. In *Handbook of simulation optimization*, pages 207–243. Springer, 2015.

[180] Pao-Lu Hsu and Herbert Robbins. Complete convergence and the law of large numbers. *Proceedings of the National Academy of Sciences of the United States of America*, 33(2):25, 1947.

[181] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems*, 26(1):12–19, 2010.

[182] Daniel Casas-Orozco, Daniel Laky, Vivian Wang, Mesfin Abdi, X Feng, E Wood, Carl Laird, Gintaras V Reklaitis, and Zoltan K Nagy. Pharmapy: an object-oriented tool for the development of hybrid pharmaceutical flowsheets. *Computers & Chemical Engineering*, page 107408, 2021.

[183] Francisco Trespalacios and Ignacio E Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014.

[184] Joel A Paulson and Congwen Lu. Cobalt: Constrained bayesian optimization of computationally expensive grey-box models exploiting derivative information. *arXiv preprint arXiv:2105.04114*, 2021.