Reliable Controller Synthesis: Guarantees for Safety-Critical System Testing and Verification

Thesis by
Prithvi Akella

In Partial Fulfillment of the Requirements for the
Degree of
Doctorate of Philosophy in Mechanical Engineering

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2023
Defended June 8, 2023

Prithvi Akella
ORCID: 0000-0003-4375-0015

ACKNOWLEDGEMENTS

To properly acknowledge everyone whom I would like to thank would likely amount to another thesis in its entirety, so instead I'll say this: To everyone I met on my Caltech journey, thank you. In whatever capacity you joined me on my journey, our time together has left a lasting impact that I am sure I will remember for years to come. Wherever our paths lead us in the future, should they cross again, I hope we never become strangers, and I would be delighted to hear the stories of your travels.

That being said, there are some individuals I would like to note. Firstly, I would like to thank all my mentors, wherein I would be remiss if I didn't first mention my advisor, Dr. Aaron Ames. While I may have grumbled every so often about the incessant deadlines and breakneck research pace our group has become synonymous with, it has offered me the opportunity to learn a tremendous amount while I have been here, and I cannot thank you enough for that. Thank you also, to Dr. Joel Burdick. From all our conversations over the years, whether on research or life in general, you have been a veritable font of wisdom and are one of the kindest individuals I have ever met. I will always cherish the conversations we had, and I look forward to more conversations over coffee in the future. Thank you, Dr. Richard Murray, for always seeming to have the most insightful questions regarding my work. I cannot fathom where my research might have gone were it not for your insight and friendly conversations, and I look forward to many more such conversations in the years to come as well. Thank you, Dr. Kanianthra Mani Chandy, for being one of the warmest professors I have worked with as a TA. You never failed to ask how my research was going and if you could help, and should teaching be in my future, I hope to emulate a small part of your kindness. Thank you, Dr. Guillaume Blanquart, for all the Zoom social hours, the cooking sessions, the advice, and the generally supportive influence you have provided during my time here. I am overjoyed that we have become friends over the years, and I look forward to many more such moments in the future. Finally, thank you, Dr. Reza Ahmadi, for all your help, advice, and mentorship over the years. From a challenging boss to a close confidant, words cannot express the impact you have had on my Ph.D. career, and I am very thankful for your presence as part of it.

Of course, no mention of faculty and mentors would be complete without proper acknowledgment of their incredible support staff. So, to all the administrative staff I've interfaced with over the years, notably Mikaela Laite, Lynn Seymour, Sonya

Lincoln, Jenni Campbell, and Holly Golcher, thank you. You have made my time here significantly easier, from helping me run all the SOPS events, to helping with the banquet when I became injured. It would be an understatement to say that I would only have been able to accomplish a fraction of what I have without all of your assistance.

Likewise, no social event would happen without my friends. So, to everyone in the OASIS group, thank you for all the great times over snacks and chai and all the amazing dance parties. To Caltech Improv and Caltech Theater generally, thank you for all the raucous Wednesday evening improvisation sessions that kept me sane during deadline season and the opportunity to express my creative side after being stuck in an office doing math all day. To all the SOPS sports team members, thank you for never failing to balance a good time with our competitive spirit and all the great memories with softball, volleyball, and inner tube water polo. To all my office members both past and present, thank you for putting up with my playing music randomly during the day, my random conversations when I got bored of work, and my using the office as a temporary storage place for all the events I threw over the years. Finally, thank you to my lab and all members of the Gates Thomas building. I like to think I had a hand in making our community more social and welcoming, but in truth, our community has always been a warm and welcoming one.

I would like to thank a few friends in particular, though. First, to Anushri Dixit, thank you for being a phenomenal colleague, friend, and confidant. Honestly, making the office pot of coffee and catching up was a highlight of my day, as were our usually successful attempts at playing pranks on Jagannadh Boddapati, whom I'd like to thank as well. From starting off as my housemate first year to becoming someone whom I consider a brother over time, I have appreciated all the moments we shared together. To Wyatt and Christy Ubellacker, thank you for all the morning swims and pleasant conversations. Honestly, waking up to go swimming in the mornings, especially when it was very cold outside, wasn't the most enjoyable time, but looking forward to swimming with friends made it easier. Also, thank you to both Skylar Wei and Wyatt for all their help with experiments over the years. I honestly could not have completed my Ph.D. without your help. To everyone in the food group, thank you for indulging me in my love for food. You all have become some of my closest friends towards the latter half of my Ph.D., and I wouldn't have it any other way. I look forward to many more adventures, whether food related or otherwise, in the years to come.

I would like to especially thank some of my closest friends who have also become my brothers over the years: Jack Weeks, Andy Akerson, Ethan Pickering, Connor McMahan, Zachary Iton, and Eshaan Patheria. Honestly, the words here will not do justice to the times we spent together, but I should like to note a few things. Jack, from being the person who sat next to me first year to living together for the remainder of grad school and all the moments that came with it, thank you for everything. Andy, the other member of the wolf pack, thank you for the late-night research conversations and the never-ending enthusiasm you bring to our times together. Ethan and Connor, the pandemic would not have been the same were it not for our weekly late-night chatting sessions together with Jack and Andy. Those moments were the highlights of those years, and I look forward to the time when we're all in the same town once again, and we can recreate those nights. Zac, from our first year when you thought you were not going to find people to share in your passions, to all the traditions we've made since then, thank you for all those shared moments. Lastly, Eshaan, thank you broba for introducing me to Bombay slang, for the many trips to SF and back, and generally for the warm, chaotic energy you bring to all our moments together. I cannot conceive of my graduate school career without any of you in it.

Finally, I would like to thank my family who helped shape the man I am today. This Ph.D. and thesis are dedicated to all of you who have supported me in my academic journey throughout, even if you may not have known what I was doing for research all the time. Words cannot express my love for all of you. Finally, to my father who did not have the chance to read this thesis, thank you for everything.

- Prithvi Akella

ABSTRACT

The well-known quote by George Box states that "All models are wrong, but some are useful", and the controls and robotics communities alike have followed a similar paradigm to make significant theoretical and practical advances in the study of controllable systems to date. However, recent robotic system requirements include formal considerations for system safety, especially as we engineer systems that are required to work alongside us in our daily lives. As such, current research directions require analyses that consider these inaccurate system models, our inaccurate understanding of the environments in which these systems operate, and their combined effects on safe, effective system operation, *e.g.* the canonical autonomous driving problem in exceedingly difficult-to-model urban environments. Recently, this has led to burgeoning efforts in a formal study of controller verification. Specifically, verification denotes the process of determining whether a controller steers its system to exhibit desired behaviors despite the variety of environments the system might face during operation, *e.g.* whether the autonomous car's controller successfully drives the car to a destination without crashing into obstacles or pedestrians along the way. However, formalization of such a verification pipeline has proved difficult to date, especially since both the models we use for controller synthesis and our understanding of system environments are typically inaccurate.

As a result, this thesis describes our efforts in the development of a formal verification pipeline that addresses a few key challenges in traditional approaches to safety-critical system verification. The first contribution centers on difficult, reactive test synthesis. By test synthesis, we mean the construction of a (potentially difficult) environment in which we require the system under test to perform its objective, *e.g.* placement of parked cars around which an autonomous vehicle must park. Typically phrased as an optimization problem over the space of allowable environments, these tests are "static" insofar as they do not react to the system's choices made during the test. We posit that such reactivity could more accurately identify worst-case system behavior. As a result, we phrase reactive, maximally difficult test synthesis as a game-theoretic optimization problem, leveraging the same control theoretic tools that facilitate safety-critical controller synthesis—control barrier functions and signal temporal logic. We prove that our proposed synthesis technique is always solvable and always produces a realizable test environment. Finally, we showcase our results by synthesizing reactive tests for both single and multi-agent systems.

The second set of contributions centers on our efforts in uncertainty quantification. Due to un-modeled system and environmental aspects affecting system evolution in unpredictable ways, real-life systems need not realize the same paths every time. As such, typical analyses phrase verification as an optimization problem minimizing the expected value of a function over system trajectories with the expectation taken over this path variability, the distribution for which is assumed to be known. However, we posit that such an analysis should be risk-aware, *i.e.* account for this variability in a more principled fashion than an expectation-specific analysis, and should not assume *apriori* knowledge of the distribution corresponding to path variability, as it will be unknown in practice. To that end, we develop methods to bound a subset of risk measures for random variables whose distributions are unknown. This subset includes both Value-at-Risk and other, coherent risk measures heavily utilized in the controls and robotics communities. Simultaneously, we note that the same procedure can be applied to a wide class of non-convex optimization problems. In doing so, we develop a percentile-based optimization approach that rapidly identifies percentile solutions to optimization problems, *i.e.* a 90-th percentile solution is as good as 90% of solutions in the considered decision space.

The third set of contributions focuses on the application of the prior mathematical developments to facilitate both risk-aware safety-critical system verification and controller synthesis. We phrase risk-aware controller verification as a risk-measure identification problem and utilize the prior bounding results to provide an efficient, dimensionally-independent verification procedure. Then, we phrase risk-aware controller synthesis as an optimization problem maximizing the bound provided by our risk-aware verification method and show this problem is solvable by the percentile optimization methods mentioned prior. Finally, we lay the foundation for the utilization of the aforementioned mathematical developments in other aspects of controls and robotics and communities more broadly. We show how risk-measure bounding can augment models both offline and online to robustify safety-critical controllers, how percentile optimization can facilitate "optimal" input selection and guarantee generation for non-convex finite-time optimal controllers, and how multiple applications of the percentile approach can also bound the optimality gap of reported percentile solutions. We showcase all these results on hardware for multiple systems and highlight the data efficiency of our proposed approaches.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Barrier-Based Test Synthesis for Safety-Critical Systems Subject to Timed Reach-Avoid Specifications," *Transactions on Automatic Control (Submitted)*, arXiv:2301.09622, arXiv:2301.09622, Jan. 2023. DOI: 10.48550/arXiv.2301.09622. arXiv: 2301.09622 [eess.SY],
P.A. extended prior work on formal test synthesis to guarantee the existence and maximal difficulty of tests generated via a prior framework, wrote the code for the examples, and wrote the manuscript.

[2] P. Akella and A. D. Ames, "Bounding Optimality Gaps for Non-Convex Optimization Problems: Applications to Nonlinear Safety-Critical Systems," *62nd Conference on Decisions and Control (Submitted)*, Apr. 2023. DOI: 10.48550/arXiv.2304.03739. arXiv: 2304.03739 [math.OC],
P.A. defined the variance function and the secondary percentile optimization technique, proved the theoretical results, and wrote the manuscript.

[3] P. Akella, A. Badithela, R. M. Murray, and A. D. Ames, "Lipschitz Continuity of Signal Temporal Logic Robustness Measures: Synthesizing Control Barrier Functions from One Expert Demonstration," *62nd Conference on Decisions and Control (Submitted)*, Apr. 2023. DOI: 10.48550/arXiv.2304.03849. arXiv: 2304.03849 [eess.SY],
P.A. assisted in the theoretical development, wrote the code for the examples, and wrote part of the manuscript.

[4] P. Akella, W. Ubellacker, and A. D. Ames, "Probabilistic Guarantees for Nonlinear Safety-Critical Optimal Control," *2023 International Conference Intelligent Robots and Systems (IROS) (Submitted)*, arXiv:2303.06258, Mar. 2023. DOI: 10.48550/arXiv.2303.06258. arXiv: 2303.06258 [math.OC],
P.A. developed the percentile-based optimization procedure leveraged for optimal control guarantees, wrote the code for the examples, helped with experimentation, and primarily authored the manuscript.

[5] P. Akella, M. Ahmadi, and A. D. Ames, "A Scenario Approach to Risk-Aware Safety-Critical System Verification," *arXiv e-prints*, arXiv:2203.02595, Mar. 2022. DOI: 10.48550/arXiv.2203.02595. arXiv: 2203.02595 [eess.SY],
P.A. developed the theoretical results and the first notion of a Robustness Random Variable, wrote the code for the examples, and wrote the manuscript.

[6] P. Akella and A. D. Ames, "A barrier-based scenario approach to verifying safety-critical systems," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 062–11 069, 2022. DOI: 10.1109/LRA.2022.3192805,

P.A. developed the underlying theory, helped write the code for the quadrupedal experiments, and wrote the manuscript.

[7] P. Akella and A. D. Ames, "Disturbance bounds for signal temporal logic task satisfaction: A dynamics perspective," *IEEE Control Systems Letters*, vol. 6, pp. 2018–2023, 2022. DOI: 10.1109/LCSYS.2021.3137267,
P.A. developed the disturbance bound optimization problem solved considered in the paper, wrote the code for the examples, and wrote the manuscript.

[8] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-Based Bounds for Coherent Risk Measures: Applications to Policy Synthesis and Verification," *The Artificial Intelligence Journal (Under Review)*, Apr. 2022. DOI: 10.48550/arXiv.2204.09833. arXiv: 2204.09833 [cs.AI],
P.A. developed the risk-measure bounds and the theoretical results on risk-aware verification and controller synthesis, wrote the code for the examples, and wrote the manuscript.

[9] P. Akella, W. Ubellacker, and A. D. Ames, "Safety-Critical Controller Verification via Sim2Real Gap Quantification," *2023 International Conference on Robotics and Automation (ICRA) (Accepted)*, arXiv:2209.09337, Sep. 2022. DOI: 10.48550/arXiv.2209.09337. arXiv: 2209.09337 [eess.SY].

[10] P. Akella, W. Ubellacker, and A. D. Ames, "Test and Evaluation of Quadrupedal Walking Gaits through Sim2Real Gap Quantification," *arXiv e-prints*, Jan. 2022. DOI: 10.48550/arXiv.2201.01323. arXiv: 2201.01323 [eess.SY],
P.A. developed the segmentation procedure to identify the sim2real gap as a composition of simulator and hardware robustnesses, helped write the code to run the experiments, and authored the manuscript.

[11] P. Akella, S. X. Wei, J. W. Burdick, and A. D. Ames, "Learning Disturbances Online for Risk-Aware Control: Risk-Aware Flight with Less Than One Minute of Data," *Conference on Learning for Dynamics and Control (L4DC) (Accepted)*, arXiv:2212.06253, arXiv:2212.06253, Dec. 2022. DOI: 10.48550/arXiv.2212.06253. arXiv: 2212.06253 [eess.SY],
P.A. defined Surface-at-Risk, wrote the algorithm to bound the Surface-at-Risk subject to noisy measurements, helped with experiments, and wrote the vast majority of the manuscript.

[12] H. Krasowski, P. Akella, A. Ames, and M. Althoff, "Verifiably Safe Reinforcement Learning with Probabilistic Guarantees via Temporal Logic," *2023 Conference on Decision and Control (Submitted)*, arXiv:2212.06129, arXiv:2212.06129, Dec. 2022. DOI: 10.48550/arXiv.2212.06129. arXiv: 2212.06129 [cs.RO],
P.A. assisted in the development of the Safe RL pipeline, helped write code and run experiments, and made minor changes to the manuscript.

[13] P. Akella, U. Rosolia, and A. D. Ames, "Learning Performance Bounds for Safety-Critical Systems," *arXiv e-prints*, arXiv:2109.04026, arXiv:2109.04026, Sep. 2021. DOI: 10.48550/arXiv.2109.04026. arXiv: 2109.04026

[eess.SY],
P.A. developed the underlying theory, wrote the Bayesian Optimization algorithm leveraged in the simulated examples, and wrote the manuscript.

[14] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Formal test synthesis for safety-critical autonomous systems based on control barrier functions," *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 790–795, 2020. DOI: 10.1109/CDC42340.2020.9303776,
P.A. developed the theoretical results, wrote the code for the examples, and wrote the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

The current landscape of robotics research heavily emphasizes the collective desire to engineer systems that reliably assist us in our daily lives and in the same scenarios in which we live them [1]–[3]. For example, we ask these systems to cook us our food [4]–[6], assist the sick and elderly [7]–[10], explore space on our behalf [11]–[13], and perhaps most prominently, drive us places [14]–[17]. To provide for this requirement on safe and reliable assistance, the control and robotics communities have pushed for a formal study of these concepts in recent years, *e.g.* in the context of control barrier functions [18], temporal logic [19], [20], and safe model predictive control [21]–[23] among other formalizations [24]–[26]. In these contexts, the nominal safety-critical controller synthesis paradigm follows a well-worn path: (1) develop a model for the system of interest (from first principles, system identification, or otherwise); (2) develop a safety-critical controller against simulations over this nominal model, likely leveraging the control-theoretic safety tools mentioned prior; (3) implement the controller on the system of interest; and (4) tune parameters until the controller exhibits the desired behavior reliably [27]–[31].

Simultaneously, to address the philosophical impact of that last step in the *apriori* controller-synthesis paradigm, the same two communities have recently seen a large push in the study of verification [32]–[34]. Succinctly, verification references the process of determining whether a controlled system exhibits its desired behavior in the environments in which it is required to operate, *e.g.* whether an autonomous car successfully drives individuals to their destinations despite any weather patterns or traffic conditions it may face. Then as mentioned, interest in a formal study of verification arose due to the discrepancy between safety-critical controller performance in simulation and on hardware—discrepancies in safety-critical performance between steps (2) and (3) of the aforementioned pipeline. For context, this variance arises as our models are oftentimes inaccurate representations of reality. Indeed, it is for this reason that most major robotics companies have teams of dedicated testing engineers to test and verify their specific systems before widespread production.

1.1 Motivating Questions and Corresponding Contributions

This thesis' contributions will center on a few advancements in verification and their utilization in a verifiable controller synthesis pipeline. To motivate these contributions, note first that verification has a well-studied history, from its roots in model checking [35], [36] to recent information-theoretic results [37], [38].

Reactive Difficult Test Generation: However, the first set of motivating questions arises from difficult test generation, which is typically phrased as an optimization problem in recent works [38]–[41]. Typically, the decision spaces for these optimization problems correspond to static environments frustrating the controller's ability to realize the desired system behavior. A canonical example in the autonomous driving literature concerns the placement of static cars around which the ego agent is required to navigate and safely park. While static tests are interesting, reactive tests, *e.g.* a situation where another car reacts to park in the same space chosen by the ego agent, might provide more useful information on worst-case system behavior. As such, the first two questions addressed are as follows:

1. *How can we generate reactive, difficult tests of system behavior?*

   (C) We develop a game-theoretic test-synthesis technique for nonlinear safety-critical systems subject to timed reach-avoid specifications.

2. *What can we guarantee about such a method?*

   (C) We guarantee that the aforementioned test-synthesis method will always produce a realizable test of system behavior, *i.e.* the game-theoretic optimization problem always has a solution, and that this solution is the most difficult test of system behavior at the system state x in question.

Risk-Aware Controller Verification and Synthesis: The second set of motivating questions arises from the canonical verification optimization problems leveraged in the information-theoretic works cited prior [39]–[41]. Intuitively, robots need not realize the same path every time, even when asked to repeat the same behavior. As such, each of the aforementioned works identifies worst-case expected system behavior over the uncertain paths the system may realize. However, each realized path is still a physical system behavior that should be accounted for in a holistic verification analysis. Indeed the robotics community has argued for a risk-aware accounting of uncertainty during synthesis [42], [43] by leveraging the risk measures

popularized for similar analyses carried out in the financial community [44], [45]. However, the primary hindrance in our ability to directly implement such risk-aware analyses, is that we lack knowledge of the uncertainties affecting system evolution, and as such, have no knowledge of the distribution we aim to analyze. As such, the second set of questions is as follows:

3. *How do we calculate risk measures for random variables whose distributions are unknown?*

   (C) We construct a randomized convex optimization problem, solvable by existing scenario optimization techniques, that produces upper bounds for value-at-risk and a large subset of coherent risk measures. We also provide probabilistic guarantees on the accuracy of these upper bounds.

   (C) We extend this risk-measure estimation procedure to rapidly synthesize percentile solutions to a large class of non-convex optimization problems. Here, percentile solutions are those solutions that outperform fractions of the decision space, *i.e.* a solution s^* is in the 90%-ile, if it is better at optimizing for an objective than 90% of other decisions $s \in \mathbb{S}$.

4. *Can we leverage these methods to formalize a pipeline for risk-aware controller synthesis and verification?*

   (C) We phrase risk-aware controller verification — typically posed as an optimization problem in the existing information-theoretic literature — as a risk-measure identification problem. Then, we leverage the prior risk-measure estimation results to provide conservative, probabilistic verification statements for arbitrary nonlinear safety-critical systems subject to any quantifiable behavioral specification.

   (C) We phrase risk-aware controller synthesis as an optimization problem solvable via the percentile methods developed prior and exhibit the efficacy of generated controllers both in simulation and on hardware.

Probabilistic Guarantees for Optimal Control: The third set of motivating questions arose from our prior contributions and their potential applications to all facets of the controller synthesis paradigm, *e.g.* model synthesis and validation, controller generation, and controller implementation. Specifically, we know our system models are inaccurate, prompting efforts in the learning community to identify this

model mismatch, *i.e.* the sim2real gap, at least as it concerns controller performance on hardware [46]–[48]. However, this procedure can be phrased as a non-convex optimization problem, and as such, is potentially solvable via the percentile method developed prior. In other veins, nonlinear optimal controllers are practically difficult to use given the computational complexity of finding a terminal invariant set [49]–[53] and the understanding that determination of solution optimality is equivalent to solving a Hamilton-Jacobi-Bellman problem [54]. As such, could percentile methods offer efficient means of identifying useful solutions? Similarly, parameter tuning for controller synthesis has been expressed as a non-convex optimization problem solvable by Bayesian methods [55]–[57]. However, these methods do not typically allow for risk-aware objectives and have unknown sample complexities. As such, could the prior risk-measure bounding and percentile methods resolve these issues? These notions lead to the third set of questions which are as follows:

5. *Can we verify the models we use for controller synthesis?*

   (C) We pose sim2real gap identification as a verification problem addressable by the aforementioned risk-aware verification pipeline. We demonstrate the accuracy of the reported sim2real gap bounds on a multi-agent unicycle system and a quadruped.

6. *Can we translate controller guarantees in simulation to hardware?*

   (C) We construct discrete-time stochastic nonlinear system models leveraging the prior sim2real gap identification results. Then, using the prior verification pipeline, we verify the safety-critical controller against the stochastic model. Then, we demonstrate that those controllers that exhibit a high probability of realizing effective behavior on the stochastic model similarly exhibit satisfactory performance on hardware.

7. *Can we efficiently identify control inputs for (potentially) non-convex optimal controllers?*

   (C) We show that all finite-time optimal control problems subject to torque bounds are solvable by percentile optimization techniques. Furthermore, we demonstrate the efficacy of these approaches on hardware.

   (C) We phrase the generation of controller guarantees as a verification problem and utilize the prior risk-measure estimation results to probabilisti-

cally determine recursive feasibility and maximum runtimes of nonlinear finite-time optimal controllers directly on hardware.

8. *Can we determine the sub-optimality of any of these approaches in their respective veins?*

   (C) We define a variance function over the information set generated by taking a percentile solution to a non-convex optimization problem. Then, we show that maximizing this variance function via a second application of the same percentile method provides an upper bound on the optimality gap of the reported percentile solution.

1.2 Structure

Chapter 2: This chapter reviews control barrier functions and signal temporal logic — two concepts heavily utilized throughout the thesis.

Chapter 3: This chapter details the efforts made in reactive test synthesis — motivating questions (1) and (2) from the prior section.

Chapter 4: This chapter details the efforts made in uncertainty quantification — motivating questions (3) and (8) in the prior section.

Chapter 5: This chapter details the efforts made in risk-aware controller synthesis and verification — motivating question (4) in the prior section.

Chapter 6: This chapter details the efforts made in the utilization of percentile optimization techniques to facilitate guarantees across all aspects of the controller synthesis paradigm — motivating questions (5)-(7) in the prior section.

Chapter 7: This chapter summarizes the contributions along with their impact and discusses future directions.

Chapter 2

REVIEWING CONTROL BARRIER FUNCTIONS AND SIGNAL TEMPORAL LOGIC

2.1 Control Barrier Functions

Continuous Time

Inspired by barrier methods in optimization (see Chapter 3 of [58]), control barrier functions are a tool used to ensure safety in safety-critical systems that are control-affine, *i.e.* with state x and control input u:

$$\dot{x} = f(x) + g(x)u, \quad x \in \mathcal{X} \subseteq \mathbb{R}^n,\ u \in \mathcal{U} \subseteq \mathbb{R}^m. \tag{2.1}$$

Here, $\mathcal{X}$ is the state space, and $\mathcal{U}$ is the input space. Control Barrier Functions (CBFs) are defined against an inequality using class-$\mathcal{K}_e$ functions $\alpha : \mathbb{R} \to \mathbb{R}$. These functions are such that $\alpha(0) = 0$ and $r \geq s \iff \alpha(r) \geq \alpha(s)$. Then, a control barrier function is defined as follows.

Definition 1. A *control barrier function* $h : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function satisfying the following inequality $\forall\ x \in \mathcal{X}$ and for some class-$\mathcal{K}_e$ function α:

$$\sup_{u \in \mathbb{R}^m} \left[\frac{\partial h}{\partial x} \left(f(x) + g(x)u \right) \right] \geq -\alpha(h(x)).$$

Similarly, we can define a time-varying control barrier function (TVCBF) as follows.

Definition 2. A *time-varying control barrier function* $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a differentiable function satisfying the following inequality $\forall\ (x, t) \in \mathcal{X} \times \mathbb{R}_{\geq 0}$ and for some class-$\mathcal{K}_e$ function α:

$$\sup_{u \in \mathbb{R}^m} \left[\frac{\partial h}{\partial x} \left(f(x) + g(x)u \right) + \frac{\partial h}{\partial t} \right] \geq -\alpha(h(x, t)). \tag{2.2}$$

In practice, these functions are utilized to guarantee the forward invariance of their 0-superlevel sets C, defined as follows for a TVCBF, as a CBF can be seen as a special class of their time-varying counterparts:

$$C = \{(x, t) \in \mathcal{X} \times \mathbb{R}_{\geq 0} \mid h(x, t) \geq 0\}. \tag{2.3}$$

To formalize that notion of forward invariance, we can define the set of control inputs satisfying the inequality (2.2):

$$K_{cbf}(x,t) = \left\{ u \in \mathcal{U} \;\middle|\; \dot{h}(x,u,t) \geq -\alpha(h(x,t)) \right\}. \tag{2.4}$$

Then, let $\phi_t^U(x_0)$ denote the flow of the control affine system (2.1) from the initial condition $x_0 \in \mathcal{X}$ when steered by a controller $U : \mathcal{X} \times \mathbb{R}_{\geq 0} \to \mathcal{U}$, *i.e.*

$$\phi_0^U(x_0) = x_0, \text{ and } \dot{\phi}_t^U(x_0) = f\left(\phi_t^U(x_0)\right) + g\left(\phi_t^U(x_0)\right) U\left(\phi_t^U(x_0), t\right). \tag{2.5}$$

From the study of nonlinear dynamical systems, we know that each solution $\phi^U(x_0)$ has an interval of existence $I \subseteq \mathbb{R}_{\geq 0} \cup \{\infty\}$, a set of times over which the solution exists. Then a subset $A \subset \mathcal{X} \times \mathbb{R}_{\geq 0}$ being forward invariant corresponds to the notion that the flow of the dynamical system always remains within the set A for all times t in the flow's interval of existence I.

Definition 3. Let $\phi^U(x_0)$ be the flow of a nonlinear system as per equation (2.5), let $I \subseteq \mathbb{R}_{\geq 0} \cup \{\infty\}$ be its interval of existence, and let $A \subset \mathcal{X} \times \mathbb{R}_{\geq 0}$. The set A is *forward invariant* for the solution $\phi^U(x_0)$ if and only if $\forall\, t \in I,\ (\phi_t^U(x_0), t) \in A$.

Then (TV)CBFs are useful insofar as they prescribe a (time-varying) set of control inputs such that if a controller U always chooses inputs in the corresponding set, the 0-superlevel set of the (TV)CBF will be rendered forward invariant. Phrased formally, the corresponding theorem will follow [18].

Theorem 1. *Let $\phi^U(x_0)$ be the flow of a nonlinear system as per equation 2.5 with an interval of existence I, let h be a TVCBF as per Definition 2 with* 0*-superlevel set C as per equation* (2.3)*, and let the valid input set at any state and time pair $K_{cbf}(x,t)$ be as per equation* (2.4) *for this TVCBF h. Then, if $(x_0, 0) \in C$*

$$U\left(\phi_t^U(x_0), t\right) \in K_{cbf}\left(\phi_t^U(x_0), t\right)\ \forall\, t \in I \implies C \text{ is forward invariant.}$$

Discrete Time

Keeping with existing terminology, continuous-time control barrier functions are simply referred to as control barrier functions, though their discrete-time counterparts are referred to as discrete-time control barrier functions. That being said, the purpose of these functions remains the same. Only the system abstraction changes. Specifically, for discrete-time control barrier functions, we assume the nominal nonlinear system model is as follows [59]:

$$x_{k+1} = f(x_k, u_k),\ x_k \in \mathcal{X} \subset \mathbb{R}^n,\ u_k \in \mathcal{U} \subset \mathbb{R}^m. \tag{2.6}$$

The definition of a discrete control barrier function stems immediately from the system dynamics (2.6) — this definition is adapted from Definition 2 in [60]:

Definition 4. For the discrete-time control system (2.6), a *discrete time-varying control barrier function* $h : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ satisfies the following condition for some class-$\mathcal{K}_e$ function α such that $\forall\ (x, t) \in \mathcal{X} \times \mathbb{R}_{\geq 0}$:

$$\exists\, u_k \in \mathcal{U} \text{ s.t. } h(x_{k+1}, t_{k+1}) - h(x_k, t_k) \geq -\alpha(h(x_k), t_k).$$

Then, these discrete-time control barrier functions can similarly be used to guarantee forward invariance of their 0-superlevel sets. As this invariance guarantee mirrors the same continuous-time exposition, the corresponding definitions and theoretical statements will not be reproduced for the sake of brevity.

2.2 Signal Temporal Logic

Signal Temporal Logic (STL) allows for succinct representation of complex time-varying system behavior [19]. STL formulas ψ are defined recursively as follows:

$$\psi ::= \mu \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2 \mid \psi_1 \,\mathrm{U}_{[a,b]}\, \psi_2.$$

Here, $a, b \in \mathbb{R}_{\geq 0} \cup \{\infty\}$, $a \leq b$, and μ denotes an atomic proposition which is evaluated over states $x \in \mathbb{R}^n$ and returns a boolean value if a function $b_\mu : \mathbb{R}^n \to \mathbb{R}$ is positive at x:

$$\mu(x) = \text{True} \iff b_\mu(x) \geq 0,\ [\![\mu]\!] = \{x \in \mathbb{R}^n \mid b_\mu(x) \geq 0\}. \tag{2.7}$$

We denote signals $s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and the space of all signals $\mathcal{S}^{\mathbb{R}^n} = \{s \mid s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n\}$. Then, we denote that a signal s satisfies a specification ψ at time t via the notation $(s, t) \models \psi$. The satisfaction operator $\models$ is defined recursively as follows[1]:

$$\begin{aligned}
(s,t) &\models \mu && \iff \mu(s(t)) = \text{True},\\
(s,t) &\models \neg\psi && \iff (s,t) \not\models \psi,\\
(s,t) &\models \psi_1 \vee \psi_2 && \iff (s,t) \models \psi_1 \vee (s,t) \models \psi_2,\\
(s,t) &\models \psi_1 \wedge \psi_2 && \iff (s,t) \models \psi_1 \wedge (s,t) \models \psi_2,\\
(s,t) &\models \psi_1 \,\mathrm{U}_{[a,b]}\, \psi_2 && \iff \exists\, t' \in [t+a, t+b] \text{ s.t. } ((s,t') \models \psi_2) \ \wedge \ldots\\
& && \qquad (\forall\, t'' \in [t+a, t'])\ (s,t'') \models \psi_1)\ .d
\end{aligned}$$

[1]There are two conventions for defining the until operator U. We adopt the convention that the signal must, at at least one point in time, simultaneously satisfy both ψ_1 and ψ_2, and the other convention does not require such a satisfaction overlap. As will be shown in sections to follow, this convention more accurately models typically use-cases in robotics where the specification to be satisfied for all time is typically a safety specification, *i.e.* obstacle avoidance.

Furthermore, every STL specification ψ comes equipped with a robustness measure ρ that evaluates signals $s \in \mathcal{S}^{\mathbb{R}^n}$. If for some time $t \in \mathbb{R}_{\geq 0}$, $\rho(s,t) \geq 0$, then the signal s satisfies the specification ψ [19], [36], [61], [62].

Definition 5. A function $\rho_\psi : \mathcal{S}^{\mathbb{R}^n} \times \mathbb{R}_+ \to \mathbb{R}$ is a *robustness measure* for an STL specification ψ if $\rho_\psi(s,t) \geq 0 \iff (s,t) \models \psi$.

Example 1. *Let* $\psi = \neg(\text{True}\, \mathrm{U}_{[0,2]}(|s(t)| > 2))$*, then any real-valued signal* $s : \mathbb{R}_{\geq 0} \to \mathbb{R}$ *satisfies* ψ *at time t, i.e.* $(s,t) \models \psi$ *if* $\forall\ t' \in [t, t+2]$, $|s(t')| \leq 2$*. The corresponding robustness measure* $\rho_\psi(s,t) = \min_{t' \in [t,t+2]}\ 2 - |s(t')|$.

While robustness measures defined according to Definition 5 align with the proposition definition in equation (2.7) and prior work in the controls community [63], [64], it is not the only way of defining such a measure, *e.g.* see Definition 3 of [19] or Section 2.3 in [61].

Chapter 3

REACTIVE TEST SYNTHESIS

This chapter was adapted from:

[1] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Barrier-Based Test Synthesis for Safety-Critical Systems Subject to Timed Reach-Avoid Specifications," *Transactions on Automatic Control (Submitted)*, arXiv:2301.09622, arXiv:2301.09622, Jan. 2023. DOI: 10.48550/arXiv.2301.09622. arXiv: 2301.09622 [eess.SY],

[2] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Formal test synthesis for safety-critical autonomous systems based on control barrier functions," *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 790–795, 2020. DOI: 10.1109/CDC42340.2020.9303776,

The first step in any verification procedure involves running the system to be verified through a series of tests and evaluating system performance. While the following chapters will focus more on the procedure as a whole, this chapter will focus on that initial testing step. Specifically, herein we detail our efforts in the generation of a reactive test-synthesis method for nonlinear safety-critical systems subject to timed reach-avoid specifications. As the test-synthesis problem has been well studied in existing literature, we will first start with a more in-depth review of existing work in this vein, before mentioning our specific contributions in this chapter.

3.1 Introduction

For safety-critical autonomous systems where failure can mean loss of human life, *e.g.* autonomous cars, autonomous flight vehicles, *etc*, it is natural to ask the following question: does the system's controller effectively render the system safe while steering it in satisfaction of its objective? Prior work in the literature approaches this question in two distinct ways. The first line of work aims to develop methods to verify whether a given controller ensures that its system satisfies its desired objective, despite any perturbation from an allowable set. Termed *verification*, the development and application of these techniques are of widespread study, but they are not the immediate focus of this chapter [36], [65]–[69]. The second version of this problem arises primarily in the test and evaluation of safety-critical cyber-physical systems. Here, either a high-dimensional state space or a requirement to search over system

trajectories frustrates the immediate application of the aforementioned verification techniques. As such, there is a non-trivial amount of work aimed at systematically generating more difficult tests of the onboard control architecture and evaluating system performance with rising test difficulty. Additionally, the vast majority of these techniques require *a-priori* knowledge of the controller-to-be-tested and a simulator of the system-controller pair. In a related and, to the best of our knowledge, a less explored vein, we believe the search for problematic system behavior independent of apriori controller knowledge is dually useful, as will be further explained.

Related Work

A growing area in the test and evaluation literature (T&E) centers on the design and generation of tests utilizing a simulator of the system-under-test (SUT) [70]. Here, a test corresponds to a specific environment setup in which test engineers evaluate the SUT's ability to satisfy a specified set of behaviors—the system objective. To facilitate a rigorous objective satisfaction analysis, these objectives are oftentimes expressed as linear or signal temporal logic specifications which come equipped with specification satisfaction methods [71], [72]. For signal temporal logic specifically, each specification comes equipped with a robustness measure—functions over state signal trajectories whose positive evaluation corresponds to specification satisfaction. To that end, some current work focuses on developing smoother robustness measures and using them as specification satisfaction monitors for real-time adaptation [73]–[77].

The test-synthesis question stems naturally from the existence of such robustness measures. More specifically, provided a quantifiable set of phenomena that can frustrate specification satisfaction, the motivating question asks whether one can determine phenomena that minimize this robustness measure. Here, decreasing robustness indicates increasing test difficulty as if a system has negative robustness with respect to specification satisfaction, then it has failed to satisfy this specification—its objective [72]. Each of the pre-eminent tools for automated test synthesis and falsification of system simulators expresses this parameter search as an optimization problem—S-Taliro [41], Breach [40], and more recently, VerifAI in conjunction with SCENIC [39]. Indeed there has also been a wealth of work using these tools to generate difficult tests of system behavior for multiple systems [78]–[83]. There has also been work on optimally generating tests for specific controllers, independent of these tools [84]–[87].

The overarching goal of test generation, however, is to uncover problematic true system behavior without exhaustively testing the true system. As such, there has been some work aimed at taking difficult simulator tests and realizing them on the real SUT [88]. However, significantly more work aims at adapting the aforementioned tools to generate successively harder tests of real-system behavior [89]–[92]. As such, these tests fall into two categories. Namely, they are *controller-specific* and *static* test cases insofar as they usually identify one parameter in a parametric set of disturbances that can frustrate system specification satisfaction for a given controller.

Provided that the goal of test synthesis is to determine difficult tests of system behavior, we posit that this study can and should be done controller-independent, as what is difficult for the system to achieve should be independent of the controller used to steer it. Furthermore, while static tests can uncover problematic system behavior, a time-varying environment might identify more problematic phenomena, as expressed in the static and reactive autonomous car example mentioned in the prior chapter. As such, we endeavor to develop a formal method for generating adversarial, controller-agnostic, and time-varying tests of safety-critical system behavior.

Summary of Contributions in this Chapter

Our contributions are five-fold and will be itemized as follows:

1. For both continuous and discrete-time systems, we develop game-theoretic, adversarial test-synthesis procedures based on control barrier functions and timed reach-avoid specifications. Subject to some assumptions on the system to be tested, these techniques satisfy the following criteria:
    - they are guaranteed to produce a realizable test of system behavior, and
    - they are provably the most difficult test of system behavior at that system state.
2. We extend our continuous-time analysis to develop a test-synthesis procedure for tests that perturb the system dynamics directly. We similarly prove the existence and maximal difficulty of tests in this setting as well.
3. We extend our discrete-time analysis to develop a predictive test-synthesis procedure that is guaranteed to be realizable and maximally difficult.
4. We extend both our continuous and discrete-time test-synthesis techniques to the scenario when the feasible space of tests may be time-varying or otherwise

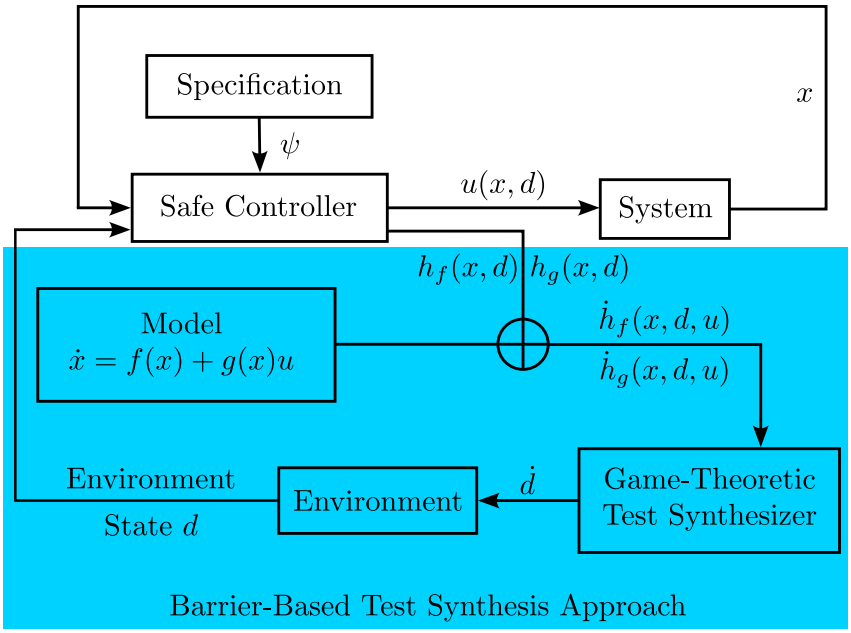

Figure 3.1: A general flowchart of our test-synthesis procedure for safety-critical systems subject to Timed-Reach Avoid Specifications as described in Chapter 3. We assume the specification ψ that influences the safe controller is express-able via control barrier functions h_f, h_g. Simultaneously, these same barrier functions are used in a game-theoretic test-synthesis procedure that exploits model knowledge to develop tests that are provably realizable and maximally difficult.

constrained. In this setting, we prove that our method is guaranteed to produce realizable and maximally difficult tests of system behavior.

5. We showcase the results of our test-generation procedure for each case. For the unconstrained examples, we showcase our results in simulation, mention some deficiencies resolved in our constrained extensions, and showcase the results of our constrained test-synthesis procedure by testing a quadruped robot's ability to navigate while avoiding moving obstacles.

For context, we restrict our study to timed reach-avoid specifications, a subset of STL specifications, as they are commonly used in the controls literature to represent basic robotic objectives, *e.g.* reach a goal and avoid obstacles [63], [93], [94].

Chapter Structure

To start, Section 3.2 will set up and formally state the problem under study in this chapter. Then, Section 3.3 details our adversarial test-synthesis procedure in the continuous setting and illustrates our main results in continuous time through a

simple example. Likewise, Section 3.4 details our results in the discrete setting and illustrates these results through a simple example as well. Finally, Section 3.5 extends the results of both of the prior sections by developing a test-synthesis procedure that has similar guarantees on existence and difficulty in a constrained test-synthesis scenario where the space of feasible tests varies with time or the system state. The same section also details the application of our test-synthesis procedure to provide difficult tests for a quadrupedal system in its satisfaction of a simple objective. Before starting, we will briefly define some notation.

Notation

The set $\mathbb{Z}_+ = \{0, 1, 2, \dots\}$. A function $\alpha : (-b, a) \to \mathbb{R} \cup \{-\infty, \infty\}$ where $a, b \in \mathbb{R}_{++}$, is an extended class-κ function κ_e if and only if $\alpha(0) = 0$, and for $r > s$, $\alpha(r) > \alpha(s)$. For any set A define $|A|$ to be the cardinality of A, *i.e.* the number of elements in A. For a function $f : X \to Y$ we say $f \in C^k(X)$ if f has at least k (partial) derivatives and its k-th (partial) derivative(s) is/are continuous, *i.e.* for $x \in \mathbb{R}^2$, $f = x^T x \in C^2(\mathbb{R}^2)$. For a set A, 2^A is the set of all subsets of A, *i.e.* $2^A = \{B \mid B \subseteq A\}$. Finally, we define two common STL specifications used throughout the chapter:

$$\mathbf{F}_{[a,b]}\, \mu = \text{True}\, \mathrm{U}_{[a,b]}\, \mu, \quad \mathbf{G}_{[a,b]}\, \mu = \neg(\mathbf{F}_{[a,b]}\, \neg\mu). \tag{3.1}$$

Here, $\mathbf{F}_{[a,b]}\, \mu$ indicates a specification where μ is to be true at some time $t' \in [t + a, t + b]$ with respect to some initial time t. Likewise, $\mathbf{G}_{[a,b]}\, \mu$ indicates a specification where μ is to be true for all times $t' \in [t + a, t + b]$ given an initial time t. Oftentimes, safety specifications require continued satisfaction of a predicate μ. In these cases, we will use the shorthand $\mathbf{G}_\infty\, \mu$.

3.2 Problem Formulation and Statement

To formally state the problem under study in this chapter, we will describe some common definitions and assumptions.

Definitions and Common Assumptions

To generate a test-synthesis procedure, we need to first formally define a test and the system specifications we aim to test through our procedure. We will start with the latter and use it to formalize the former. As mentioned, we restrict our analysis to timed reach-avoid specifications as these are commonly used specifications in the robotics literature [63], [95]–[97]. These specifications are defined as follows,

where **F** and **G** were defined in equation (3.1):

$$\psi = \mathbf{F}_{[0,t_{\max}]}\, \mu \wedge_{j\in\mathcal{J}} \mathbf{G}_{\infty}\, \omega_j. \tag{3.2}$$

Here, $\mathcal{J} = \{1, 2, \ldots, |\mathcal{J}|\}$ is merely a set of indices demarcating different safety objectives the system is to maintain, and $t_{\max}$ denotes the time by which the objective μ is to be achieved. To briefly remark, it is possible to group together all safety objectives ω_j into one overarching safety objective $\bar{\omega}$ and collapse specifications of the form in (3.2) to read as: $\psi = \psi = \mathbf{F}_{[0,t_{\max}]}\, \mu \wedge \mathbf{G}_{\infty}\, \bar{\omega}$. However, as will arise in the analysis to follow, finding a barrier function for the combined proposition $\bar{\omega}$ will likely prove far more difficult than finding a barrier function for each component specification. As a result, the above definition keeps in mind the practicality of identifying such barrier functions, though theoretically there is no difference. That being said, this leads to the first assumption in our work.

Assumption 1. We assume the specification ψ (3.2) is satisfiable.

Assumption 1 restricts against mutually exclusive or conflicting safety objectives in the overarching specification form (3.2). We make this assumption as from a test-generation perspective, determining tests for a specification that could never be satisfied is irrelevant. Any test for such an unsatisfiable specification would be hard as the system could never satisfy its objective by definition. That being said, determining the satisfiability of a given signal temporal logic specification is the subject of current work [98]. To elucidate the specifications of this form, we will provide an example.

Example 2. *Consider an idealized unicycle system on a 2-d plane. Let this system's goal be to navigate to a region defined as the* 0*-superlevel set of a function* $h^F : \mathbb{R}^2 \to \mathbb{R}$ *while avoiding a set of obstacles defined as the conjunction of the* 0*-sublevel sets of multiple other functions* $h^G_j : \mathbb{R}^2 \to \mathbb{R}$, $\forall\, j \in \mathcal{J} = \{1, 2, \ldots\}$. *See Figure 3.2 for an illustration. Then, the system's specification* $\psi = \mathbf{F}_{\infty}\, \mu \wedge_{j\in\mathcal{J}} \mathbf{G}_{\infty}\, \neg\omega_j$ *where* $[\![\mu]\!] = C_{h^F}$ *and* $[\![\neg\omega_j]\!] = C_{h^G_j}$. *For* ψ *to satisfy Assumption 1, there must be at least one set of obstacle and goal locations wherein the robot is capable of navigating to the goal while avoiding all obstacles, i.e. the goal shouldn't always be at infinity or always encapsulated by obstacles.*

Continuing with Example 2, the most natural test of this agent's behavior would be to see if it could satisfy its specification irrespective of the locations of obstacles

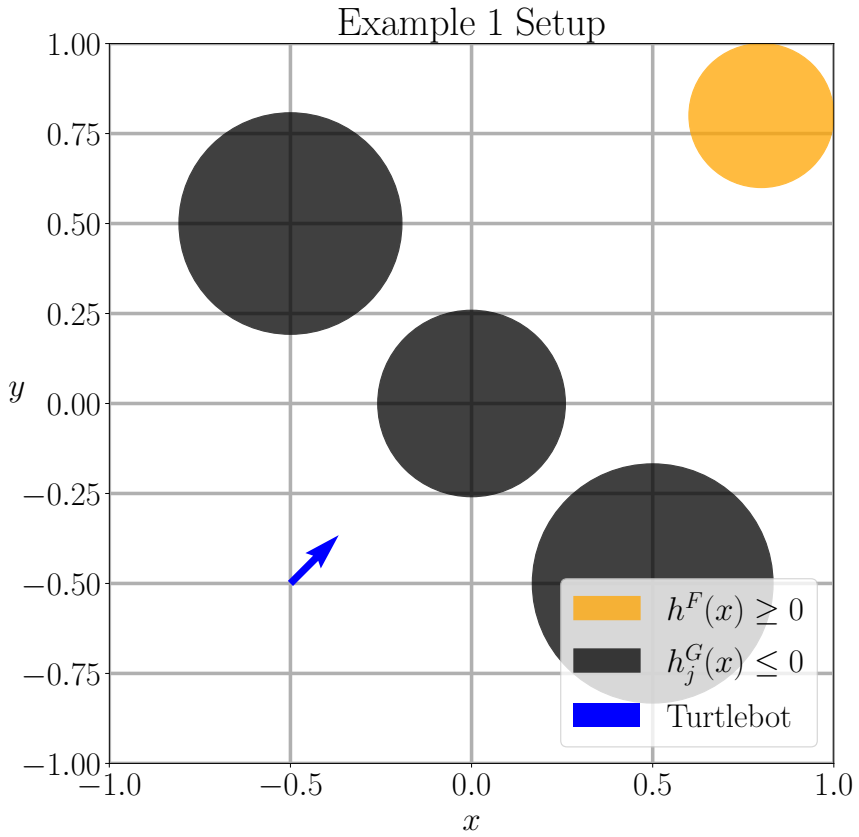

Figure 3.2: Example setup for Example 2. The agent is shown via the blue arrow, the obstacles via the black circles of varying sizes, and the goal via the golden circle. The 0-superlevel and sublevel sets of control barrier functions corresponding to these objectives follow the same color scheme as shown in the legend.

in its environment. The specific setup of obstacle locations would correspond to a specific test. This notion underlies how we will formalize tests. To start, we will formally define the system's environment.

Definition 6. The *environment* E is the state of the world in which the system operates including the state of the system itself, *e.g.* the cave in which a robot is traversing coupled with any motor failures the robot may have suffered, the airspace in which a jet flies along with any engine failures, *etc*. The state of the environment will be represented through the *environment state vector* x_E.

Here, we note that the environment's state vector x_E may be large, indeed even infinite, for real-life systems. As such, if we wanted to test and evaluate a system's ability to satisfy its specification independent of the environment it might face, we need some way of modeling at least part of the environment to understand its effects

on system specification satisfaction. Note, this does not imply that we know and can model everything about our system's intended environment. Rather, we assume that there exists some subset of this environment that we can model and can effectively use to test system behavior, *e.g.* we can model wind speeds a drone might face during operation and will test our drone by requiring it to satisfy its objective when subject to a set of wind-speeds that we can realize through our model. As such, we will make the following definition segmenting the system's environment into the set of modelable phenomena and un-modelable phenomena.

Definition 7. The state x_E of the environment E can be segmented into the system state x, a set of modelable phenomena $d \in \mathcal{D}$, and a set of un-modelable phenomena $w \in \mathbb{W}$, *i.e.* $x_E^T = [x^T, d^T, w^T]^T$. The space of modelable phenomena $\mathcal{D}$ is the *feasible test space* and each $d \in \mathcal{D}$ is a *test parameter vector*.

For Example 2 then, the environment E is the 2-d plane within which the agent operates and the obstacles lie. Here, the environment state x_E can be segmented into the agent's state, modelable obstacle locations $d \in \mathcal{D}$, and un-modelable table friction, signal delays, motor constants, *etc*, all of which comprise $w \in \mathbb{W}$. As motivated earlier, an example test would correspond to placing the obstacles at known locations *apriori* and then allowing the agent to try navigating to its goal. This motivates our definition of a test that will follow.

Definition 8. A *test* is a specific environment setup modeled by a specific *test parameter vector* $d \in \mathcal{D}$.

Per this definition, the outcome of the agent navigating to its goal need not be the same for two similar tests—this is primarily due to the lack of knowledge of $w \in \mathbb{W}$. However as these variables are un-modelable, we cannot restrict them during a test. As a result, when we formalize our test-generation procedure, we will only focus on determining a suitable choice of test parameter vector d, as the unknown phenomena w are by definition, unknown. Also, we would expect that running the same test twice might yield different outcomes each time. This scenario oftentimes happens in reality, *e.g.* running the same robot twice and experiencing slightly different behaviors each time, due to friction, battery power loss, *etc*. We will extend Example 2 to better illustrate Definition 8.

Example 3. *In the setup in Example 2, let each obstacle be defined as the interior of a circle and define* $[\![\omega_j]\!] = \{x \mid \|x - d_j\| \leq r_j\}$ *for each* $j \in \mathcal{J}$. *Then, define a vector*

$d = [d_1, r_1, d_2, r_2, \dots] \in \mathbb{R}^{3|\mathcal{J}|}$. *Each specific vector d constitutes a set of known obstacle locations. Requiring the agent to navigate to its goal in an environment E formed from that choice of obstacle locations d is an example test.*

Our goal is to constructively determine adversarial, time-varying tests of system behavior with respect to satisfaction of a timed reach-avoid specification in a controller-agnostic fashion. We first note that as a direct consequence of Definition 8, we can define a time-varying test as one where the test parameter vector d varies with time, *i.e.* an adversarial test law would be a function $\mathbb{T} : \mathbb{R}_+ \to \mathcal{D}$. For Example 3, a time-varying test would correspond to a scenario where the obstacles are moving while the agent is navigating to its goal. To facilitate the development of such a procedure, we require one assumption on the existence of control barrier functions corresponding to the timed reach-avoid specification ψ (3.2) that the system is to satisfy. This assumption permits us to make a relation between the satisfaction of the specification ψ and the positivity of their corresponding control barrier functions.

Assumption 2. Let μ, ω_j be predicates comprising the timed reach-avoid specification ψ (3.2). We assume there exists $\forall\ j \in \mathcal{J}$,

$$h_j^G : \mathbb{R}^n \times \mathcal{D} \to \mathbb{R} \text{ s.t. } h_j^G(x, d) \geq 0 \iff x \in [\![\omega_j]\!],$$
$$h^F : \mathbb{R}^n \times \mathcal{D} \to \mathbb{R} \text{ s.t. } h^F(x, d) \geq 0 \iff x \in [\![\mu]\!],$$

where each h_j^G and h^F are (discrete) control barrier functions as per Definition 1.

For context, Assumption 2 is not too restrictive. It builds off prior work that constructs control barrier functions for single/multi-agent systems subject to signal temporal logic specifications more broadly [63], [93], [94], [99]–[101]. As timed reach-avoid specifications are a subset of signal temporal logic specifications, and the existence of control barrier functions for these types of signal temporal logic tasks has been determined *apriori*, we will simply assume their existence for the time being. Now we can formally state the problem under study in this chapter.

Problem Statement

As mentioned, our goal is to develop an adversarial, time-varying test-generation procedure for safety-critical systems in a controller-agnostic fashion. Ideally, we would also like to develop a procedure that works for both continuous and discrete-time systems. Furthermore, we would like our generated tests to be maximally difficult with respect to a difficulty metric. As a first step in developing a concrete

problem statement, we would like to formalize a test difficulty metric—some function that is minimized at a given state x by the hardest test vector d at that state x. To that end, we will first define a space of feasible inputs $\mathcal{U}(x, d)$.

Definition 9. The *instantaneous feasible input space* $\mathcal{U}(x, d) \subseteq \mathcal{U}$ is the space of inputs satisfying the following condition: if $\forall\ t \geq 0$ (respectively all $k \in \mathbb{Z}_+$), the control inputs $u \in \mathcal{U}(x(t), d)$ (respectively $u \in \mathcal{U}(x_k, d)$), the resulting state trajectory $x(t)$ (respectively x_k) satisfies $\varphi \triangleq \wedge_{j\in\mathcal{J}}\, \mathbf{G}_\infty\, \omega_j$, *i.e.* $(x, 0) \models \varphi$.

Effectively, the feasible input space $\mathcal{U}(x, d)$ is the space of inputs that steer the system in satisfaction of its safety specifications $\mathbf{G}_\infty\, \omega_j$ in equation (3.2). On a related note, we will also require a function that discriminates between feasible input choices taken by the system in satisfaction of its desired objective $\mathbf{F}_{[0,t_{\max}]}\, \mu$. We will define this action discriminator function v as follows.

Definition 10. The *action discriminator* is a function $v : \mathcal{X} \times \mathcal{D} \times \mathcal{U} \to \mathbb{R}$ with $\mathcal{D}$ the feasible test space as per Definition 7 that satisfies the following condition: if $\forall\ t \geq 0$ (respectively all $k \in \mathbb{Z}_+$), the control inputs u are such that $v(x(t), d, u) \geq 0$ (respectively $v(x_k, d, u) \geq 0$), then the resulting state trajectory $x(t)$ (respectively x_k) satisfies $\varphi = \mathbf{F}_{[0,t_{\max}]}\, \mu$, *i.e.* $(x, 0) \models \varphi$.

Now, we can use the feasible input space $\mathcal{U}(x, d)$ and our action discriminator v to formalize our controller-agnostic difficulty measure.

Definition 11. An *instantaneous difficulty metric* $M : \mathcal{X}\times\mathcal{D} \to \mathbb{R}$ has a well-defined minimizer $\forall\ x \in \mathcal{X}$,

$$\mathbb{T}(x) = \underset{d\in\mathcal{D}}{\operatorname{argmin}}\ M(x, d),$$

that satisfies one of the two conditions below for the same state x,

$$\mathcal{U}(x, \mathbb{T}(x)) = \varnothing,\ \text{or}\ \mathbb{T}(x) = \underset{d\in\mathcal{D}}{\operatorname{argmin}}\ \max_{u\in\mathcal{U}(x,d)}\ v(x, d, u).$$

Here, $\mathcal{U}(x, d)$ is the system's feasible input space as per Definition 9, and v is the action discriminator as per Definition 10.

Intuitively then, our difficulty metric M quantifies test difficulty through minimizing the "maximum possible increment" the system could take towards satisfying its desired objective μ while maintaining its safety specifications ω_j. Here, the

"maximum possible increment" to satisfaction is provided through the action discriminator v which closely resembles the robustness measures traditional to signal temporal logic [72], [75], [102]. Effectively, our action discriminator is an instantaneous version of this robustness measure in that it assigns positive values to those system actions u which bring the system closer to satisfying its objective. It is this instantaneous notion that allows us to develop time-varying tests. As such, our formal problem statement will follow.

Problem 1. *For both control system abstractions* (2.1) *or* (2.6)*, let the system's operational specification ψ satisfy equation* (3.2)*. Then, for either system abstraction, develop (perhaps different) adversarial test-synthesis procedures $\mathbb{T} : \mathcal{X} \to \mathcal{D}$ that*

- *are guaranteed to produce a realizable test, i.e. $\forall\, x \in \mathcal{X},\ \exists\, d \in \mathcal{D}$ such that $d = \mathbb{T}(x)$;*
- *are the most difficult tests of system behavior at that state, independent of control input, i.e. $\mathbb{T}(x) = \operatorname{argmin}_{d \in \mathcal{D}} M(x, d)$ for some difficulty measure M that satisfies definition 11.*

Here, we note that while we have defined our specifications to be evaluated over continuous time, they can still be used to express specifications for discrete-time systems [93], [96], [99], [103]. In the discrete case, time is indexed by $k \in \mathbb{Z}_+$ as opposed to the continuous analog where time is measured on the positive reals.

3.3 Continuous-Time Test Generation

This section states and proves the first half of our main results in this chapter, the development of an adversarial, time-varying test-synthesis procedure for continuous, control-affine control systems of the form (2.1) subject to timed reach-avoid specifications ψ of the form in (3.2). We will briefly describe the overarching methodology behind our approach, state the developed minimax problem for test synthesis, and end with two theorems regarding its use.

Overarching Idea: Based on the recursive definition of the satisfaction relation $\models$, a controller for the nominal system (2.1) guarantees system satisfaction of the timed reach-avoid specification ψ in (3.2) if and only if,

$$\forall\, t \geq 0,\ x(t) \in \cap_{j \in \mathcal{J}} [\![\omega_j]\!],\ \text{and}\ \exists\, t' \in [0, t_{\max}]\ \text{ s. t. }\ x(t') \in [\![\mu]\!]. \tag{3.4}$$

Based on Assumption 2, the requirement in equation (3.4) translates to the following statement, with C as the 0-superlevel set for the control barrier function shown as a subscript:

$$\forall\, t \geq 0,\ x(t) \in \bigcap_{j \in \mathcal{J}} C_{h_j^G},\ \text{and } \exists\, t' \in [0, t_{\max}]\ \text{ s. t. } x(t') \in C_{h^F}. \tag{3.5}$$

Without loss of generality, we can assume that at $t = 0$,

$$x(0) \in \bigcap_{j \in \mathcal{J}} C_{h_j^G},\ \text{and } x(0) \notin C_{h^F}, \tag{3.6}$$

as otherwise, the system would either never be able to satisfy ψ—as it started in an unsafe region, *i.e.* $x(0) \notin [\![\omega_j]\!]$ for at least one $j \in \mathcal{J}$—or it would satisfy ψ by remaining stationary. Neither case is interesting from a test-synthesis perspective. Hence, as each h_j^G and h^F is a control barrier function, one way of transitioning from the starting condition (3.6) to the end condition (3.5) is for the controller to satisfy the following inequalities for some functions $\alpha, \alpha_j \in \kappa_e$ and $\tau > 0$:

$$\dot{h}^F(x(t), d, u) \geq -\alpha\left(h^F(x(t), d)\right) + \tau,\ \forall\, t \in [0, T],$$
$$\dot{h}_j^G(x(t), d, u) \geq -\alpha_j\left(h_j^G(x(t), d)\right),\ \forall\, t \geq 0.$$

The above conditions provide us a way of generating quantifiably adversarial tests—generate tests that are designed to minimize satisfaction of these inequalities.

Statement of Continuous-Time Results

To formalize this satisfaction minimization idea mentioned prior, we will first specify a set of feasible inputs $\mathcal{U}(x, d)$ and an action discriminator function v as follows:

$$\mathcal{U}(x, d) = \left\{ u \in \mathcal{U} \;\middle|\; \dot{h}_j^G(x, d, u) \geq -\alpha_j\left(h_j^G(x, d)\right),\ \forall\, j \right\}, \tag{3.7}$$

$$v(x, d, u) = \dot{h}^F(x, d, u) - \tau,\ \text{for some } \tau > 0. \tag{3.8}$$

Here, we first note that $\mathcal{U}(x, d)$ identifies those inputs $u \in \mathcal{U}$ that satisfy the CBF condition expressed in Definition 1 for the barrier functions h_j^G. As such, by Theorem 1 and Assumption 2, we know that $\mathcal{U}(x, d)$ is a valid feasible input space as per Definition 9. Likewise, for some $\tau > 0$, v is also a valid action discriminator as per Definition 10. As we want to keep our test-synthesis framework controller-agnostic, we will initially propose the following minimax problem over all feasible inputs in equation (3.7) as our test synthesizer:

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \max_{u \in \mathcal{U}(x,d)} \dot{h}^F(x, d, u). \tag{3.9}$$

Remark on Local/Global Conservatism: However, this proposed synthesis technique has two shortcomings. First, our proposed synthesis technique may suffer from a local-global problem—that by determining a worst-case test at a given state x we find a time-varying test sequence $\mathbb{T}(x(t))$ that is only locally optimal, *i.e.* locally difficult but not globally difficult. Rectifying this shortcoming while maintaining our controller-agnostic approach would require us to optimize over control sequences in the inner maximization problem in (3.9). As we assume nonlinear dynamics in (2.1), however, this would result in a non-convex inner maximization problem wherein it would be difficult to determine the feasibility of any resulting minimax problem. In the proposed case, we can guarantee both feasibility and maximal test difficulty. That being said, the determination of tests that are globally difficult and verifying wholistic system behavior is the subject of future chapters.

The second shortcoming is that it may be the case that there exist test parameter vectors $d \in \mathcal{D}$ such that the feasible input space $\mathcal{U}(x, d) = \varnothing$, as we have made no effort to restrict against this scenario. In these cases, the inner maximization problem would be ill-posed, frustrating any further analysis. However, were there such a test parameter vector d, we would like to identify it as a worst-case test. Indeed this is one of the conditions we used to define our difficulty metric in Definition 11. To facilitate analysis in the scenario where the feasible input space might be empty then, we will define a function $\mathcal{F}$ which filters a solution based on the emptiness (or lack thereof) of a provided set. More accurately, for two scalars $\epsilon, \zeta \in \mathbb{R}$, an arbitrary set $A \subset \mathbb{R}^m$, and a vector $a \in \mathbb{R}^m$, define $\mathcal{F}$ as follows:

$$\mathcal{F}(\epsilon, a, A, \zeta) = \begin{cases} \epsilon & \text{if } a \in A, \\ \zeta & \text{else.} \end{cases} \tag{3.10}$$

Then, we will make one assumption on the system dynamics (2.1), the feasible test space $\mathcal{D}$, and our control barrier functions.

Assumption 3. Both the state space $\mathcal{X}$ and the feasible test space $\mathcal{D}$ are compact, the input space $\mathcal{U}$ is a compact polytope in $\mathbb{R}^m$, and $h^F, h^G_j \in C^1(\mathcal{X} \times \mathcal{D})$.

For context, Assumption 3 is not that restrictive. First, we restrict the space of feasible tests $\mathcal{D}$ to a compact set as we do not expect our test parameter vector d to tend to $\pm\infty$. Furthermore, if d is bounded, we expect our realized test to be capable of taking values on the boundary. For an example, consider Example 3 where the obstacles are allowed to take center locations on the boundary of the

hyper-rectangle in which they are confined. Then, the assumptions of compactness on the state space $\mathcal{X}$ and polytopic compactness of the input space $\mathcal{U}$ are satisfied by most torque-bounded robotic systems. Finally, the assumption of continuity of the control barrier functions and their first partial derivatives is an easily satisfied restriction on the smoothness of our control barrier functions. For an example, consider Example 2 where this holds.

Then, we will define a minimum satisfaction value m that meets the following inequality $\forall\, d \in \mathcal{D}$ and $x \in \mathcal{X}$:

$$m \leq \min_{u \in \mathcal{U},\ x \in \mathcal{X},\ d \in \mathcal{D}} \dot{h}^F(x, d, u). \tag{3.11}$$

While it is unclear at the moment whether such an m exists, we will formally prove its existence in the proof for Theorem 2 to follow. Now, we can formally state our test-synthesis procedure, with $\mathcal{F}$ as in (3.10), m as in equation (3.11), and $\mathcal{U}(x, d)$ as per equation (3.7).

$$\mathbb{T}(x) = \operatorname*{argmin}_{d \in \mathcal{D}} \max_{u \in \mathcal{U}} \mathcal{F}\left(\dot{h}^F(x, d, u), u, \mathcal{U}(x, d), m\right). \tag{3.12}$$

This leads to our first theorem in this chapter—that minimax problem (3.12) is guaranteed to have a solution $\forall\, x \in \mathcal{X}$.

Theorem 2. *Let Assumption 3 hold. The test synthesizer in* (3.12) *has a solution* $d \in \mathcal{D}$ *for every* $x \in \mathcal{X}$*, i.e.*

$$\forall\, x \in \mathcal{X}\ \exists\, d \in \mathcal{D}\ \text{ s. t. }\ d = \mathbb{T}(x).$$

Concerning the second aspect of our problem statement then, we can define an instantaneous, controller-agnostic difficulty measure $M : \mathcal{X} \times \mathcal{D} \to \mathbb{R}$ as the interior maximization problem in (3.12).

$$M(x, d) = \max_{u \in \mathcal{U}} \mathcal{F}\left(\dot{h}^F(x, d, u), u, \mathcal{U}(x, d), m\right). \tag{3.13}$$

As in the case of defining a set of feasible inputs, we need to show that this difficulty measure satisfies the conditions in Definition 11 to be valid. This leads to the following Lemma.

Lemma 1. *M as defined in equation* (3.13) *is a difficulty measure as per Definition 11 with feasible input space* $\mathcal{U}(x, d)$ *as per equation* (3.7) *and action discriminator v as per equation* (3.8).

Then, Theorem 2 and Lemma 1 directly provide for the following corollary regarding minimization of M.

Corollary 1. *Let Assumption 3 hold. The test synthesizer in* (3.12) *minimizes the difficulty measure M in* (3.13) *over all $d \in \mathcal{D}$, i.e.*

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \; M(x, d)$$

Proof of Continuous-Time Results

This section will contain all necessary lemmas and proofs for all theoretical results stated in Section 3.3. To start, we will reiterate a known result from the study of minimax problems as taken from the proof for Theorem 1 in [104]:

Lemma 2. *(From Theorem 1 in [104]) Let $\mathbb{X}$ and $\mathbb{Y}$ be compact sets, and let $f : \mathbb{X} \times \mathbb{Y} \to \mathbb{R}$ be a function that is continuous in both its arguments. The following minimax problem has a solution, i.e.*

$$\exists\, x^* \in \mathbb{X},\; y^* \in \mathbb{Y} \text{ s.t. } f(x^*, y^*) = \min_{x \in \mathbb{X}} \max_{y \in \mathbb{Y}} f(x, y).$$

Second, for any state x, we can partition the space of feasible tests $\mathcal{D}$ into a set that does not permit feasible inputs and its complement:

$$\Gamma(x) \triangleq \{d \in \mathcal{D} \mid \mathcal{U}(x, d) = \varnothing\}. \tag{3.14}$$

With Lemma 2 and Γ above, we can prove Theorem 2.

Proof: This proof will follow a case-by-case argument. These cases are (Case 1) $\Gamma(x) \neq \varnothing$ and (Case 2) $\Gamma(x) = \varnothing$. Here, $\Gamma(x)$ is as defined in (3.14).

Case 1 $\Gamma(x) \neq \varnothing$**:** In this case, $\forall\, d \in \Gamma(x)$, $\mathcal{U}(x, d) = \varnothing$. By definition of $\mathcal{F}$ in equation (3.10), this implies that $\forall\, d \in \Gamma(x)$

$$\mathcal{F}\left(\dot{h}^F(x, d, u), u, \mathcal{U}(x, d), m\right) = m,\; \forall\, u \in \mathcal{U}, \tag{3.15}$$

with m as in equation (3.11). As mentioned earlier, we still need to formally prove that such an m exists. This arises through the application of the Extreme Value Theorem. As $h^F \in C^1(\mathcal{X} \times \mathcal{D})$ by Assumption 3 and the dynamics in (2.1) are control-affine, $\dot{h}^F(x, d, u)$ is continuous in all three of its arguments. By compactness of the feasible spaces for the minimization problem in (3.11), Extreme Value Theorem guarantees a solution to the same minimization problem, and setting m to be that solution suffices to prove the existence of m.

For the remainder of the proof, we will use the notation offered by M in equation (3.13) to represent the value of the inner maximization problem in equation (3.12). To prove the required result then, we need to show that $M(x, d)$ is lower bounded by some value and that there exists some $d \in \mathcal{D}$ that achieves this value. To start, we claim that $M(x, d) \geq m, \ \forall \ d \in \mathcal{D}$. This is easily verifiable for all $d \in \Gamma(x)$ by equality (3.15). It remains to show this lower bound works for all $d \in \mathcal{D}$ such that $d \notin \Gamma(x)$. This stems from the definition of m in equation (3.11). For each $d \notin \Gamma(x)$, $\mathcal{U}(x, d) \neq \emptyset$. As a result,

$$M(x, d) = \max_{u \in \mathcal{U}(x,d)} \dot{h}^F(x, d, u) \geq m, \ \forall \ d \in \mathcal{D} \cap \Gamma(x)^C.$$

To finish the proof for this case, at least one test parameter vector $d \in \mathcal{D}$ must ensure that $M(x, d) = m$, and any vector $d \in \Gamma(x)$ satisfies this criterion.

Case 2 $\Gamma(x) = \emptyset$**:** In this case, we note that the inner maximization problem in the feedback law (3.12) is equivalent to a Linear Program:

$$\begin{aligned} \min_{d \in \mathcal{D}} \quad \max_{u \in \mathbb{R}^m} \quad & c(d)^T u, && \\ \text{subject to} \quad & Au \leq b, && (\equiv u \in \mathcal{U}), \\ & C(d)u \leq k(d), && (\equiv u \in \mathcal{U}(x, d)). \end{aligned} \tag{3.16}$$

LP duality turns equation (3.16) into the following:

$$\min_{d \in \mathcal{D}, \lambda \geq 0, \mu \geq 0} \ \max_{u \in \mathbb{R}^m} \begin{bmatrix} c(d) \\ -\lambda \\ -\mu \end{bmatrix}^T \begin{bmatrix} u \\ Au - b \\ C(d)u - k(d) \end{bmatrix}. \tag{3.17}$$

For minimax problem (3.17), we note that if we further constrain the inner maxi-

mization problem such that $u \in \mathcal{U}$, this does not change the solution:

$$\min_{d\in\mathcal{D},\lambda\geq 0,\mu\geq 0} \max_{u\in\mathcal{U}} \begin{bmatrix} c(d) \\ -\lambda \\ -\mu \end{bmatrix}^T \begin{bmatrix} u \\ Au-b \\ C(d)u-k(d) \end{bmatrix} \tag{3.18}$$

$$= \min_{d\in\mathcal{D},\lambda\geq 0,\mu\geq 0,\gamma\geq 0} \max_{u\in\mathbb{R}^m} \begin{bmatrix} c(d) \\ -\lambda \\ -\mu \\ -\gamma \end{bmatrix}^T \begin{bmatrix} u \\ Au-b \\ C(d)u-k(d) \\ Au-b \end{bmatrix},$$

$$= \min_{d\in\mathcal{D},\beta\geq 0,\mu\geq 0} \max_{u\in\mathbb{R}^m} \begin{bmatrix} c(d) \\ -\beta \\ -\mu \end{bmatrix}^T \begin{bmatrix} u \\ Au-b \\ C(d)u-k(d) \end{bmatrix},$$

$$= (3.17).$$

Additionally, for any $d \in \mathcal{D}$, minimax problem (3.17) has a solution. This stems from the fact that $\Gamma(x) = \varnothing$, and as a result, $\mathcal{U}(x, d)$ is a non-empty, closed polytope in $\mathbb{R}^m$ (see equation (3.14) for reference). Therefore, the inner maximization problem in (3.16) has a solution, and via LP duality, so to does (3.17) have a solution. Furthermore, as minimax problem (3.18) is equivalent to minimax problem (3.17), so to does (3.18) have a solution for any $d \in \mathcal{D}$. In addition, the d-dependent Lagrange multipliers for this solution $\lambda^*(d) < \infty$ and $\mu^*(d) < \infty$, as a solution u^* exists. As this is valid $\forall\ d \in \mathcal{D}$, we note that $\exists\ M_\lambda < \infty$ and $M_\mu < \infty$ such that $\lambda^*(d) \leq M_\lambda$ and $\mu^*(d) \leq M_\mu$ element-wise $\forall\ d \in \mathcal{D}$. If this were not the case, then there exists at least one $d \in \mathcal{D}$ such that $\lambda^*(d) \to \infty$ or $\mu^*(d) \to \infty$, implying infeasibility of the inner maximization problem in (3.16), which is a contradiction. As a result, we can uniformly upper bound λ, μ in (3.18):

$$(3.16) = \min_{\substack{d\in\mathcal{D}, \\ 0\leq\lambda\leq M_\lambda, \\ 0\leq\mu\leq M_\mu}} \max_{u\in\mathcal{U}} \begin{bmatrix} c(d) \\ -\lambda \\ -\mu \end{bmatrix}^T \begin{bmatrix} u \\ Au-b \\ C(d)u-k(d) \end{bmatrix}. \tag{3.19}$$

Finally, the minimax problem (3.19) satisfies the conditions for Lemma 2, guaranteeing a solution, *i.e.* $\exists\ d \in \mathcal{D}$ such that $d = \mathbb{T}(x)$.

For an arbitrary $x \in \mathcal{X}$, the cases above prove that $\exists\ d \in \mathcal{D}$ such that $d = \mathbb{T}(x)$. As the state x was left arbitrary, this result is valid $\forall\ x \in \mathcal{X}$, completing the proof. ■

This concludes the proof for Theorem 2. Next, we will prove Lemma 1, to show that we minimize a valid difficulty measure with our proposed approach.

Proof: This proof will proceed in a case-by-case fashion as had the proof for Theorem 2. These cases will be $\Gamma(x) \neq \varnothing$ (Case 1) and $\Gamma(x) = \varnothing$ (Case 2), with $\Gamma(x)$ as defined in equation (3.14). In both cases, however, we know via Theorem 2 that there exists a minimizer $\mathbb{T}(x)$ of our proposed state-based difficulty metric M. Here, $\mathbb{T}(x)$ is defined in equation (3.12) and M is defined in equation (3.13).

Case 1 $\Gamma(x) \neq \varnothing$: In this case, we know via the proof for Theorem 2 that the minimizer $\mathbb{T}(x) \in \Gamma(x)$. As a result, $\mathcal{U}(x, \mathbb{T}(x)) = \varnothing$. As such, we know that in this case, any minimizers d of M—which are guaranteed to exist via Theorem 2—are such that $\mathcal{U}(x, d) = \varnothing$. As such, they satisfy the first criteria for M to be a valid difficulty measure as per Definition 11.

Case 2 $\Gamma(x) = \varnothing$: In this case, we know via the proof for Theorem 2 that the minimizer $\mathbb{T}(x)$ yields a non-empty feasible input space, *i.e.* $\mathcal{U}(x, \mathbb{T}(x)) \neq \varnothing$. As a result, by definition of $\mathbb{T}(x)$ in equation (3.12), the filtration function $\mathcal{F}$ in equation (3.15), and the action discriminator v in equation (3.8) we have the following equality:

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \max_{u \in \mathcal{U}(x,d)} v(x, d, u).$$

As such, the minimizer $\mathbb{T}(x)$ satisfies the second condition for M to be a valid difficulty metric in this case.

As the choice of $x \in \mathcal{X}$ was left arbitrary, the prior logic holds $\forall\, x \in \mathcal{X}$, thus concluding the proof that M as per equation (3.13) is a valid difficulty measure as per Definition 11. ■

Finally, Corollary 1 is a direct consequence of Theorem 2 and Lemma 1.

Proof: This is a consequence of Theorem 2 and Lemma 1. ■

Corollaries—Perturbing the System Dynamics

In the prior section, we stated and proved two theorems regarding the existence and maximal difficulty of the adversarial, time-varying tests generated by our proposed technique, minimax problem (3.12). However, this setting only accounts for the scenario where the test permits perturbation of the truth regions for the timed reach-avoid predicates μ and ω_j (by Assumption 2). What if instead, we wanted a test where we simulated a motor failure, increased or decreased system friction, or other

system-specific failures? This changes the system dynamics as follows, where the dependence of the dynamics f, g on d correspond to simulations of motor failure and the Cd term corresponds to increased or decreased friction:

$$\dot{x} = f(x, d) + g(x, d)u + Cd,\ C \in \mathbb{R}^{n \times p}. \tag{3.20}$$

Here, we can prove that our proposed test-synthesis procedure is still guaranteed to produce realizable and maximally difficult tests, as expressed through the following two Corollaries. Corollary 2 states that our synthesizer in (3.12) is still guaranteed to produce realizable tests of system behavior in this setting. Likewise, Corollary 3 proves that these tests are maximally difficult with respect to the same difficulty metric M as in (3.13). In both cases, all time derivatives are taken with respect to the dynamics in equation (3.20). We will start first with Corollary 2.

Corollary 2. *Let the system dynamics be as in* (3.20)*, let Assumption 3 hold, and let f and g both be continuous in d. The test synthesizer in* (3.12) *is guaranteed to have a solution $d \in \mathcal{D}\ \forall\ x \in \mathcal{X}$, i.e.*

$$\forall\ x \in \mathcal{X}\ \exists\ d \in \mathcal{D}\ \text{ s. t. }\ d = \mathbb{T}(x).$$

Proof: The proof for this corollary follows in the footsteps of the proof for Theorem 2 in a similar case-by-case fashion. We can partition the feasible test space $\mathcal{D}$ with $\Gamma(x)$ as defined prior in equation (3.14), and set up the same two cases as prior. In the first case, $\Gamma(x) \neq \varnothing$ and changing the system dynamics does not change the outcome. The optimal solution $\mathbb{T}(x) \in \Gamma(x)$. However, the latter case where $\Gamma(x) = \varnothing$ does change slightly. The interior maximization problem is still a Linear Program, and the entire chain of logic until equation (3.19) still holds. However, to proceed with the last step and use Lemma 2 to complete the proof, we need to guarantee that the matrix multiplication in equation (3.19) is continuous in d, λ, μ and u. Continuity in λ, μ, and u is assured via linearity in those terms. Finally, continuity in d is assured via the assumptions of continuity in the statement of Corollary 2. As a result, we can use Lemma 2, thus completing the proof. ■

Next, Corollary 3 will prove the optimal difficulty of the tests generated via our synthesizer in the perturbed setting.

Corollary 3. *Let the system dynamics be as in* (3.20)*, let Assumption 3 hold, and let f and g all be continuous in d. The test synthesizer in* (3.12) *minimizes the difficulty measure M in equation* (3.13)*, i.e.,*

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}}\ M(x, d).$$

Proof: This is a consequence of Corollary 2. ■

Continuous-Time Examples

In this section, we will illustrate our main results through examples extending Examples 2 and 3. For completeness, we will state the system dynamics as follows:

$$x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \; \dot{x} = \underbrace{\begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}}_{g(x)} u, \; x \in [-1,1]^2 \times [0, 2\pi], \tag{3.21}$$
$$u = [u_1, u_2]^T \in [-0.2, 0.2] \times [-1, 1].$$

Equation (3.21) implies that $\mathcal{X} = [0,1]^2 \times [0, 2\pi]$ and $\mathcal{U} = [-0.2, 0.2] \times [-1, 1]$, and both satisfy the conditions for Assumption 3. To generate a minimax test synthesizer of the form in equation (3.12) we require a system specification and associated control barrier functions. For our example, our specification

$$\begin{aligned} \psi &= \mathbf{F}\, \mu \wedge_{j \in \mathcal{J}} \mathbf{G}\, \omega_j, \\ [\![\mu]\!] &= \{x \in \mathcal{X} \mid \|Px - g\| \leq 0.25\}, \\ [\![\omega_j]\!] &= \left\{x \in \mathcal{X} \mid \|Px - o_j\| \geq 0.175\right\}. \end{aligned}$$

Here, $g \in \mathbb{R}^2$ denotes the center of a goal region the system is to enter, $o_j \in \mathbb{R}^2$ denotes the center of an obstacle the system is to stay away from, and P projects the system state onto the $x - y$ plane. Then, the control barrier functions for ψ[1] are

$$\begin{aligned} h^F(x) &= 0.25^2 - \|Px - g\|_2^2, \\ h_j^G(x, o_j) &= \|Px - o_j\|_2^2 - 0.175^2, \\ \frac{\partial h^F}{\partial x} &= -2P^T(Px - g), \; \frac{\partial h_j^G}{\partial x} = 2P^T(Px - o_j). \end{aligned} \tag{3.22}$$

Additionally, it is easily verifiable that both control barrier functions above satisfy the conditions for Assumptions 2 and 3. In this case, $h^F(x) \geq 0 \iff x \in [\![\mu]\!]$ and $h_j^G(x, o_j) \geq 0 \iff x \in [\![\omega_j]\!]$. Finally, we need to formalize our test parameter vector d and the space in which it lives. Continuing with Example 3, we will assume the only perturbable objects in the environment are the center locations of

[1]Note multiple other control barrier functions could have been formed for this problem, though for simplicity we provide the ones in equation (3.22).

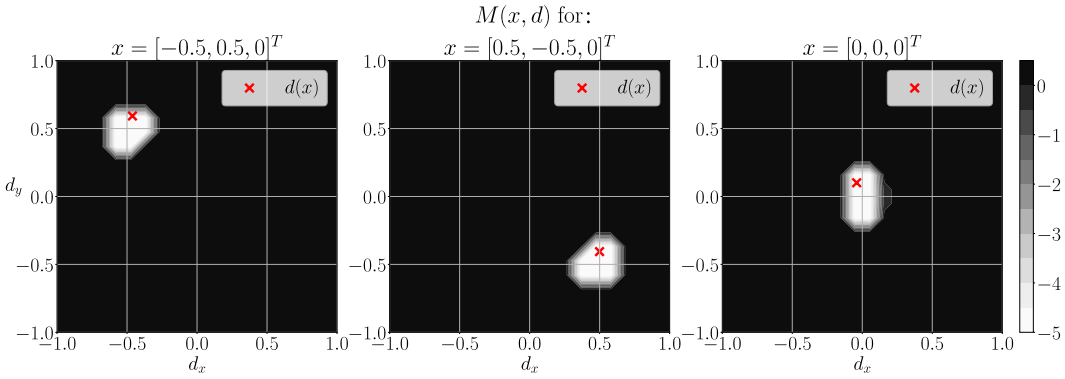

Figure 3.3: Minimization of the difficulty measure M defined in the continuous test-generation examples section provided in this chapter. Notice that in each of the three cases shown, the output of the minimax test synthesizer accurately identifies a test that minimizes the corresponding difficulty measure—the color bar is shown on the right-hand side. This result was theorized in Corollary 1.

our obstacles o_j. We will assume the number of obstacles $N_o = |\mathcal{J}|$, resulting in the following test parameter vector:

$$d = [o_1^T, o_2^T, \dots]^T, \quad d \in \mathcal{D} = [-1, 1]^{2N_o} \subset \mathbb{R}^{2N_o}. \tag{3.23}$$

For this example setting, our minimax testing law $\mathbb{T}$, feasible input space $\mathcal{U}(x, d)$, and difficulty measure M are as follows, with $\mathcal{J} = \{1\}$ as we will show examples with only one obstacle:

$$\mathcal{U}(x,d) = \left\{ u \in \mathcal{U} \;\middle|\; \frac{\partial h_1^G}{\partial x}^T u \geq -10 h_1^G(x, o_1) \right\},$$

$$M(x,d) = \max_{u \in \mathcal{U}} \mathcal{F}\left(\frac{\partial h^F}{\partial x}^T u, u, \mathcal{U}(x,d), -5 \right), \tag{3.24}$$

$$\mathbb{T}(x) = \operatorname*{argmin}_{d \in \mathcal{D}} \max_{u \in \mathcal{U}} \mathcal{F}\left(\frac{\partial h^F}{\partial x}^T u, u, \mathcal{U}(x,d), -5 \right). \tag{3.25}$$

Finally, we note that all optimization problems to be solved in this section will utilize a variant of the algorithm described in [105].

Figure Analysis: It is evident that the example autonomous agent setting described in equation (3.21) with corresponding control barrier functions in equation (3.22) and d defined in equation (3.23) satisfies the conditions for Theorem 2 and Corollary 1. As a result, we would expect that our test synthesizer $\mathbb{T}$ defined in equation (3.25) should always produce a realizable test of system behavior $\forall\, x \in \mathcal{X}$.

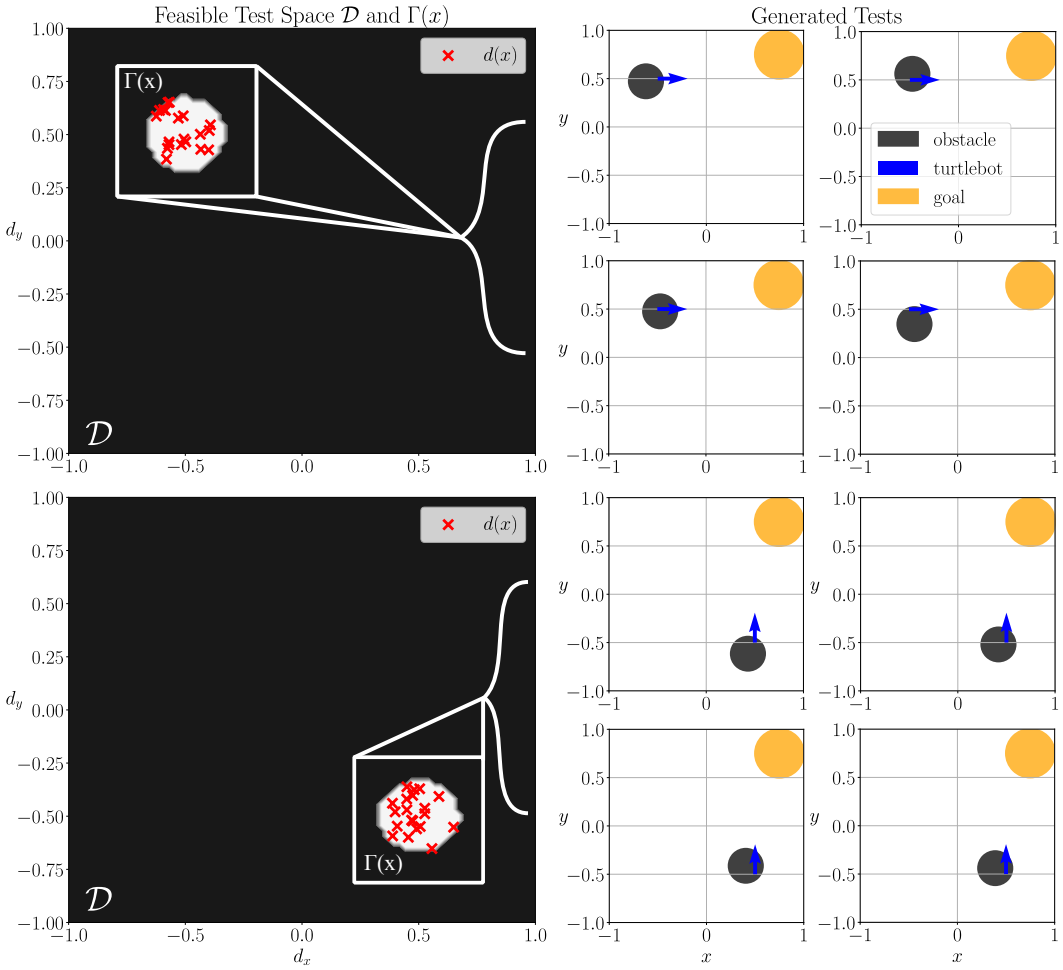

Figure 3.4: Example obstacle placements produced by our test-synthesis procedure defined in Section 3.3 (Top Left) Feasible, partitioned test space. Notice how twenty different solutions to the same minimax problem all yield a test parameter vector inside the white region as theorized in the proof for Theorem 2. (Top Right) Four specific tests generated by the same test synthesizer. (Bottom) Figures showing the same information as above for a different system state.

We also expect this generated test to minimize the difficulty measure M defined in equation (3.24). Figure 3.3 shows this result for three separate system states $x = [-0.5, 0.5, 0]^T$, $x = [0.5, -0.5, 0]^T$, and $x = [0, 0, 0]^T$. Specifically, notice how the red "x" indicating the solution to the test synthesizer (3.25) always lies within the minimizing region of $M(x, d)$ at that state x—the white region in each graph. Indeed, over 1000 randomized runs where the initial state x is perturbed uniformly over the state-space $\mathcal{X}$ in equation (3.21), the test synthesizer finds a test parameter d such that $M(x, d) = -5$, where -5 is the minimum value. Figure 3.4 goes a step farther and shows 8 example tests generated by our test synthesizer. As shown

in the left-hand side figures in Figure 3.4, each of the twenty solutions to the test synthesizer in (3.25) lies in the partitioning set $\Gamma(x)$ within the feasible test space $\mathcal{D}$. Solutions are shown via red "x"-es and $\Gamma(x)$ via the white regions in both left-hand side figures. This phenomenon of the solutions lying within the non-empty set $\Gamma(x)$ is expected via the proof for Theorem 2 as $\Gamma(x) \neq \varnothing$. Specifically, the reason $\Gamma(x) \neq \varnothing$ is that we have not constrained against the obstacle lying on top of the agent to be tested. Due to the agent's limited actuation capacity, overlapping the obstacle with the agent results in an infeasible control barrier function condition, rendering $\mathcal{U}(x, d) = \varnothing$. This fact is corroborated through the 8 example tests shown on the right-hand side. In each of these tests, the obstacle lies atop the agent which is located at the base of the arrow and is heading in the direction the arrow indicates. Ideally, we would like to constrain against such trivial solutions in our test-synthesis framework, and our efforts in that vein will be detailed in a section to follow.

3.4 Discrete-Time Test Generation

Similar to the prior section, this section will state and prove the latter half of our main results: the development of an adversarial, time-varying test-synthesis procedure for discrete-time control systems of the form in (2.6) subject to timed reach-avoid specifications ψ as in (3.2). As before, we will briefly describe the overarching methodology behind our approach, state the developed minimax problem for test synthesis, and end with two, similar theorems to the continuous case.

Overarching Idea: As in the continuous case, Assumption 2 lets us express satisfaction of the reach-avoid specification ψ (3.2) via control barrier functions and their 0-superlevel sets. Specifically, $\forall\ k \in \mathbb{Z}_+$ and $k_{\max} = \min\{k \in \mathbb{Z}_+ \mid t_{\max} \leq k\Delta t\}$ for some $\Delta t \geq 0$, the discrete state trajectory $x_k\ \forall\ k \in \mathbb{Z}_+$ satisfies ψ at $k = 0$, *i.e.* $(x, 0) \models \psi$ if and only if:

$$x_k \in \bigcap_{j \in \mathcal{J}} C_{h_j^G} \text{ and } \exists\ k \in \{0, 1, \ldots, k_{\max}\}\ \text{ s. t. } x_k \in C_{h^F}.$$

As in the continuous setting, we will also make the same assumption on the starting, system state as expressed in equation (3.6):

$$x_0 \in \bigcap_{j \in \mathcal{J}} C_{h_j^G} \text{ and } x_0 \notin C_{h^F}.$$

Here however lies a difference. If we naively used the same control barrier function decrement conditions in Definition 4 to identify inequalities to constrain a minimax problem for test generation, the resulting inner maximization problem would not be

concave. As a result, we would not be able to use Theorem 2 and its proof to gain any insight into this scenario. However, the reason we required concavity of the inner maximization problem was that concavity guaranteed a solution—a feasible control input for a given test parameter vector d. To facilitate the provision of similar guarantees, we will make the following assumption.

Assumption 4. The input space $\mathcal{U}$ for the discrete-time system (2.6) and the feasible test space $\mathcal{D}$ are finite, *i.e.* $|\mathcal{U}| < \infty$ and $|\mathcal{D}| < \infty$.

For context, this assumption is easily satisfied by any system described by a finite-action Markov Decision Process with the space of feasible tests corresponding to edges that can be turned on/off.

In this setting, we can still define a space of feasible inputs $\mathcal{U}(x, d)$ and an action discriminator v as we did for the continuous setting.

$$\mathcal{U}(x, d) = \left\{ u \in \mathcal{U} \;\middle|\; \forall\, j \in \mathcal{J}\ h_j^G(f(x, u), d) \geq 0 \right\}, \tag{3.26}$$

$$v(x, d, u) = h^F(f(x, u), d) - h^F(x, d) - \tau,\ \tau > 0. \tag{3.27}$$

In effect then, our proposed test synthesizer will be very similar to its continuous-time counterpart. Specifically, we will still have an outer minimization problem over a space of feasible tests. Additionally, the goal is to minimize the maximum possible increment in a control barrier function subject to the enduring positivity of multiple other control barrier functions. Keeping these parallels in mind, the statement and results for our proposed, adversarial, time-varying discrete-time test-synthesis procedure will follow.

Statement of Discrete-Time Results

To provide a parallel to the continuous setting, we first define a difference function for the incremental change in a control barrier function after an action.

$$\Delta h(x, u, d) = h\left(f(x, u), d\right) - h(x, d).$$

Then, our proposed test-generation method is as follows:

$$\mathbb{T}(x) = \operatorname*{argmin}_{d \in \mathcal{D}} \max_{u \in \mathcal{U}} \mathcal{F}\left(\Delta h^F(x, u, d), u, \mathcal{U}(x, d), m\right). \tag{3.28}$$

Here, $\mathcal{F}$ is defined in equation (3.15). In the discrete setting, the definition of m changes slightly and will be reproduced here:

$$m \leq \min_{u \in \mathcal{U},\ x \in X,\ d \in \mathcal{D}} \Delta h^F\left(x, u, d\right).$$

As before, we have not formally stated whether such an m exists. However, we will prove its existence in the proofs to follow. Intuitively though, m is defined as the lower bound to a finite series of finite optimization problems, each of which is guaranteed to have a solution. Therefore, so too is m guaranteed to exist. As before, this results in our second theorem which states that minimax problem (3.28) is guaranteed to have a solution $\forall\, x \in \mathcal{X}$.

Theorem 3. *Let Assumption 4 hold. The test synthesizer in* (3.28) *is guaranteed to have a solution $d \in \mathcal{D}$ for every $x \in \mathcal{X}$:*

$$\textit{i.e.,}\ \forall\, x \in \mathcal{X}\ \exists\, d \in \mathcal{D}\ \text{ s.t. }\ d = \mathbb{T}(x).$$

Additionally, we can also define a very similar difficulty measure $\bar{M}$ as to its continuous counterpart M as in equation (3.13):

$$\bar{M}(x, d) = \max_{u \in \mathcal{U}}\ \mathcal{F}\left(\Delta h^F(x, u, d), u, \mathcal{U}(x, d), m\right). \tag{3.29}$$

As before, we need to prove that our proposed difficulty measure satisfies Definition 11. The following lemma expresses this statement.

Lemma 3. *$\bar{M}$ as defined in equation* (3.29) *is a valid difficulty measure as per Definition 11 with feasible input space $\mathcal{U}(x, d)$ as per equation* (3.26) *and action discriminator v as per equation* (3.27)*.*

Finally, with respect to this difficulty measure $\bar{M}$ we have another corollary regarding the optimal difficulty of the generated tests:

Corollary 4. *Let Assumption 4 hold. The test synthesizer in* (3.28) *minimizes the difficulty measure $\bar{M}$ in* (3.29) *over all $d \in \mathcal{D}$, i.e.*

$$\mathbb{T}(x) = \operatorname*{argmin}_{d \in \mathcal{D}}\ \bar{M}(x, d).$$

As before, we will prove these statements in the next section.

Proof of Discrete-Time Results

Similar to Section 3.3, before stating the proofs of both main results in the discrete-time setting, we will first state a useful Lemma.

Lemma 4. *For any non-empty, finite set $A \subset \mathbb{R}$, $\exists\, m, M \in \mathbb{R}$* s.t.*,*

$$m \leq a \leq M,\ \forall\, a \in A.$$

The proof of Theorem 3 will follow.

Proof: This proof amounts to two separate uses of Lemma 4 and will follow a similar partitioning analysis as in the continuous setting. We will group the cases for expediency. Specifically, we can define a barred-Gamma set similar to its counterpart in equation (3.14). With $\mathcal{U}(x, d)$ the feasible input set (3.26),

$$\bar{\Gamma}(x) = \{d \in \mathcal{D} \mid \mathcal{U}(x, d) = \varnothing\}. \tag{3.30}$$

We will also use the notation offered by $\bar{M}$ in equation (3.29) to denote the value of the inner maximization problem in equation (3.28). Then, in the discrete-setting we can rewrite minimax problem (3.28) with $\bar{M}(x, d)$ as follows, based on the definition of $\mathcal{F}$ in equation (3.15):

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}} \begin{cases} \bar{M}(x, d) & \text{if } d \notin \bar{\Gamma}(x) \\ m & \text{else.} \end{cases}$$

Then, the two cases can be resolved simultaneously. If $\bar{\Gamma}(x) = \varnothing$, the above optimization problem collapses to a minimization of $\bar{M}(x, d)$. Each $\bar{M}(x, d)$ is guaranteed to exist via Lemma 4, and as a result, a solution to the larger optimization problem is guaranteed to exist via Lemma 4 as the space of all tests $\mathcal{D}$ is finite. In the event that $\bar{\Gamma}(x) \neq \varnothing$, then any choice of $d \in \bar{\Gamma}(x)$ yields $m \leq \bar{M}(x, d') \; \forall \; d' \in \mathcal{D} \cap \bar{\Gamma}(x)^C$. As such the choice of $d \in \bar{\Gamma}(x)$ solves the above optimization problem. This holds $\forall \; x \in \mathcal{X}$, thus concluding the proof. ■

Likewise, the proof for Lemma 3 will follow.

Proof: We will follow a case-by-case analysis with the cases offered by $\bar{\Gamma}(x)$ defined in equation (3.30). In either case however, as per Theorem 3, we know that there exists a $d \in \mathcal{D} \; \forall \; x \in \mathcal{X}$ that solves minimax problem (3.28). By definition of the proposed difficulty measure $\bar{M}$ in equation (3.29), so to do these solutions also minimize $\bar{M}$. To be clear in the remainder of this proof, we will call these solutions d^* mimicking the notation used in Definition 11. Then, in the event that $\bar{\Gamma}(x) \neq \varnothing$, by the proof for Theorem 3 we know that $d^* \in \bar{\Gamma}(x)$. As a result, $\mathcal{U}(x, d^*) = \varnothing$ by definition of $\bar{\Gamma}(x)$ in equation (3.30). Therefore, $\bar{M}$ satisfies the first condition for being a difficulty measure. In the second case, $\bar{\Gamma}(x) = \varnothing$ and by the proof for Theorem 3 and definition of the action discriminator v in equation (3.27),

$$\begin{aligned} d^* &= \underset{d \in \mathcal{D}}{\operatorname{argmin}} \; \max_{u \in \mathcal{U}(x,d)} \; \Delta h^F(x, u, d), \\ &= \underset{d \in \mathcal{D}}{\operatorname{argmin}} \; \max_{u \in \mathcal{U}(x,d)} \; v(x, d, u) + \tau. \end{aligned}$$

Therefore, in the case where $\bar{\Gamma}(x) = \varnothing$, $\bar{M}$ satisfies the second condition to be a difficulty measure as per Definition 11. ■

The proof of Corollary 4 then stems from Theorem 3 and Lemma 3.

Proof: This is a consequence of Theorem 3 and Lemma 3. ■

Corollaries—Predictive Test Synthesis

As mentioned, the discrete setting also permits us to predict future system states as well. We will show that generating tests in this predictive framework amounts to a simple change in notation, with the majority of the prior section's analysis carrying over. To start, we will assume an arbitrary N-step horizon for predictive test synthesis. To do so requires a few definitions. The first will provide a notational simplification for arbitrary, finite N-step horizon state predictions.

$$\begin{gathered}\mathbf{u} = [u_1, u_2, \ldots, u_N], \quad \mathbf{f}(x, \mathbf{u}, 2) = f(f(x, u_1), u_2), \\ x_{\mathbf{u}}^N = \mathbf{f}(x, \mathbf{u}, N).\end{gathered} \tag{3.31}$$

Then, we can define the set of feasible input sequences and an action discriminator satisfying Definition 10. Here, we note that $\mathbf{u}$ and $x_{\mathbf{u}}^N$ are as defined in equation (3.31), and $\mathcal{U}^N = \mathcal{U} \times \mathcal{U} \ldots$ N times.

$$\mathcal{U}^N(x, d) = \left\{ \mathbf{u} \in \mathcal{U}^N \;\middle|\; \forall\, j \in \mathcal{J},\ h_j^G\left(x_{\mathbf{u}}^N, d\right) \geq 0 \right\}, \tag{3.32}$$

$$\Delta^N h(x, \mathbf{u}, d) = h\left(x_{\mathbf{u}}^N, d\right) - h(x, d), \tag{3.33}$$

$$v(x, d, u) = \Delta^N h^F(x, \mathbf{u}, d) - \tau. \tag{3.34}$$

With these terms, we can propose a predictive test-synthesis procedure that is similar to its one-step counterpart in (3.28). Our proposed test-synthesis procedure and difficulty measure $\tilde{M}$ are as follows:

$$\mathbb{T}(x) = \operatorname*{argmin}_{d \in \mathcal{D}} \max_{\mathbf{u} \in \mathcal{U}^N} \xi^N(x, \mathbf{u}, d), \tag{3.35}$$

$$\tilde{M}^N(x, d) = \max_{\mathbf{u} \in \mathcal{U}^N} \xi^N(x, \mathbf{u}, d), \tag{3.36}$$

$$\xi^N(x, \mathbf{u}, d) = \mathcal{F}\left(\Delta^N h^F(x, \mathbf{u}, d), \mathbf{u}, \mathcal{U}^N(x, d), m\right).$$

As prior, we define m as follows:

$$m \leq \min_{\mathbf{u} \in \mathcal{U}^N,\, x \in \mathcal{X},\, d \in \mathcal{D}} \Delta^N h^F(x, \mathbf{u}, d).$$

Then, the next corollary formally states that the test-synthesis procedure in equation (3.35) is guaranteed to produce realizable tests of system behavior.

Corollary 5. *Let Assumption 4 hold. The test synthesizer in* (3.35) *is guaranteed to have a solution $d \in \mathcal{D}\ \forall\ x \in \mathcal{X}$, i.e.,*

$$\forall\ x \in \mathcal{X},\ \exists\ d \in \mathcal{D}\ \text{ s.t. }\ d = \mathbb{T}(x).$$

Proof: The proof for this corollary follows directly in the footsteps of the proof for Theorem 3. More aptly, for any choice of finite prediction horizon N, we can make the following redefinition:

$$x_{k+1} = x^N_{\mathbf{u}} = \mathbf{f}(x_k, \mathbf{u}, N) = \tilde{f}(x_k, \mathbf{u}). \tag{3.37}$$

This redefinition effectively constructs a new, single-step discrete-time system whose input space $\mathcal{U}^N$ is still finite. Then, Theorem 3 provides the desired result. ■

In a similar fashion, we can also prove that the proposed difficulty measure $\bar{M}^N$ is a valid difficulty measure as per Definition 11.

Lemma 5. *$\tilde{M}^N$ as defined in equation* (3.36) *is a valid difficulty measure as per Definition 11 with feasible input space $\mathcal{U}^N(x, d)$ as per equation* (3.32) *and action discriminator v as per equation* (3.34)*.*

Proof: Following the same redefinition as in equation (3.37), we find that our proposed difficulty measure $\bar{M}^N$ collapses to a one-step difficulty measure where the input space $\mathcal{U}^N$ is still finite. Then, Lemma 3 provides the desired result. ■

In a similar fashion, we can also formally state and prove that the tests generated by minimax problem (3.35) are maximally difficult.

Corollary 6. *Let Assumption 4 hold. The test synthesizer in* (3.35) *minimizes the difficulty measure $\tilde{M}$ in* (3.36) *over all $d \in \mathcal{D}$, i.e.*

$$\mathbb{T}(x) = \underset{d \in \mathcal{D}}{\operatorname{argmin}}\ \tilde{M}^N(x, d).$$

Proof: Again, this Corollary stems directly from Corollary 5. ■

Examples

Figure 3.5 provides a picture for our example in this section—the placement of obstacles on a discrete grid to frustrate an agent's ability to reach its goal. The agent

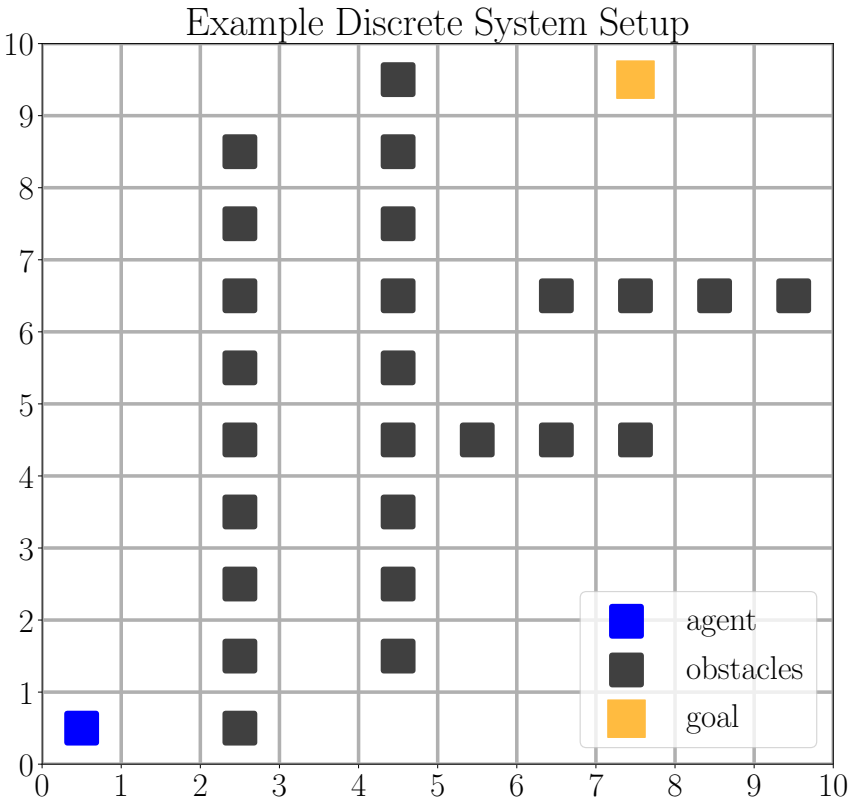

Figure 3.5: Example discrete time setting for the examples referenced in Section 3.4.

is modeled as a discrete transition system:

$$x_{k+1} = \underbrace{\begin{cases} x_k \pm [1,0] & \text{if } u_k = \text{left } (-) \text{ or right } (+) \\ x_k \pm [0,1] & \text{if } u_k = \text{down } (-) \text{ or up } (+) \\ x_k & \text{if } u_k = \text{stay or action infeasible.} \end{cases}}_{f(x_k,u_k)}$$

$$x_k \in \{0,1,2,\ldots,9\}^2 = \mathcal{X},$$

$$u_k \in \{\text{left},\ \text{right},\ \text{up},\ \text{down},\ \text{stay}\} = \mathcal{U}$$

Our test parameter d and specification ψ are as follows, with $g = [g^0, g^1] \in \mathcal{X}$ the goal-cell:

$$d = [d^0, d^1],\ d \in \mathcal{D} \subseteq \mathcal{X}, \text{and } \psi = \mathbf{F}_\infty\, \mu \wedge \mathbf{G}_\infty\, \omega,$$

$$[\![\mu]\!] = \{g\}\,,\ [\![\omega]\!] = \left\{x \in \mathcal{X} \,\middle|\, x \neq d\right\}.$$

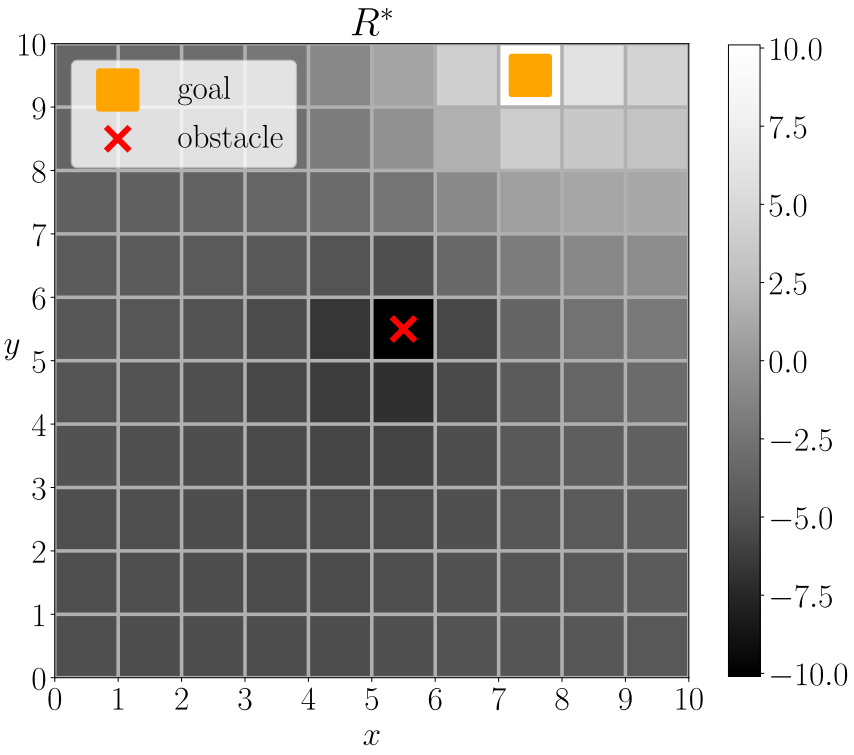

Figure 3.6: Depiction of the gradient function R^* utilized to generate example control barrier functions, with an obstacle placed at the center for depiction purposes.

To construct a variable discrete barrier function then, we will first define a reward matrix $R(d)$ inspired by the work done in [60].

$$
\begin{aligned}
R(d) = \underset{V \in \mathbb{R}^{10\times 10}}{\operatorname{argmax}} \quad & \mathbf{1}^T V \mathbf{1} \\
\text{subject to} \quad & V[g^0, g^1] = 10, \\
& V[d^0, d^1] = -10, \\
& V[x_k^0, x_k^1] = \sum_{u \in \mathcal{U}} 0.2 V[x_{k+1}^0, x_{k+1}^1], \\
& \forall\, x_k \in \mathcal{X},\ x_k = [x_k^0, x_k^1].
\end{aligned} \tag{3.38}
$$

We note that the optimization problem in equation (3.38) is almost always solvable. For more curious readers, please reference hitting times and absorption probabilities for Markov Chains in [106]. The only cases precluding a solution occur when the two sets overlap, *i.e.* the goal overlaps with at least one obstacle, yielding an inconsistent feasible set. As such, we will modify this reward matrix $R(d)$ to generate our barrier

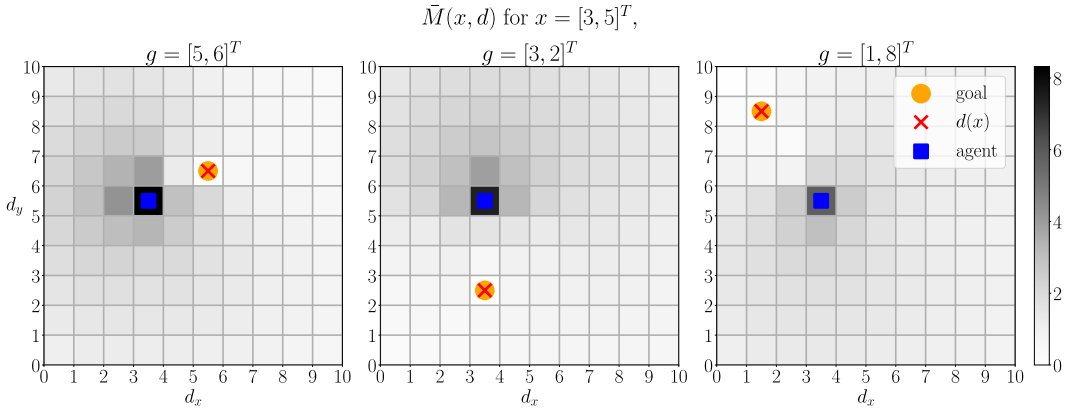

Figure 3.7: Shown above is a series of color plots indicating the difficulty measure defined in the discrete-time examples subsection. The difficulty measure varies with goal location as it is influenced by a barrier function dependent on the goal location. As can be seen in each of the three cases above, however, the test synthesizer accurately identifies a test that minimizes this difficulty measure—it places the obstacle shown via the red "x" over the goal. The associated color bar is to the right.

functions, *i.e.* $R^*(d) \in \mathbb{R}^{10\times 10}$, and

$$R^*(d)[i,j] = \begin{cases} 0 & \text{if (3.38) is infeasible,} \\ 10.1 & \text{if } i = g^0,\ j = g^1, \\ -10.1 & \text{if } i = d^0,\ j = d^1, \\ R(d)[i,j] & \text{else.} \end{cases}$$

$$h^F(x,d) = R^*(d)[x^0, x^1] - 10,$$
$$h^G(x,d) = R^*(d)[x^0, x^1] + 10.$$

Figure 3.6 depicts our resulting $R^*(d)$ after this procedure, for an example case where $d = [5,5]$ and the goal $g = [7,9]$. Then, the specific versions of our feedback law and difficulty measure are as follows, with d representing the grid cell location of our single obstacle:

$$\mathcal{U}(x,d) = \{u \in \mathcal{U} \mid h^G(f(x,u),d) \geq 0\},$$
$$\bar{M}(x,d) = \max_{u\in\mathcal{U}} \mathcal{F}\left(\Delta h^F(x,u,d), u, \mathcal{U}(x,d), -15\right), \tag{3.39}$$
$$\mathbb{T}(x) = \operatorname*{argmin}_{d\in\mathcal{D}} \max_{u\in\mathcal{U}} \mathcal{F}\left(\Delta h^F(x,u,d), u, \mathcal{U}(x,d), -15\right). \tag{3.40}$$

Figure Analysis: Over 1000 randomized trials wherein the system's initial state and goal are chosen randomly such that they don't overlap, the test synthesizer (3.40)

satisfactorily identifies a test parameter d for each case such that d minimizes the difficulty measure $\bar{M}$ in (3.39) at that state x. This should also be expected as the proofs in the discrete-time case mirror their continuous counterparts and the continuous feedback law also exhibited a similar capacity to identify realizable and maximally difficult tests. To that end, Figure 3.7 portrays both the optimal obstacle setup overlaid on the difficulty measure contour map for the state/goal pair listed. As can be seen in each of the three cases shown, the test synthesizer (3.40) accurately identifies an obstacle location that minimizes the corresponding difficulty measure—the color bar indicating evaluations of the difficulty measure is shown on the right-hand side. Additionally, we can see that in each of the cases, the optimal obstacle location is to place the obstacle directly on top of the goal. We expect this behavior to be the most difficult test, as in this scenario, there is nothing the system could ever do to reach its goal while simultaneously satisfying its safety specification. In the following section, we will revisit this example and constrain against such tests to yield a more useful outcome.

3.5 Extensions—Constrained Test Synthesis

In this section, we will extend our prior results on the guaranteed realizability and maximal difficulty of our test-synthesis procedure(s), when the space of feasible tests $\mathcal{D}$ is a function of time and the system state. This lets us constrain against trivial test cases such as (but not limited to) obstacles overlapping with the agent/goal.

Corollaries for Constrained Testing

To start, we assume there exists a set-valued function that maps from the state space $\mathcal{X}$ and time t to $\mathbb{R}^p$.

$$\mathbb{D} : \mathcal{X} \times \mathbb{R}_+ \to 2^{\mathbb{R}^p} \;\text{ s. t. }\; \mathbb{D}(x,t) = \mathcal{D} \subset \mathbb{R}^p.$$

This results in the following change to the test synthesizer for the continuous setting in equation (3.12):

$$\mathbb{T}(x,t) = \underset{d \in \mathbb{D}(x,t)}{\operatorname{argmin}} \; \max_{u \in \mathcal{U}} \; \mathcal{F}\left(\dot{h}^F(x,d,u), u, \mathcal{U}(x,d), m\right), \tag{3.41}$$

$$m \leq \min_{u \in \mathcal{U},\, x \in \mathcal{X},\, d \in \mathbb{D}(x,t)} \dot{h}^F(x,d,u).$$

In the discrete setting, we will directly mention the change with respect to the predictive test-synthesis framework. Our test-synthesis procedure is as follows,

with Δ as defined in equation (3.33).

$$\begin{aligned}
\mathbb{T}(x,t) &= \underset{d\in\mathbb{D}(x,t)}{\operatorname{argmin}} \max_{\mathbf{u}\in\mathcal{U}^N} \xi(x,\mathbf{u},d), \\
\xi(x,\mathbf{u},d) &= \mathcal{F}\left(\Delta h^F(x,\mathbf{u},d), \mathbf{u}, \mathcal{U}^N(x,d), m\right), \\
m &\leq \min_{\mathbf{u}\in\mathcal{U}^N, x\in\mathcal{X}, d\in\mathbb{D}(x,t)} \Delta h^F(x,\mathbf{u},d).
\end{aligned} \tag{3.42}$$

As the space of feasible tests is now variable, we need to change the statements of Assumptions 3 and 4 to match. The analog of Assumption 3 is as follows.

Assumption 5. Each feasible test space $\mathcal{D} \in \mathcal{R}(\mathbb{D})$ is a compact set, the input space $\mathcal{U}$ for the continuous time system (2.1) is a closed, convex polytope in $\mathbb{R}^m$, and $h^F, h^G_j \in C^1(\mathcal{X}\times\mathcal{D})$.

Likewise, the analog of Assumption 4 is as follows.

Assumption 6. The input space $\mathcal{U}$ for the discrete-time system (2.6) is finite, and each feasible test space $\mathcal{D} \in \mathcal{R}(\mathbb{D})$ is also finite.

Then, we can state and prove the following corollaries guaranteeing the realizability of the tests generated by each feedback law.

Corollary 7. *Let Assumption 5 hold. The test synthesizer in* (3.41) *is guaranteed to have a solution* $d \in \mathbb{D}(x,t)\ \forall\ x \in \mathcal{X},\ t \in \mathbb{R}_+$*, i.e.,*

$$\forall\ x \in \mathcal{X},\ t \in \mathbb{R}_+\ \exists\ d \in \mathbb{D}(x,t)\ \text{ s.t. }\ d = \mathbb{T}(x,t).$$

Proof: This is an application of Theorem 2 for each $\mathcal{D} \in \mathcal{R}(\mathbb{D})$ which is assumed to be compact via Assumption 5. ■

Corollary 8. *Let Assumption 6 hold. The test synthesizer in* (3.42) *is guaranteed to have a solution* $d \in \mathbb{D}(x,t)\ \forall\ x \in \mathcal{X},\ t \in \mathbb{R}_+$*, i.e.,*

$$\forall\ x \in \mathcal{X},\ t \in \mathbb{R}_+\ \exists\ d \in \mathbb{D}(x,t)\ \text{ s.t. }\ d = \mathbb{T}(x,t).$$

Proof: This is an application of Corollary 5 for each $\mathcal{D} \in \mathcal{R}(\mathbb{D})$ which is assumed to be finite via Assumption 6. ■

These modified test-generation laws also minimize their respective difficulty measures over the constrained test-generation set $\mathbb{D}(x,t)$.

Corollary 9. *Let Assumption 5 hold. The test synthesizer* (3.41) *minimizes the difficulty measure* M (3.13) *over all* $d \in \mathbb{D}(x,t)$*, i.e.*

$$\mathbb{T}(x) = \underset{d \in \mathbb{D}(x,t)}{\operatorname{argmin}} \; M(x,d).$$

Proof: This proof stems from the application of Corollary 1 $\forall\, \mathcal{D} \in \mathcal{R}(\mathbb{D})$. ■

In the discrete setting, we have the following corollary.

Corollary 10. *Let Assumption 6 hold. The test synthesizer* (3.42) *minimizes the difficulty measure* $\tilde{M}$ (3.36) *over all* $d \in \mathbb{D}(x,t)$*, i.e.*

$$\mathbb{T}(x) = \underset{d \in \mathbb{D}(x,t)}{\operatorname{argmin}} \; \tilde{M}(x,d).$$

Proof: This proof stems via repeated application of Corollary 6 for each finite set $\mathcal{D} \in \mathcal{R}(\mathbb{D})$. ■

Remark on Environment Dynamics: Sometimes, there may exist torque bounds on $\dot{d}$, *e.g.* environment dynamic constraints, and as stated the mentioned approach could not account for such constraints. However, one could augment the system state to include the modelable state of the environment, *e.g.* the overall "state" x would be the state of the system-under-test x_s and the state of the modelable aspect of the environment subject to dynamic constraints d_t. Then, the test-generation procedure would provide the dynamics d that the modelable environment states d_t must follow. See [107] for work in this vein.

Applications to a Constrained Hardware Test

The constrained test-synthesis results permit us to start applying our procedure to testing hardware systems in their operating environments. Specifically, we test a quadruped's ability to navigate to a goal while avoiding moving robots attempting to block its path. To start, we idealize the quadruped as a single integrator system:

$$\dot{x} = u,\; x \in \mathcal{X} \triangleq [-1,4] \times [-2,3],\; u \in \mathcal{U} \triangleq [-5,5]^2.$$

To construct our specification ψ and our test parameter vector d, we will denote the goal as $g = [3.5, 2.5]^T$ and the obstacle agent locations on the 2-d plane as $o_j \in \mathcal{X}$. Then ψ is as follows:

$$[\![\mu]\!] = \{x \in \mathbb{R}^2 \mid \|x - g\| \leq 0.3\},$$
$$[\![\omega_j]\!] = \{x \in \mathbb{R}^2 \mid \|x - o_j\| \geq 0.3\},$$
$$\psi = \mathbf{F}_\infty \mu \wedge_{j=1,2} \mathbf{G}_\infty \omega_j.$$

The combined locations of these obstacles will be our test parameter vector, *i.e.* $d = [o_1^T, o_2^T]^T \in \mathcal{X}^2$.

To generate tests, our barrier functions h^F and h_j^G and test space map $\mathbb{D}$ are as follows (here x is the quadruped's planar position, *i.e.* $x = [x_1, x_2] \in \mathcal{X}$ and for a scalar $\epsilon \in \mathbb{R}$, $\lfloor \epsilon \rfloor$ denotes rounding down and $\lceil \epsilon \rceil$ denotes rounding up):

$$h^F(x) = 0.3 - \|x - g\|,\ h_j^G(x, d) = \|x - o_j\| - 0.3,$$
$$d = [o_1^T, o_2^T]^T \in \mathbb{D}(x) \triangleq \{\lfloor x_1 \rfloor, \lfloor x_2 \rfloor\} \times \{\lceil x_1 \rceil, \lceil x_2 \rceil\}.$$

As an example then $\mathbb{D}(x = [0.3, 1.7]) = \{0, 1\} \times \{1, 2\}$. For testing purposes, we will calculate the optimal obstacle locations offline and direct the obstacles to move between the identified grid points at test time. Theoretically, the identified grid points should correspond to the most difficult test of quadruped behavior while it ambulates within a grid cell—asking the obstacles to move between grid points as the quadruped moves between grid cells should likewise be more difficult. We repeated the experiment twice, and recorded the minimum value of both barrier functions:

$$h(x, d) = \min_{j=1,2}\ h_j^G(x, d),$$

for the entire multi-system trajectory. The corresponding time-series data is shown in Figure 3.8.

Figure Analysis: In both tests, the quadruped tries "cutting" corners between grid cells, as at least one of the obstacles moves to the associated grid point that the quadruped is trying to move through. This causes a momentary loss of safety as evidenced by the sharp spike where the minimum barrier value goes negative (around 20 seconds). The quadruped quickly corrects this mistake and resumes its normal trajectory while repeating this cutting behavior a few more times. That being said, it maintains a positive barrier value both times. This references the "dip" in the yellow trajectory around 45 seconds, and the two "dips" in the blue trajectory around 60 and 70 seconds. Figure 3.8 depicts this momentary loss of safety. This procedure, however, shows an example implementation of our worst-case tests on hardware systems. Additionally, it shows that the tests that we theorize to be the most difficult uncover problematic system behavior as required of our testing procedure.

3.6 Conclusion

This chapter detailed our efforts to develop an adversarial test synthesis approach for autonomous systems based on control barrier functions and timed reach-avoid

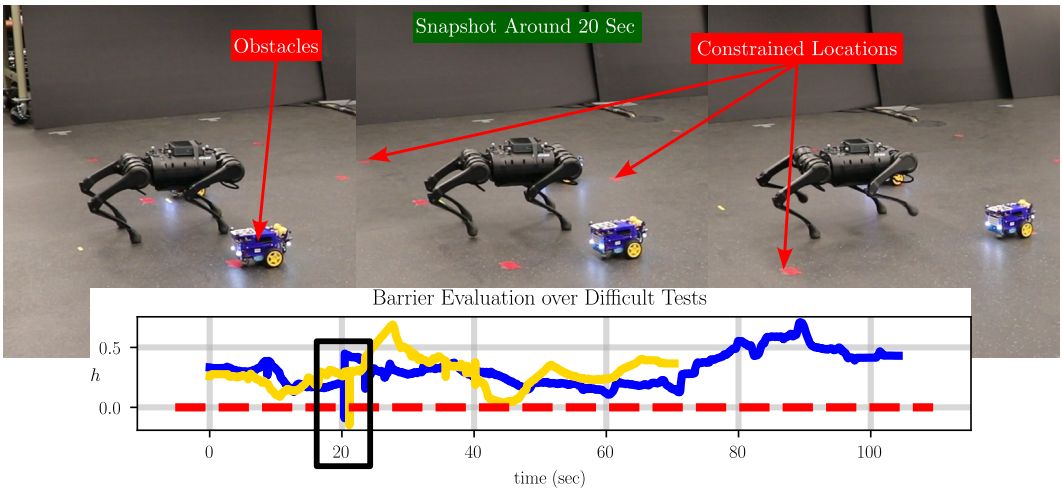

Figure 3.8: Depiction of the safety failure identified during the implementation of the constrained testing procedure described in Section 3.5. The goal is to determine moving obstacle locations defined at the vertices of a 5×5 grid, while a quadruped ambulates to a goal (off-screen). Shown above are depictions of the trial whose time-series data for the composite barrier function is shown in yellow. Around 20 seconds, we see the quadruped (Left) start to try to cut corners between grid cells. (Middle) This sends a signal to our test-synthesis procedure to ask the obstacle to move to cut off its path. (Right) The quadruped reacts to the moving obstacle, but slowly, causing the momentary lapse in safety as signified by the sharp spike in the barrier function going negative. The obstacle waypoints were chosen by our test-synthesis technique. Repeating this experiment from the same starting procedure once more, yielded similar behavior as signified by the blue trajectory.

specifications. We proved that our approach will always produce realizable and maximally difficult tests of system behavior as our synthesis techniques are guaranteed to have solutions that minimize a corresponding difficulty measure—a concept we introduce and define. Finally, we show the efficacy of our procedure in generating tests for simple toy examples useful in a real-world context—unicycle systems and grid-world abstractions, both of which are used for baseline navigation control algorithms in other works. We also show how such an abstraction can easily be extended to a useful hardware system test—testing a quadruped's ability to navigate within a grid while avoiding obstacles. However, just because a system passes a difficult test does not equate to a verification statement. As such, the chapters to follow will detail efforts made to streamline risk-aware verification and controller synthesis. To facilitate the statement of these contributions, the next chapter details our efforts in uncertainty quantification.

Chapter 4

UNCERTAINTY QUANTIFICATION

This chapter was adapted from:

[1] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-Based Bounds for Coherent Risk Measures: Applications to Policy Synthesis and Verification," *The Artificial Intelligence Journal (Under Review)*, Apr. 2022. DOI: 10.48550/arXiv.2204.09833. arXiv: 2204.09833 [cs.AI],

[2] P. Akella and A. D. Ames, "Bounding Optimality Gaps for Non-Convex Optimization Problems: Applications to Nonlinear Safety-Critical Systems," *62nd Conference on Decisions and Control (Submitted)*, Apr. 2023. DOI: 10.48550/arXiv.2304.03739. arXiv: 2304.03739 [math.OC],

[3] P. Akella, M. Ahmadi, and A. D. Ames, "A Scenario Approach to Risk-Aware Safety-Critical System Verification," *arXiv e-prints*, arXiv:2203.02595, Mar. 2022. DOI: 10.48550/arXiv.2203.02595. arXiv: 2203.02595 [eess.SY],

As mentioned prior, this chapter and the chapters to follow will focus on the verification pipeline more broadly. Specifically, we motivated earlier in the introduction that one of the primary hindrances to theoretical advancements in verification centers on our inability to calculate risk measures over system state trajectories. This inability arises as we typically do not have perfect knowledge of the uncertainties affecting system evolution, and as such, the distribution over realized paths is similarly unknown. Therefore, before formalizing a pipeline for risk-aware verification, this chapter details our efforts in uncertainty quantification. Specifically, it details our procedure to bound risk measures for random variables whose distributions are unknown, the application of the same principles to rapidly generate percentile solutions for non-convex optimization problems, and how multiple applications of the percentile method produce a bound on the optimality gap of the reported percentile solution. To start, we will provide a more in-depth analysis of existing work that highlights the need for the risk-aware contributions we detail herein.

4.1 Introduction

The problem of optimal policy generation under uncertainty has been well-studied in the learning community, most notably via Reinforcement Learning [108]–[112].

In this setting, the agent-environment interaction is modeled via a Markov Decision Process (MDP) where the state of the agent-environment pair is assumed to lie in some finite set [113]. Then, the agent's action at an initial state determines the set of states from which the successive state is randomly sampled according to some distribution. Each such transition is assigned a reward via a reward function, resulting in the traditional Reinforcement Learning policy generation problem—determining a policy that maximizes the expected, time-discounted reward achievable by the agent undergoing such uncertain transitions. The Partially Observable Markov Decision formulation of this problem is considered in [114]–[117].

However, for safety-critical control applications, expectation-maximization policies fail to consider variances in outcomes which could lead to catastrophic behavior [42]. This underscores the need to formally consider risk in both policy development and verification. To that end, Majumdar *et al.* [43] argue that coherent risk measures [44] like Conditional-Value-at-Risk (CVaR) and Entropic-Value-at-Risk (EVaR) which underlie the treatment of financial risk utilized by the vast majority of the world's banks [45], should similarly underlie the formal treatment of risk in robotics. This paradigm is also shared broadly within the learning and control communities, since these g-entropic risk measures like CVaR and EVaR exhibit nice mathematical properties, *e.g.* convexity, and permit a formal assessment of risk previously unaccounted for by non-coherent risk measures like Value-at-Risk. In short, this has led to the widespread adoption of these coherent risk measures as the de facto standard for formal consideration of risk in policy generation [43], [118]–[124].

While there exist a plethora of policy synthesis techniques in a risk-sensitive setting, there exist few verification techniques—especially for g-entropic risk measures—that account for unstructured uncertainty. For example, numerous works propose risk-aware verification procedures for specific systems [125]–[127]. These methods verify their systems of interest against existing widespread standards, *e.g.* in [127] the authors verify a multi-agent collaborative robotic system against the international standards for safe human-robot interaction, ISO 10218 [128], [129]. As such, the verification analyses in these works are limited to their specific systems of interest, and risk is typically defined against the corresponding standard. In effect, these works do not use the same notion of risk as utilized in policy development. For more abstract, black-box approaches to verification, Corso *et al.* provide a very nice survey of existing techniques [130]. In short, however, the existing verification techniques that account for uncertain system measurements, *e.g.* Bayesian Optimization [131],

[132] or Reinforcement Learning [133], [134], follow the same expectation-specific analysis that prompted interest in a risk-aware approach.

Motivating Questions and Chapter Contributions

This inability to address risk-aware verification through the same risk measures utilized for policy development stems primarily from the inability to evaluate these measures over unknown probability distributions. While there exist concentration inequalities for specific coherent risk measures [135]–[138], there do not exist similar bounds for other risk measures, *e.g.* Value-at-Risk, Entropic-Value-at-Risk, *etc*. Furthermore, traditional risk-aware policy synthesis schemes typically assume *apriori* knowledge of the uncertainties affecting system evolution, though practically we have little knowledge of these distributions. To address our inability to evaluate risk measures for unknown probability distributions and lay the foundation for risk-aware verification and synthesis in the next chapter, we detail the following contributions in this chapter:

1. We detail our efforts in the provision of procedures providing probabilistic upper bounds for Value-at-Risk and any g-entropic risk measure — a large subset of coherent risk measures.

2. We develop a percentile optimization procedure that leverages the prior risk-measure estimation method to provide percentile solutions to a wide class of non-convex optimization problems. Specifically, for an optimization problem with decision variables s in a decision space $\mathbb{S}$, a percentile solution s^* in the 90%-ile outperforms 90% of decisions $\mathrm{s}' \in \mathbb{S}$ with respect to optimizing for the objective in question.

3. Finally, we show how multiple applications of the same percentile approach can bound the optimality gap of percentile solutions to a wide class of non-convex optimization problems.

Chapter Structure

To start, Section 4.2 reviews scenario optimization and introduces the risk measures studied. Then, Section 4.3 details our efforts in upper-bounding risk-measure evaluation for scalar random variables whose distributions are unknown. Section 4.4 leverages these upper-bounding results to provide a method that outputs percentile solutions to a wide class of non-convex optimization problems. Finally, Section 4.5

details how one can apply the same percentile optimization technique twice to also bound the optimality gap of a reported percentile solution.

Remark on a purely probabilistic viewpoint: A central approach taken in this chapter will be to classify outcomes of the random variable of interest as either useful — greater than a risk-measure of interest — or not. As such, the work detailed in this chapter can also be viewed from a purely probabilistic lens, if the current treatment, which caters more to the robotics and controls community, is found wanting. Intuitively, the detailed approach to risk-measure bounding can be viewed as taking enough samples of a random variable until we receive a sample that exceeds a cutoff value of interest. This is an event known to occur with non-zero probability even if we do not know the underlying distribution, as this non-zero probability arises via the definition of these risk measures. As such, identifying such a sample is equivalent to transforming risk-measure estimation into a Bernoulli random variable sampling problem with known success probabilities. Therefore, you will see the corresponding confidence of $1-(1-\epsilon)^N$ arising in most theoretical results in this chapter, where ϵ corresponds to the success probability for the corresponding Bernoulli random variable and N the number of samples taken of such a variable while searching for success. Keeping this intuitive approach in mind, the chapter can be viewed as a guide on how to construct this Bernoulli random variable for each risk measure studied and for any scalar random variable X.

4.2 Reviewing Scenario Optimization and Risk Measures

Scenario Optimization

Scenario optimization identifies robust solutions to uncertain convex optimization problems of the following form [139]:

$$\begin{aligned} \mathrm{q}^* = \underset{\mathrm{q}\in\mathrm{Q}\subset\mathbb{R}^d}{\operatorname{argmin}} \quad & c^T\,\mathrm{q}, \\ \text{subject to} \quad & \mathrm{q}\in\mathrm{Q}_\delta,\ \delta\in\Delta. \end{aligned} \tag{UP}$$

Here, (UP) is an uncertain program as δ is a sample of some random variable in the probability space $\Sigma = (\Delta, \mathcal{F}, \mathbb{P})$. Here, Δ is the sample space, $\mathcal{F}$ is the (perhaps unknown) event space, and $\mathbb{P}$ is the (perhaps) unknown probability measure. Convexity is assured via an assumed convexity in the constraint spaces Q and Q_δ, where the latter is the subspace of Q defined by the associated sample $\delta \in \Delta$. Furthermore, since Δ is typically a set of infinite cardinality, *i.e.* $|\Delta| = \infty$, identification of a solution q^* such that $\mathrm{q}^* \in \mathrm{Q}_\delta\ \forall\ \delta \in \Delta$ is typically infeasible.

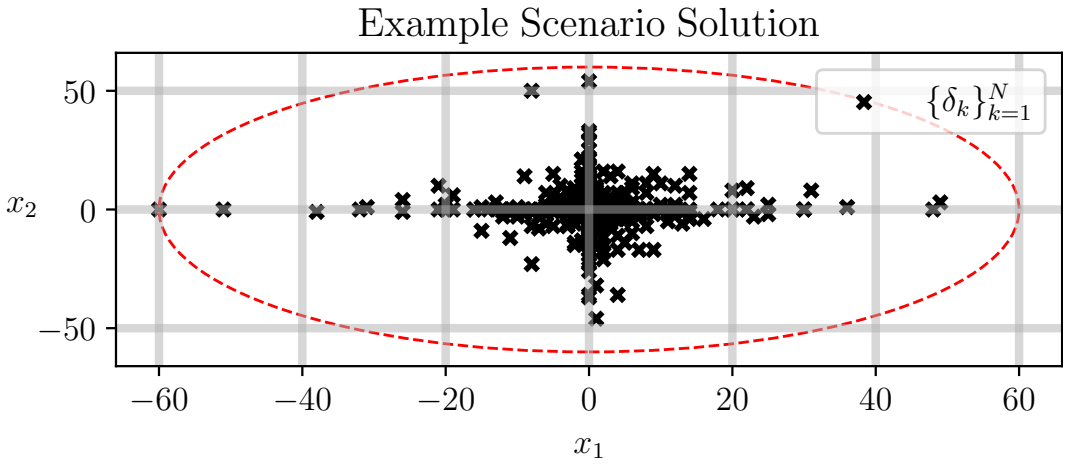

Figure 4.1: An example of using scenario optimization to calculate the smallest circle radius required to encapsulate at least 0.987 of the probability mass of the underlying random variable with which samples were taken. In this case, samples δ are the randomly sampled points on the 2-d plane shown in black, the overarching decision space is the positive reals, and the sample-specific constraint spaces $\mathrm{Q}_\delta = \{r \in \mathbb{R}_+ \mid r \geq \|\delta\|\}$. Hence, the specific uncertain program would be to minimize the radius r subject to $r \in \mathrm{Q}_\delta$ for all sample-able δ.

To resolve this issue, scenario optimization solves a related optimization problem formed from a set of N samples δ and provides a probabilistic guarantee on the robustness of the corresponding solution q_N^*. Specifically, given an N-sized set of samples $\{\delta_\mathrm{j}\}_{\mathrm{j}=1}^{\mathrm{N}}$, we could construct the following scenario program:

$$\begin{aligned} \mathrm{q}_N^* = \underset{\mathrm{q}\in\mathrm{Q}\subset\mathbb{R}^d}{\operatorname{argmin}} \quad & c^T\,\mathrm{q}, \\ \text{subject to} \quad & \mathrm{q} \in \mathrm{Q}_{\delta_i},\ \forall\, \delta_i \in \{\delta_\mathrm{j}\}_{\mathrm{j}=1}^{\mathrm{N}}\,. \end{aligned} \tag{RP-N}$$

Then, we require the following assumption.

Assumption 7. The scenario program (RP-N) is solvable for any N-sample set $\{\delta_\mathrm{j}\}_{\mathrm{j}=1}^{\mathrm{N}}$ and has a unique solution q_N^*.

Assumption 7 guarantees the existence of a scenario solution q_N^* for (RP-N) for any provided sample set $\{\delta_\mathrm{j}\}_{\mathrm{j}=1}^{\mathrm{N}}$. As such, we can define a set containing those samples $\delta \in \Delta$ for which the scenario solution q_N^* does not lie in the corresponding constraint set Q_δ, *i.e.* $F(\mathrm{q}) = \{\delta \in \Delta \mid \mathrm{q} \notin \mathrm{Q}_\delta\}$. With this set definition we can formally define the *violation probability* of our solution.

Definition 12. The *violation probability* $V(\mathrm{q})$ is defined as the probability of sampling a constraint δ to which q is not robust, *i.e.* $V(\mathrm{q}) = \mathbb{P}[\delta \in F(\mathrm{q})]$.

Then, we have the following theorem:

Theorem 4 (Adapted from Theorem 1 in [139])**.** *Let Assumption 7 hold. The following inequality is true, with V as per Definition 12:*

$$\mathbb{P}^N \left[V\left(\mathrm{q}_N^*\right) > \epsilon \right] \leq \sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}.$$

In Theorem 4 above, N is the number of sampled constraints δ for the scenario program (RP-N), q_N^* is the scenario solution to the corresponding scenario program, $V(\mathrm{q}_N^*)$ is the violation probability of that solution as per Definition 12, d is the dimension of the decision variable q, and $\mathbb{P}^N$ is the induced probability measure over sets of N-samples of δ given the probability measure $\mathbb{P}$ for δ. An example scenario solution is shown in Figure 4.1.

g-Entropic Risk Measures

Risk measures φ map scalar random variables X to the real line. More accurately, consider a probability space $(\Omega, \mathcal{F}, P)$ with the sample space Ω, the event space $\mathcal{F}$, and a probability measure P. A scalar random variable (R.V.) X is a mapping from the sample space to the real-line, *i.e.* $X : \Omega \to \mathbb{R}$. The space of all scalar random variables defined for this probability space is $\tilde{X} = \{X \mid X : \Omega \to \mathbb{R}\}$. For $p \geq 1$, we define L_p as the space of all random variables with p-bounded expectation, *i.e.* $L_p = \{X \in \tilde{X} \mid \mathbb{E}[|X|^p] < \infty\}$, and L_∞ is the set of all bounded scalar random variables. Then, a risk measure φ maps from a subset $\mathbf{X} \subseteq \tilde{X}$ to the extended real-line $\mathbb{R} \cup \{-\infty, \infty\}$, *i.e.* $\varphi : \mathbf{X} \to \mathbb{R} \cup \{-\infty, \infty\}$. A coherent risk measure is defined as follows[1] [44].

Definition 13. A *coherent risk measure* $\varphi : \mathbf{X} \subseteq \tilde{X} \to \mathbb{R} \cup \{-\infty, \infty\}$ satisfies the following four properties:

1. Translation Invariance: $\varphi(X + c) = \varphi(X) + c$,
2. Sub-Additivity: $\varphi(X_1 + X_2) \leq \varphi(X_1) + \phi(X_2),\ X_1, X_2 \in \mathbf{X}$,
3. Monotonicity: If $X_1, X_2 \in \mathbf{X}$ and $X_1(\omega) \leq X_2(\omega)\ \forall\ \omega \in \Omega$, then $\varphi(X_1) \leq \varphi(X_2)$,

[1] Typically, we see the translation invariance property in Definition 13 written as $\varphi(X + c) = \varphi(X) - c$ and similarly, the monotonicity property written as $\varphi(X_1) \geq \varphi(X_2)$ subject to the same conditions. We make the aforementioned changes to align with our definitions of CVaR and EVaR as per Definitions 16 and 17, which represents a shift in the intuitive meaning of these measures as compared to the financial literature.

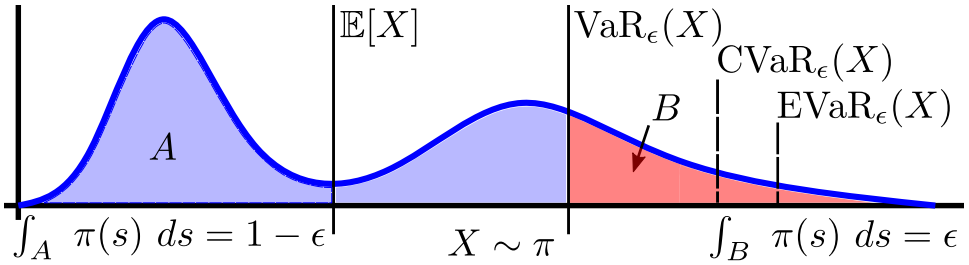

Figure 4.2: Example of three common risk measures at risk-level $\epsilon \in (0, 1]$—Value-at-Risk ($\mathrm{VaR}_\epsilon(X)$), Conditional-Value-at-Risk ($\mathrm{CVaR}_\epsilon(X)$), and Entropic-Value-at-Risk ($\mathrm{EVaR}_\epsilon(X)$)—for a scalar random variable X.

4. Positive Homogeneity: $\varphi(\lambda X) = \lambda\varphi(X),\ \forall\, X \in \mathbf{X},\ \lambda \geq 0.$

Ahmadi-Javid [140] defines g-entropic risk measures as those risk measures satisfying the following definition (Adapted from Definition 5.1 in [140]):

Definition 14. Let $g : \mathbb{R} \to \mathbb{R}$ be a convex function with $g(1) = 0$ and let $\beta \geq 0$. The *g-entropic risk measure with divergence level β* $\mathrm{ER}_{g,\beta}$ is defined as follows, with P the probability measure for X, "$Q \ll P$" denoting absolute continuity of Q with respect to P, and $\frac{dQ}{dP}$ the Radon-Nikodym derivative:

$$\mathrm{ER}_{g,\beta}(X) \triangleq \sup_{Q \in \mathcal{P}} \mathbb{E}_Q[X],$$
$$\mathcal{P} = \left\{ Q \,\middle|\, Q \ll P,\ \int g\left(\frac{dQ}{dP}\right) dP \leq \beta \right\}.$$

All such g-entropic risk measures are coherent risk measures as expressed in Theorem 3.2 in [140]. Of more immediate use, however, will be their representation as infimum problems, as mentioned in the following theorem.

Theorem 5 (Adapted from Theorem 5.1 in [140])**.** *Let g be a closed convex function with $g(1) = 0$ and $\beta \geq 0$. For a random variable $X \in L_\infty$ with probability measure P and a g-entropic risk measure $\mathrm{ER}_{g,\beta}$ as per Definition 14, the following equivalency holds with g^* the convex-conjugate [141] of g:*

$$\mathrm{ER}_{g,\beta}(X) = \inf_{t>0,\mu\in\mathbb{R}} t\left[\mu + \mathbb{E}_P\left[g^*\left(\frac{X}{t} - \mu + \beta\right)\right]\right] \tag{4.1}$$

Finally, a few common risk measures—Value-at-Risk (VaR), Conditional-Value-at-Risk (CVaR), and Entropic-Value-at-Risk (EVaR)—are defined below.

Definition 15. The *Value-at-Risk* level $\epsilon \in [0, 1]$, denoted as $\text{VaR}_\epsilon(X)$, is the infimum over $\zeta \in \mathbb{R}$ of ζ such that samples x of the scalar random variable X with distribution π lie below ζ with probability greater than or equal to $1 - \epsilon$, *i.e.*

$$\text{VaR}_\epsilon(X) = \inf\{\zeta \mid \mathbb{P}_\pi[x \leq \zeta] \geq 1 - \epsilon]\}.$$

Definition 16. The *Conditional-Value-at-Risk* level $\epsilon \in (0, 1]$, denoted as $\text{CVaR}_\epsilon(X)$, is the expected value of all samples x of the scalar random variable X with distribution π that are greater than or equal to the Value-at-Risk level ϵ for X, *i.e.*

$$\text{CVaR}_\epsilon(X) = \mathbb{E}_\pi[x \mid x \geq \text{VaR}_\epsilon(X)] = \inf_{z \in \mathbb{R}} z + \frac{\mathbb{E}_\pi\left[\max(X - z, 0)\right]}{\epsilon}.$$

Definition 17. The *Entropic-Value-at-Risk* level $\epsilon \in (0, 1]$ denoted as $\text{EVaR}_\epsilon(X)$ is defined as the infimum over $z > 0$ of the Chernoff bound for the scalar random variable X with distribution π, *i.e.*

$$\text{EVaR}_\epsilon(X) = \inf_{z > 0} \frac{1}{z} \ln\left(\frac{\mathbb{E}_\pi\left[e^{zX}\right]}{\epsilon}\right)$$

The relationship between the three risk measures is shown in Figure 4.2. Notably, for a random variable X and some risk level $\epsilon \in (0, 1]$, $\text{VaR}_\epsilon(X) \leq \text{CVaR}_\epsilon(X) \leq \text{EVaR}_\epsilon(X)$, as shown in [140].

4.3 Upper Bounding Risk Measures

For all contributions detailed in this section, we will denote as X a scalar random variable whose distribution π is unknown. Then, provided N independently chosen samples x_i of X, we can construct the following scenario program:

$$\begin{aligned} \zeta_N^* = \underset{\zeta \in \mathbb{R}}{\text{argmin}} \quad & \zeta, \\ \text{subject to} \quad & \zeta \geq x_i, \ \forall\, x_i \in \{x_k\}_{k=1}^N. \end{aligned} \tag{UB-RP-N}$$

Then our first result on upper bounding the Value-at-Risk of X follows.

Theorem 6. *Let ζ_N^* be the solution to* (UB-RP-N) *for a set of N samples $\{x_k\}_{i=k}^N$ of a random variable X with unknown distribution π. The following statement is true $\forall\, \epsilon \in [0, 1]$ and with $\text{VaR}_\epsilon(X)$ as defined in Definition 15:*

$$\mathbb{P}_\pi^N[\zeta_N^* \geq \text{VaR}_\epsilon(X)] \geq 1 - (1 - \epsilon)^N.$$

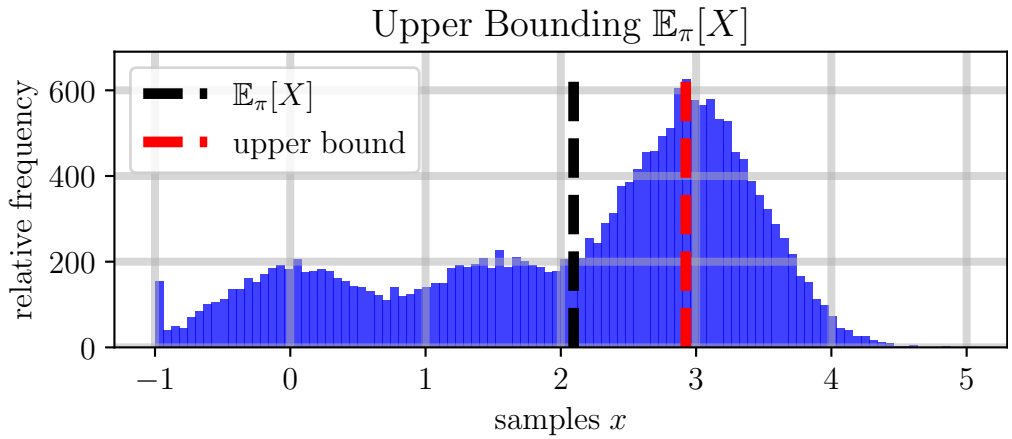

Figure 4.3: To upper bound g-entropic risk measures, we motivate that we first must be able to upper bound the expected value of a scalar random variable X. The figure above is an example of our ability to do this as per Theorem 7. Here, we upper bound (red) the expected value (black) of a multi-modal random variable X by taking $N = 20$ samples of X and knowing its upper bound $u_b = 5$. The true distribution (blue) was calculated numerically by taking 20000 samples of X.

Proof: By Theorem 4, we have that

$$\mathbb{P}_\pi^N \left[\mathbb{P}_\pi[x \leq \zeta_N^*] \geq 1 - \epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

Then the result stems via the definition of Value-at-Risk per Definition 15. ■

To start upper bounding g-entropic risk measures, we will first take a slight detour and provide a method to upper bound the expected value of random variables whose distributions are unknown. We will leverage this procedure in our approach to upper bound g-entropic risk measures which will follow this next subsection.

Upper Bounding Expectations

We will split this procedure into two steps. For the first step, if we have a scalar R.V. X with upper bound $u_b \in \mathbb{R}$, a cutoff $c \in \mathbb{R}$, and the probability mass of samples x of X that lie below this cutoff, *i.e.* $\mathbb{P}_\pi[x \leq c]$, then we can upper bound the expected value of X with this cutoff c and upper bound u_b:

$$\mathbb{E}_\pi[X] = \int_{\mathbb{R}} s\pi(s)\, ds \leq c\, \mathbb{P}_\pi[x \leq c] + u_b(1 - \mathbb{P}_\pi[x \leq c]). \tag{4.2}$$

Theorem 7 to follow, formally states that we can employ Theorem 6 to identify such a cutoff c and upper bound $\mathbb{E}_\pi[X]$ without knowledge of its distribution π provided we know a (perhaps) loose upper bound $u_b \in \mathbb{R}$.

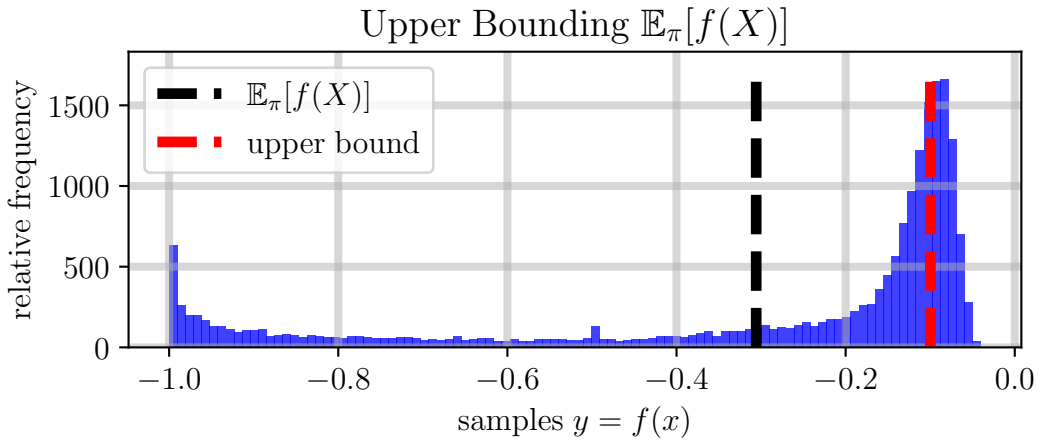

Figure 4.4: The second step to upper bounding g-entropic risk measures is upper bounding the expected value of a function f of a scalar random variable X with unknown distribution π. Above is an example of us upper bounding in red the $\mathbb{E}_\pi[f(X)]$ as shown in black, where $f(x) = \frac{-1}{x^2+1}$ for the multi-modal random variable X whose expectation we upper-bounded in Figure 4.3. To generate this upper bound we took $N = 20$ samples of the random variable $Y = f(X)$ and the distribution was calculated by taking 20000 samples. We formally prove our ability to do this upper bound in Corollary 11.

Theorem 7. *Let X be a scalar R.V. with samples x, unknown distribution function π, and upper bound $u_b \in \mathbb{R}$ such that $\mathbb{P}_\pi[x \leq u_b] = 1$. Furthermore, let ζ_N^* be the solution to* (UB-RP-N) *for a set of N independent and identically drawn samples $\{x_k\}_{k=1}^N$ of X. The following statement is true $\forall\, \epsilon \in [0,1]$:*

$$\mathbb{P}_\pi^N \left[\mathbb{E}_\pi[X] \leq \zeta_N^*(1-\epsilon) + u_b\epsilon\right] \geq 1 - (1-\epsilon)^N. \tag{4.3}$$

Proof: First, by Theorem 6 and the definition of $\text{VaR}_\epsilon(X)$ in Definition 15,

$$\mathbb{P}_\pi^N \left[\mathbb{P}_\pi[x \leq \zeta_N^*] \geq 1-\epsilon\right] \geq 1 - (1-\epsilon)^N.$$

Then by equation (4.2) we have the following inequality:

$$\mathbb{E}_\pi[X] \leq \zeta_N^*\, \mathbb{P}_\pi[x \leq \zeta_N^*] + u_b(1 - \mathbb{P}_\pi[x \leq \zeta_N^*]). \tag{4.4}$$

To finish the proof, we note that the right-hand side of the inequality (4.4) is maximized when this probability $\mathbb{P}_\pi[x \leq \zeta_N^*]$ equals its lower bound $1-\epsilon$, as $u_b \geq \zeta_N^*$. As this lower bound holds with minimum probability $1-(1-\epsilon)^N$, the result holds with the same probability, resulting in (4.3). ■

The second step needed to bound functions of the random variable X is to note that if we have a function $f : \mathbb{R} \to \mathbb{R}$, then $Y \triangleq f(X)$ is another random variable with

samples y and distribution π_Y. Provided Y has an upper bound u_b, then we can upper bound $\mathbb{E}_{\pi_Y}[Y] = \mathbb{E}_{\pi}[f(X)]$. This result does not require X to be bounded, only the image of X under f needs to be bounded as expressed in the following corollary.

Corollary 11. *Let X be a scalar R.V. with samples x and distribution π. Let $f : \mathbb{R} \to \mathbb{R}$ and let $Y = f(X)$ be another scalar R.V. with samples $y = f(x)$, distribution π_Y, and upper bound u_b such that $\mathbb{P}_{\pi_Y}[y \leq u_b] = 1$. Furthermore, let ζ_N^* be the solution to* (UB-RP-N) *for a set of N samples $\{y_k = f(x_k)\}_{k=1}^N$ of Y. The following statement is true $\forall\, \epsilon \in [0, 1]$.*

$$\mathbb{P}_{\pi}^N \left[\mathbb{E}_{\pi}[f(X)] \leq \zeta_N^*(1-\epsilon) + u_b\epsilon\right] \geq 1 - (1-\epsilon)^N.$$

Proof: This result stems via Theorem 7 where $\mathbb{E}_{\pi_Y}[Y] = \mathbb{E}_{\pi}[f(X)]$. ■

Upper Bounding g-Entropic Risk Measures

With Corollary 11, we can upper bound a large class of g-entropic risk measures, including Conditional-Value-at-Risk and Entropic-Value-at-Risk. To do so for general g-entropic risk measures as per Definition 14, we require that the convex conjugate g^* of g maps from the reals to the reals, *i.e.* $g^* : \mathbb{R} \to \mathbb{R}$, and also has an upper bound when applied to samples x of the random variable X.

Assumption 8. X is a scalar random variable with samples x, distribution π, and upper bound $\ell \in \mathbb{R}$ such that $\mathbb{P}_{\pi}[x \leq \ell] = 1$. $\mathrm{ER}_{g,\beta}$ is a g-entropic risk measure as per Definition 14 with respect to some $\beta \geq 0$ and closed convex function g such that $g(1) = 0$. $g^* : \mathbb{R} \to \mathbb{R}$ is the convex-conjugate for g, and g^* has an upper bounding function $u_b : D \subset \mathbb{R}^2 \to \mathbb{R}$ satisfying the following probabilistic inequality:

$$L(x, \mu, t) \triangleq t\left(\mu + g^*\left(\frac{x}{t} - \mu + \beta\right)\right), \tag{4.5}$$

$$\mathbb{P}_{\pi}\left[L(x, \mu, t) \leq u_b(\mu, t)\right] = 1 \;\forall\, \mu \in \mathbb{R},\; t > 0. \tag{4.6}$$

Finally, $\zeta_N^*(\mu, t)$ is the solution to (UB-RP-N) for a set of N-samples $\{y_k = L(x_k, \mu, t)\}_{k=1}^N$ of the random variable $Y = L(X, \mu, t)$ for some $\mu \in \mathbb{R},\; t > 0$.

For context, this assumption is not too restrictive, as it is easily satisfied by both CVaR and EVaR for any risk level $\alpha \in (0, 1]$, and we will show this in a later section. Then via Corollary 11, we can upper bound the objective function in (4.1) and transform (4.1) into an optimization problem over a set of sampled values $\{x_k\}_{k=1}^N$ of X which is easily solvable. The formal statement of this procedure is

provided in the following lemma, theorem, and corollary. The lemma will state that we can upper bound the expected value of the convex conjugate g^* applied to X, and the theorem will utilize this lemma to provide a high confidence upper bound on the g-entropic risk measure of interest. The corollary will formalize a relationship between the confidence in our estimate and our sample complexity.

Lemma 6. *Let Assumption 8 hold. Then,* $\forall\ \epsilon \in [0,1],\ \mu \in \mathbb{R},\ t > 0$,

$$\mathbb{P}_\pi^N \left[\mathbb{E}_\pi[L(X,\mu,t)] \leq \zeta_N^*(\mu,t)(1-\epsilon) + u_b(\mu,t)\epsilon \right] \geq 1 - (1-\epsilon)^N.$$

Proof: This is a direct application of Corollary 11. ■

Theorem 8. *Let Assumption 8 hold. Then,* $\forall\ \epsilon \in [0,1]$,

$$\mathbb{P}_\pi^N \left[ER_{g,\beta}(X) \leq \inf_{t>0,\ \mu\in\mathbb{R}} \zeta_N^*(\mu,t)(1-\epsilon) + u_b(\mu,t)\epsilon \right] \geq 1 - (1-\epsilon)^N.$$

Proof: Via Theorem 5, the g-entropic risk measure $\mathrm{ER}_{g,\beta}$ can be represented via an infimum, and by linearity of the expectation operator and the definition of L in (4.5), we have the following equality:

$$\mathrm{ER}_{g,\beta}(X) = \inf_{t>0,\mu\in\mathbb{R}} \mathbb{E}_\pi[L(X,\mu,t)].$$

Then the result holds via Lemma 6. ■

Corollary 12. *Let Assumption 8 hold,* $\gamma \in [0,1)$*, and* $\epsilon \in (0,1)$*. If for all* $\mu \in \mathbb{R}$ *and* $t > 0$*,* $\zeta_N^*(\mu,t)$ *is the solution to* (UB-RP-N) *for a set of* N *samples* $\{x_k\}_{k=1}^N$ *of the random variable* X *where* $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$*, then*

$$\mathbb{P}_\pi^N \left[ER_{g,\beta}(X) \leq \inf_{t>0,\ \mu\in\mathbb{R}} \zeta_N^*(\mu,t)(1-\epsilon) + u_b(\mu,t)\epsilon \right] \geq \gamma.$$

Proof: Use Theorem 8 where $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)} \implies 1 - (1-\epsilon)^N \geq \gamma$. ■

To summarize, Theorem 8 states that we can upper bound the g-entropic risk measure of a random variable X with unknown distribution π provided that we know an upper bounding function u_b for a function L of the random variable X. Corollary 12 provides the minimum number of samples N required to determine this upper bound with confidence $\gamma \in [0,1)$.

Specializing to CVaR and EVaR

Ahmadi-Javid [140] identifies the convex conjugate function g^* and parameter β with which CVaR can be recast as a g-entropic risk measure.

Remark 1. *The Conditional-Value-at-Risk level $\alpha \in (0, 1]$ can be recast as a g-entropic risk measure with convex conjugate function $g^*(x) = \frac{1}{\alpha}\max\{x, 0\}$ and scalar parameter $\beta = 0$ [140].*

Then, to use Theorem 8 we must show that the CVaR satisfies Assumption 8.

Lemma 7. *The Conditional-Value-at-Risk for any risk-level $\alpha \in (0, 1]$ satisfies Assumption 8.*

Proof: To start, CVaR for any risk-level α is a g-entropic risk measure with $g^*(x) = \frac{1}{\alpha}\max\{x, 0\}$ and $\beta = 0$. As a result, a solution $\zeta_N^*(\mu, t)$ will always exist for (UB-RP-N) as it is the solution to a linear program minimizing a scalar decision variable subject to a finite set of lower bounds taking values in $\mathbb{R}$. Then, to prove that the Conditional-Value-at-Risk at any risk level $\alpha \in (0, 1]$ satisfies Assumption 8, it suffices to identify an upper bounding function $u_b : D \subset \mathbb{R}^2 \to \mathbb{R}$ under the assumption that the scalar random variable X with distribution π and samples x has an upper bound $\ell \in \mathbb{R}$ such that $\mathbb{P}_\pi[x \leq \ell] = 1$. This function u_b will be defined as follows:

$$L(x, \mu, t) = t\left(\mu + \frac{1}{\alpha}\max\left\{\frac{x}{t} - \mu, 0\right\}\right), \quad u_b(\mu, t) = L(\ell, \mu, t). \tag{4.7}$$

Since this upper bounding function satisfies inequality (4.6) in Assumption 8, CVaR $\forall\ \alpha \in (0, 1]$ satisfies Assumption 8. ■

As the Conditional-Value-at-Risk satisfies Assumption 8, we can use u_b (4.7) to provide high-confidence estimates on $\text{CVaR}_\alpha(X)\ \forall\ \alpha \in (0, 1]$.

Corollary 13. *Let X be a scalar random variable with samples x, distribution π, and upper bound $\ell \in \mathbb{R}$ such that $\mathbb{P}_\pi[x \leq \ell] = 1$. Let $\alpha \in (0, 1]$, $\epsilon \in [0, 1]$, and L, u_b be as defined in* (4.7) *with respect to this upper bound ℓ and constant α. Furthermore, let $\zeta^*(\mu, t)$ be the solution to* (UB-RP-N) *for a set of N-samples $\{y_k = L(x_k, \mu, t)\}_{k=1}^N$ of the random variable $Y = L(X, \mu, t)$. Then,*

$$\mathbb{P}_\pi^N\left[\text{CVaR}_\alpha(X) \leq \inf_{\mu\in\mathbb{R},\ t>0} \zeta_N^*(\mu, t)(1 - \epsilon) + u_b(\mu, t)\epsilon\right] \geq 1 - (1 - \epsilon)^N.$$

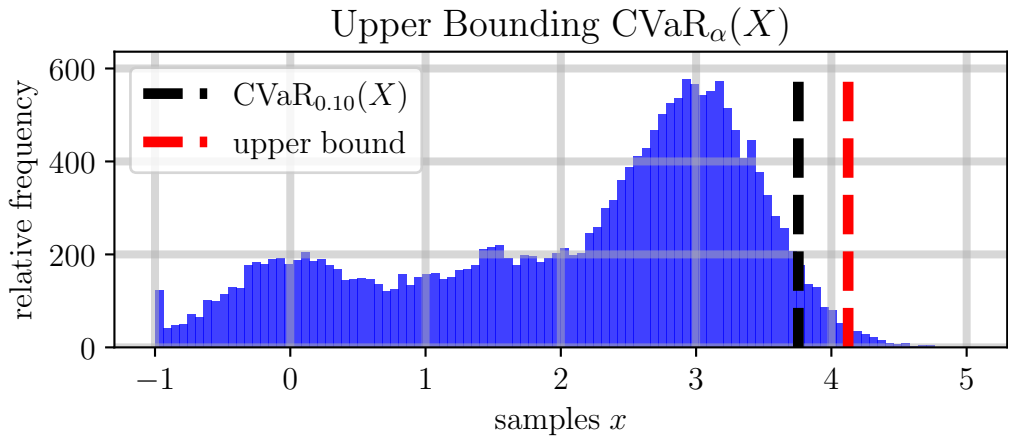

Figure 4.5: A specific g-entropic risk measure, the Conditional-Value-at-Risk for any risk-level $\alpha \in (0, 1]$ should be upper-boundable via our scenario approach as expressed in Corollary 13. Shown above is an example of our scenario approach in bounding $\mathrm{CVaR}_{\alpha=0.1}(X)$ for the R.V. X whose distribution is shown in blue. To generate the upper bound shown in red, we required $N = 45$ samples of X. The true $\mathrm{CVaR}_{0.1}(X)$ is shown in black.

Proof: This is a direct application of Theorem 8 due to Lemma 7. ■

To use Theorem 8 to provide an upper bound on the Entropic-Value-at-Risk for any risk level $\alpha \in (0, 1]$, we will follow a similar procedure as we followed for the Conditional-Value-at-Risk. First, Ahmadi-Javid [140] identifies the convex conjugate function g^* and parameter β which enables Entropic-Value-at-Risk to be cast as a g-entropic risk measure.

Remark 2. *The Entropic-Value-at-Risk level $\alpha \in (0, 1]$ can be recast as a g-entropic risk measure with convex conjugate function $g^*(x) = e^{x-1}$ and scalar parameter $\beta = -\ln(\alpha)$ [140].*

Then, we note that EVaR for any risk-level $\alpha \in (0, 1]$ satisfies Assumption 8.

Lemma 8. *The Entropic-Value-at-Risk for any risk-level $\alpha \in (0, 1]$ satisfies Assumption 8.*

Proof: This proof follows the proof of Lemma 7 insofar as it suffices to identify an upper bounding function $u_b : D \subset \mathbb{R}^2 \to \mathbb{R}$ satisfying the probabilistic inequality in Assumption 8 where $g^*(x) = e^{x-1}$ and $\beta = -\ln(\alpha)$. The following function u_b suffices, with ℓ the upper bound for X:

$$L(x, \mu, t) = t\left(\mu + e^{\frac{x}{t} - \mu - \ln(\alpha) - 1}\right), \quad u_b(\mu, t) = L(\ell, \mu, t). \tag{4.8}$$

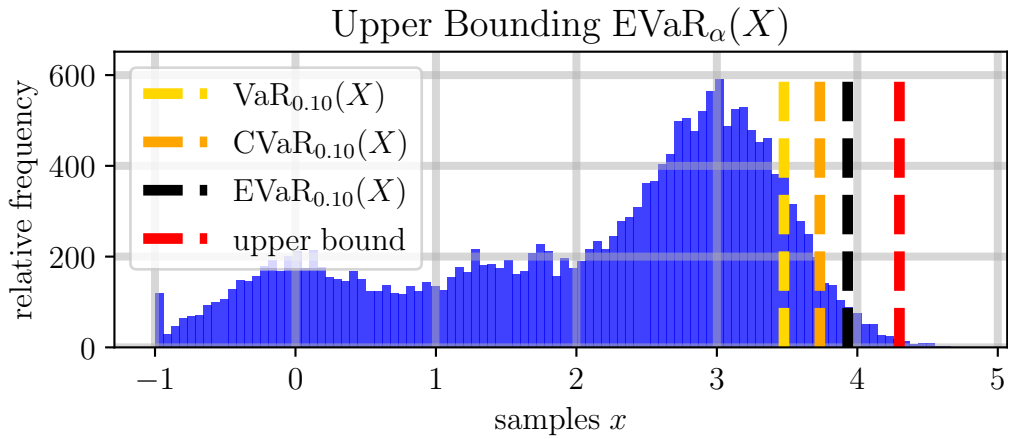

Figure 4.6: A culmination of our approach to upper-bounding g-entropic risk measures. Shown above is our attempt to upper bound the Entropic-Value-at-Risk at risk level $\alpha = 0.1$ of the scalar multi-modal random variable X whose distribution is shown in blue. To calculate this upper bound (red) with $N = 20$ samples of X, we used the same method that we used to determine the upper bound for $\mathrm{CVaR}_\alpha(X)$ in Figure 4.5. We formally state our capacity to do this in Corollary 14. Notice that the true $\mathrm{VaR}_\alpha(X) \leq \mathrm{CVaR}_\alpha(X) \leq \mathrm{EVaR}_\alpha(X)$ as also shown in Figure 4.2. As before, the true $\mathrm{EVaR}_\alpha(X)$ is shown in black.

■

Since $\mathrm{EVaR}_\alpha \ \forall \ \alpha \in (0, 1]$ satisfies Assumption 8, we can use u_b (4.7) to provide high-confidence estimates on $\mathrm{EVaR}_\alpha(X) \ \forall \ \alpha \in (0, 1]$.

Corollary 14. *Let X be a scalar random variable with samples x, distribution π, and upper bound $\ell \in \mathbb{R}$ such that $\mathbb{P}_\pi[x \leq \ell] = 1$. Let $\alpha \in (0, 1]$, $\epsilon \in [0, 1]$, and L, u_b be as defined in* (4.8) *with respect to this upper bound ℓ and constant α. Furthermore, let $\zeta_N^*(\mu, t)$ be the solution to* (UB-RP-N) *for a set of N-samples $\{y_k = L(x_k, \mu, t)\}_{k=1}^N$ of the random variable $Y = L(X, \mu, t)$. Then,*

$$\mathbb{P}_\pi^N \left[\mathrm{EVaR}_\alpha(X) \leq \inf_{\mu \in \mathbb{R},\ t>0} \zeta_N^*(\mu, t)(1 - \epsilon) + u_b(\mu, t)\epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

Proof: This is an application of Theorem 8 via Lemma 8. ■

4.4 Percentile Optimization

Interestingly, the concepts underlying the prior risk-measure estimation results can also be leveraged to produce rapid solutions to a wide class of non-convex optimization problems. To simplify the description of that procedure in this section, we will assume a general optimization problem of the following form, where

$J : \mathbb{S} \subseteq \mathbb{R}^n \to \mathbb{R}$ for some positive integer n and general domain $\mathbb{S}$ is a cost function:

$$\min_{\mathrm{s} \in \mathbb{S}} J(\mathrm{s}). \tag{4.9}$$

For our candidate problem (4.9), we further assume that the decision space $\mathbb{S}$ and cost function J satisfy the following assumption:

Assumption 9. The decision space $\mathbb{S}$ has bounded volume, *i.e.* if $\mathbb{S}$ is a discrete collection of elements, then there are finitely many elements in $\mathbb{S}$, or if it is a continuous subset of $\mathbb{R}^n$, then $\int_{\mathbb{S}} 1 \, d\,\mathrm{s} < \infty$. Likewise, the cost function J is also bounded over $\mathbb{S}$, *i.e.* $\exists\, m, M \in \mathbb{R}$ such that $m \leq J(\mathrm{s}) \leq M \;\forall\; \mathrm{s} \in \mathbb{S}$.

This assumption is not too restrictive, as a large class of optimization problems falls under this setting. This lets us define a volume fraction function $\mathcal{V} : 2^{\mathbb{S}} \to \mathbb{R}$ that outputs the volume fraction of a subset A of $\mathbb{S}$:

$$\mathcal{V}(A) = \frac{\int_A 1 \, d\,\mathrm{s}}{\int_{\mathbb{S}} 1 \, d\,\mathrm{s}}. \tag{4.10}$$

Finally, let $F : \mathbb{S} \to 2^{\mathbb{S}}$ be a set-valued function that identifies the space of decisions $\mathrm{s}' \in \mathbb{S}$ that are "better" than the provided decision s, *i.e.*

$$F(\mathrm{s}) = \{\mathrm{s}' \in \mathbb{S} \mid J(\mathrm{s}') < J(\mathrm{s})\}. \tag{4.11}$$

Then intuitively, by percentile optimization, we hope to identify solutions s^* that outperform some user-defined percent of the decision space $\mathbb{S}$. As an example aligning with the prior definitions, a solution s^* would be in the 90%-ile, if $\mathcal{V}(F(\mathrm{s})) \leq 0.1$. Phrased formally, the problem statement is as follows.

Problem 2. *For any $\epsilon \in (0, 1)$ devise a method to find a decision $\mathrm{s} \in \mathbb{S}$ such that* s *is at least in the $100(1-\epsilon)$-th percentile of all possible decisions $\mathrm{s} \in \mathbb{S}$ with respect to the cost function J, i.e. find a decision* s *such that $\mathcal{V}(F(\mathrm{s})) \leq \epsilon$, with $\mathcal{V}$ defined in equation* (4.10) *and F defined in equation* (4.11)*.*

Sampling Methods for Identification of "Good" Decisions

First, we note that for a uniform distribution over the decision space $\mathbb{S}$, the probability of sampling a decision $\mathrm{s} \in A \subset \mathbb{S}$ is equivalent to the volume fraction of A.

Lemma 9. *Let Assumption 9 hold. The following is true with $\mathcal{V}$ as in* (4.10)*:*

$$\mathbb{P}_{\mathrm{U}[\mathbb{S}]}[\mathrm{s} \in A] = \mathcal{V}(A).$$

Proof: As the distribution is uniform,

$$\mathbb{P}_{\mathrm{U}[\mathbb{S}]}[\mathrm{s} \in A] = \frac{\int_A 1 \, d\,\mathrm{s}}{\int_{\mathbb{S}} 1 \, d\,\mathrm{s}} = \mathcal{V}(A).$$

■

Second, we note that (4.9) can be recast as an uncertain program similar to (UP). Specifically, let X be a random variable whose distribution is the uniform distribution over $\mathbb{S}$, *i.e.* $\mathrm{U}[\mathbb{S}]$. Then, the corresponding random cost is $Y = J(X)$. This random cost variable Y has its own samples y and (unknown) distribution π_Y, letting us construct an uncertain program with respect to samples y of Y:

$$\begin{aligned} \zeta^* = \underset{\zeta \in \mathbb{R}}{\operatorname{argmax}} \quad & \zeta, \\ \text{subject to} \quad & \zeta \leq y, \ y \in \mathbb{R}. \end{aligned} \tag{UP-G}$$

This uncertain program (UP-G) also has an analagous scenario program:

$$\begin{aligned} \zeta_N^* = \underset{\zeta \in \mathbb{R}}{\operatorname{argmax}} \quad & \zeta, \\ \text{subject to} \quad & \zeta \leq y_i, \ y_i \in \{y_k = J(\mathrm{s}_k)\}_{k=1}^N, \ X \sim \mathrm{U}[\mathbb{S}] \text{ with samples } \mathrm{s}. \end{aligned} \tag{RP-G}$$

This scenario program is crucial to our approach for two reasons which we will prove in the theorem to follow. First, there exists a sampled decision s_i in the set of all sampled decisions $\{\mathrm{s}_k\}_{k=1}^N$ such that the cost of this decision is equivalent to the scenario solution ζ_N^*, *i.e.* $J(\mathrm{s}_i) = \zeta_N^*$. Second, via Theorem 4 we can upper bound the probability of sampling another decision $\mathrm{s}' \in \mathbb{S}$ such that $J(\mathrm{s}') < \zeta_N^* = J(\mathrm{s}_i)$. Via Lemma 9, such a probability corresponds to the volume fraction of those decisions $\mathrm{s}' \in \mathbb{S}$ that are "better" than s_i, *i.e.* this probability corresponds to $\mathcal{V}(F(\mathrm{s}_i))$. As a result, to determine a decision $\mathrm{s} \in \mathbb{S}$ that is "better" than a predetermined volume fraction $\epsilon \in (0, 1)$ of all possible decisions $\mathrm{s}' \in \mathbb{S}$ with confidence $\gamma \in [0, 1)$, we just need to take enough samples of X and solve (RP-G).

Theorem 9. *Let $\epsilon \in (0, 1)$, let $\gamma \in [0, 1)$, let Assumption 9 hold, let ζ_N^* be the solution to* (RP-G) *for an N-sample set $\{\mathrm{s}_k\}_{k=1}^N$ of X, let $\mathcal{V}$ be as defined in* (4.10)*, and let F be as defined in* (4.11)*. If $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$ then with minimum probability γ, there exists at least one sampled decision $\mathrm{s}_i \in \{\mathrm{s}_k\}_{k=1}^N$ that is in the $100(1-\epsilon)$-th percentile of all possible decisions $\mathrm{s} \in \mathbb{S}$, i.e.,*

$$\exists \ \mathrm{s}_i \in \{\mathrm{s}_k\}_{k=1}^N \ \text{ s.t. } \mathbb{P}_{\mathrm{U}[\mathbb{S}]}^N[\mathcal{V}(F(\mathrm{s}_i)) \leq \epsilon] \geq \gamma, \text{ and } \zeta_N^* = J(\mathrm{s}_i).$$

Proof: First, for any finite sample set $\{s_k\}_{k=1}^N$ there exists a decision $s_i \in \{s_k\}_{k=1}^N$ such that $\zeta_N^* = J(s_i)$. This is due to the fact that (RP-G) is a linear program maximizing a scalar decision variable subject to a series of upper bounds that take values in $\mathbb{R}$. As a result, ζ_N^* must equal one of its upper bounds to be a valid solution to (RP-G), and as such, it must be equal to $J(s_i)$ for at least one $s_i \in \{s_k\}_{k=1}^N$.

Then via Theorem 4, the following inequality holds for our scenario solution ζ_N^*, where the violation probability $V(\zeta_N^*) = \mathbb{P}_{U[\mathbb{S}]}[J(s) < \zeta_N^*]$:

$$\mathbb{P}^N_{U[\mathbb{S}]}[\mathbb{P}_{U[\mathbb{S}]}[J(s) < \zeta_N^*] \leq \epsilon] \geq 1 - (1-\epsilon)^N. \tag{4.12}$$

By definition of F in (4.11), (4.12) can be recast as follows for some $s_i \in \{s_k\}_{k=1}^N$:

$$\mathbb{P}^N_{U[\mathbb{S}]}[\mathbb{P}_{U[\mathbb{S}]}[s \in F(s_i)] \leq \epsilon] \geq 1 - (1-\epsilon)^N.$$

Via Lemma 9 we can replace the inner probability with $\mathcal{V}(F(s_i))$:

$$\mathbb{P}^N_{U[\mathbb{S}]}[\mathcal{V}(F(s_i)) \leq \epsilon] \geq 1 - (1-\epsilon)^N.$$

Then, as $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$, $\epsilon \in (0,1)$ and $\gamma \in [0,1)$, $1-(1-\epsilon)^N \geq \gamma$. ■

To summarize, Theorem 9 states that if we want to find a "good" solution to (4.9) with minimum probability γ—"good" insofar as it is in the $100(1-\epsilon)$-th percentile of all decisions $s \in \mathbb{S}$ for minimizing J—that we are required to evaluate at minimum $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$ uniformly randomly chosen decisions $s \in \mathbb{S}$. If we evaluate the cost function J for each such sampled decision in our sample set $\{s_k\}_{k=1}^N$, at least one decision $s_i \in \{s_k\}_{k=1}^N$ is guaranteed, with minimum probability γ, to produce a cost $J(s_i)$ that is in the $100(1-\epsilon)$-th percentile of all possible rewards achievable. Hence, this decision s_i that produces this cost $J(s_i)$ is also guaranteed, with minimum probability γ, to be in the $100(1-\epsilon)$-th percentile of all possible decisions $s \in \mathbb{S}$ with respect to the cost function J.

Remark on sample complexity: From Theorem 9, one correct though (perhaps) unintuitive result is that the number of samples required to produce a percentile solution with a required confidence is independent of the "size" of the decision space $\mathbb{S}$. More specifically, let there be two continuous decision spaces $\mathbb{S}_1, \mathbb{S}_2$ with volumes V_1, V_2 respectively such that $V_1 < V_2$, and let J be a cost function defined over both spaces. The prior theoretical result states that if one wished to identify a solution in the 95-th percentile with 95% confidence in both spaces with respect to the same cost function J, we are required to evaluate at least 59 samples. Note,

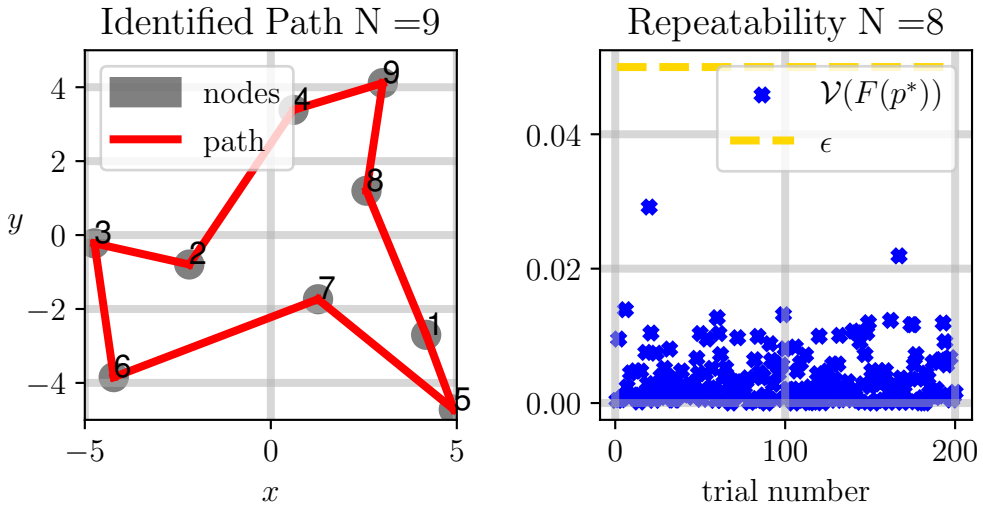

Figure 4.7: Shown above is an application of our percentile optimization procedure to identify "good" paths for a traveling salesman problem [142]. By uniformly sampling $N = 299$ paths from the set of all possible paths P with $|P| = 362880$, we can identify (left) a path that is in the 99-th percentile of all paths. This procedure also repeatably identifies "good" paths, *a.k.a* "good" decisions, as shown in the figure on the right. If we take the minimum number of samples offered by Theorem 9 to identify a path that is in the 95-th percentile with minimum probability $1 - 10^{-6}$, we see that over 200 trials—taking $N = 270$ samples each time—all determined paths are in the 95-th percentile as $\mathcal{V}(F(p^*)) \leq \epsilon = 0.05$.

this sample requirement does not scale with the volume of the decision space. This result arises as percentiles are normalized quantities with respect to the volume of the underlying decision space. In other words, the volume fraction of decisions in the 95-th percentile of a given space will be at least 5%, and this is true regardless of the size of the space. As such, uniformly sampling over the space, would similarly have a 5% chance of identifying such a solution, again, independent of the size of the underlying space. Theorem 9 leverages this fact to provide the minimum sample complexity of identifying such solutions with a given confidence, and from a purely probabilistic lens, this can be viewed as transforming the optimization problem to a Bernoulli random variable sampling problem. To note, this is the same intuitive transformation underlying the risk-measure estimation results in the prior sections.

Examples

This section provides a few examples of applying Theorem 9 to finding paths for the traveling salesman problem (TSP) that are in the 99-th percentile of all possible paths achievable [142]. Specifically, TSP references the identification of a permutation

of integers $P = [i_1, i_3, i_6, \dots]$ where such a permutation minimizes the summation of a function $d : \mathbb{Z}_+ \times \mathbb{Z}_+ \to \mathbb{R}$. That is, if we define a finite set of integers $I \subset \mathbb{Z}_+$, we can also define a set $P \subset 2^I$ of all permutations of integers in I, *i.e.* $P = \{p = (i_1, \dots, i_{|I|}) \mid \forall\, i_j \in I,\ i_j \in p$ and i_j appears only once in $p\}$. Then, the optimization problem is:

$$\min_{p \in P} \sum_{k=1}^{|I|-1} d(p_k, p_{k+1}) + d(p_{|I|}, p_1).$$

Note, this problem satisfies Assumption 9, with $\int_P 1\ ds = |P| < \infty$, implying that we should be able to identify a "good" solution per Theorem 9.

Figure Analysis: Specifically then, we will randomly generate 9 nodes $n_i \in [-5, 5]^2 \subset \mathbb{R}^2$. Our cost function $d(i, j) = \|n_i - n_j\|$. As we have generated 9 nodes, the cardinality of the set of all possible permutations $|P| = 9! = 362880$. However, according to Theorem 9, if we want a permutation $p \in P$ that is in the 99-th percentile with 95% confidence, then we need to uniformly sample $N \geq \frac{\log(1-\gamma=0.05)}{\log(1-\epsilon=0.99)} \approx 299$ permutations $p \in P$ and evaluate their corresponding costs. Doing this procedure once, we identified a permutation $p^* \in P$ that was in the 99.9997-th percentile of all paths as shown on the left in Figure 4.7. This one case shows the utility of our sample approach in providing "good" solutions to difficult optimization problems insofar as we only had to evaluate $299/362880 \approx 0.0008$ of all possible paths to generate a very good one. Indeed, the method also repeatably identifies "good" solutions as shown on the right in Figure 4.7. In this case, we required a path in the 95-th percentile with minimum probability $1 - 10^{-6}$ for a case where we randomly generated 8 nodes $n_i \in [-5, 5]^2 \subset \mathbb{R}^2$. In every case, the identified path p^* was in the 95-th percentile as evidenced by $\mathcal{V}(F(p^*)) \leq \epsilon$.

4.5 Bounding Optimality Gaps

While percentile techniques will rapidly produce "good" solutions to non-convex optimization problems, the sub-optimality of these solutions remains unclear. This section details a method to bound the optimality gaps of percentile solutions. To formally bound these gaps, we must first define them. To that end, consider the same general optimization problem as per (4.9). The optimality gap is defined as follows.

Definition 18. For general optimization problems of the form in (4.9), the *optimality gap* of a decision $\mathrm{s} \in \mathbb{S}$, denoted as $\mathrm{G}(\mathrm{s})$, is the deviance between the decision and optimal values, *i.e.* $\mathrm{G}(\mathrm{s}) = J(\mathrm{s}) - J^*$.

Then as we aim to bound percentile solutions to (4.9), we will assume that we have already taken a percentile approach to solving (4.9).

Assumption 10. Let $I_1 = \{(\mathrm{s}_i, J(\mathrm{s}_i))\}_{i=1}^{N_p}$ be a set of N_p decisions and costs for decisions s_i sampled independently via $\mathrm{U}[\mathbb{S}]$, with $\zeta^*_{N_p}$ the minimum sampled cost and $\mathrm{s}^*_{N_p}$ the (perhaps) non-unique decision with minimum cost.

Then our formal problem statement follows.

Problem 3. *Let Assumption 10 hold, and let the optimality gap* G *be as per Definition 18. Identify a non-arbitrarily large upper bound to* $\mathrm{G}(\mathrm{s}^*_{N_p})$ *and the probability with which this bound holds.*

Results for Bounded Optimization Problems

First, we note that if Assumption 10 holds, we can define the following variance function $\mathbb{V}$ over the set I_1:

$$\mathbb{V} : \mathbb{S} \to \mathbb{R} \text{ s. t. } \mathbb{V}(\mathrm{s}) = \min_{\mathrm{s}_i \in D \subseteq I_1} |J(\mathrm{s}) - J(\mathrm{s}_i)|. \tag{4.13}$$

The restriction of $\mathrm{s} \in D \subseteq I_1$ in the definition of $\mathbb{V}$ above is purely for practical purposes. From a theoretical standpoint, one could use the entire information set I_1 to define $\mathbb{V}$, but this tends to increase sample requirements as will be discussed in sections to follow (*e.g.* Figure 4.9)

Intuitively then, we aim to maximize $\mathbb{V}$ via a percentile method to identify a variance that supersedes the optimality gap $\mathrm{G}(\mathrm{s}^*_{N_p})$ of our chosen decision. To do so, we first require the following fairness assumption — that it is possible to sample variances at least as large as the optimality gap, as otherwise, it would be impossible to take a percentile approach. Formally, let Ω_r be the r-level set of $\mathbb{V}$:

$$\Omega_r = \{\mathrm{s} \in \mathbb{S} \mid \mathbb{V}(s) \leq r\}. \tag{4.14}$$

Assumption 11. Let the variance function $\mathbb{V}$ be as per (4.13), let the optimality gap G be as per Definition 18, let $\mathcal{V}$ be as per (4.10), let Ω_r be as per (4.14), and let Assumption 10 hold. The level set $\Omega_{\mathrm{G}(s^*_{N_p})}$ of decisions whose variance is at most the optimality gap of $\mathrm{s}^*_{N_p}$ does not encompass $\mathbb{S}$, *i.e.*

$$\mathcal{V}\left(\mathbb{S} \setminus \Omega_{\mathrm{G}(\mathrm{s}^*_{N_p})}\right) > 0. \tag{4.15}$$

Second, we have the following result regarding level sets Ω.

Lemma 10. *Let Ω_r be as defined in* (4.14) *and let $\mathcal{V}$ be as per* (4.10)*. The following statements are all equivalent.*

$$r \geq s \overset{(1)}{\iff} \Omega_r \supseteq \Omega_s \overset{(2)}{\iff} \mathcal{V}(\Omega_r) \geq \mathcal{V}(\Omega_s) \overset{(3)}{\iff} \mathbb{P}_{\mathrm{U}[\mathbb{S}]}[\Omega_r] \geq \mathbb{P}_{\mathrm{U}[\mathbb{S}]}[\Omega_s]$$

Proof: The first equivalency stems via the definition of Ω_r in (4.14). The second equivalency stems via the definition of the volume fraction function in (4.10). Finally, the third equivalency stems via the uniform distribution assigning probabilistic weight to subsets of $A \subseteq \mathbb{S}$ equivalent to $\mathcal{V}(A)$. ■

Third, based on our fairness assumption, we know there exists a non-zero probability of sampling decisions such that their variances are at least the optimality gap.

Lemma 11. *Let Assumption 11 hold, then there exists $p > 0$ corresponding to the probability of uniformly sampling a decision $\mathrm{s} \in \mathbb{S} \setminus \Omega_{\mathrm{G}(\mathrm{s}^*_{N_p})}$, i.e.*

$$\mathbb{P}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{S} \setminus \Omega_{\mathrm{G}(\mathrm{s}^*_{N_p})}\right] = p > 0. \tag{4.16}$$

Proof: The result holds by definition of the uniform distribution over $\mathbb{S}$ and equation (4.15). ■

Then the main result in the utilization of a percentile approach to upper bound the optimality gap stems from the prior lemmas and assumption. Formally, we aim to take a percentile approach to the following optimization problem:

$$\max_{\mathrm{s} \in \mathbb{S}} \ \mathbb{V}(\mathrm{s}), \tag{4.17}$$

which results in the following theorem.

Theorem 10. *Let Assumptions 9 and 11 hold, let p satisfy* (4.16)*, let $I_2 = \{(\mathrm{s}_i, \mathbb{V}(\mathrm{s}_i))\}_{i=1}^{N_v}$ be a set of N_v decisions s_i independently sampled from $\mathrm{U}[\mathbb{S}]$ with their corresponding variances as per* (4.13) *and with $\mathbb{V}^*_{N_v}$ the maximum sampled variance. Then, $\forall \epsilon \in [0, p]$, $\mathbb{V}^*_{N_v}$ exceeds the optimality gap of the percentile solution with confidence $1 - (1 - \epsilon)^{N_v}$, i.e.*

$$\mathbb{P}^{N_v}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{V}^*_{N_v} \geq \mathrm{G}(\mathrm{s}^*_{N_p})\right] \geq 1 - (1 - \epsilon)^{N_v}. \tag{4.18}$$

Proof: First, we know there exists a $p > 0$ satisfying (4.16) via Lemma 11. Second, we know that for the optimization problem (4.17), the decision space $\mathbb{S}$ and objective function $\mathbb{V}$ are bounded — this stems via Assumption 9. This permits us to take

a percentile approach to solving (4.17). Via Theorem 9 and looking at (4.17) as a minimization, we have the following result for all $\epsilon \in [0, 1]$:

$$\mathbb{P}^{N_v}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{P}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{V}(\mathrm{s}) \leq \mathbb{V}^*_{N_v}\right] \geq 1-\epsilon\right] \geq 1-(1-\epsilon)^{N_v}.$$

The set in the interior probability corresponds to the level set of $\mathbb{V}^*_{N_v}$, *i.e.*

$$\mathbb{P}^{N_v}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{P}_{\mathrm{U}[\mathbb{S}]}\left[\Omega_{\mathbb{V}^*_{N_v}}\right] \geq 1-\epsilon\right] \geq 1-(1-\epsilon)^{N_v}.$$

Finally, via (4.16), we know that the probability of sampling a decision in the level set corresponding to the optimality gap is $1-p$. Restricting to $\epsilon \in [0, p]$ which implies that $1-\epsilon \geq 1-p$ and substituting terms in the inequality above, we have the following result.

$$\mathbb{P}^{N_v}_{\mathrm{U}[\mathbb{S}]}\left[\mathbb{P}_{\mathrm{U}[\mathbb{S}]}\left[\Omega_{\mathbb{V}^*_{N_v}}\right] \geq \mathbb{P}_{\mathrm{U}[\mathbb{S}]}\left[\Omega_{\mathrm{G}(\mathrm{s}^*_{N_p})}\right]\right] \geq 1-(1-\epsilon)^{N_v}.$$

Then, the final result holds due to Lemma 10. ■

In summary, Theorem 10 tells us that if we wish to bound the optimality gap of a percentile solution, we need to evaluate the variance of N_v uniform samples s from $\mathbb{S}$ with respect to a subset of the information set I_1 utilized to generate the percentile solution. In practice, however, the exact probability p of sampling decisions with large enough variance will be unknown to the practitioner *apriori*. In these cases, it suffices to assume a small enough value for ϵ, *i.e.* 10^{-2} or smaller, is smaller than p. Examples along this vein will be provided in the following section. Notably, this result implies that we can utilize percentile methods to both identify decisions that outperform a large fraction of the decision space and also determine their optimality gap. Indeed, this result holds even for non-convex optimization problems, provided they satisfy Assumption 9.

Producing Solutions with Maximum Optimality Gaps

The prior section provides a method to determine the upper bound on the optimality gap of a provided solution via a secondary sampling scheme. This section provides a method to remove the secondary sampling requirement for similar optimization problems to be solved successively. In other words, consider the following general form of optimization problems, where each instance l satisfies Assumption 9:

$$J^{l*} = \min_{\mathrm{s}\in\mathbb{S}^l} J^l(\mathrm{s}),\ (\mathbb{S}^l, J^l) \in \mathbb{O},\ l \in L. \tag{4.19}$$

Furthermore, we assume that it is possible to randomly sample indices l from L, *e.g.* via the uniform distribution.

Example Setting: Here, $\mathbb{O}$ is a set containing pairs of objective functions and decision spaces. To provide an example consistent with the sections to follow, consider the following nonlinear dynamical system with state x and input u:

$$x_{k+1} = f(x_k, u_k),\ x \in \mathcal{X},\ u \in \mathcal{U} \tag{4.20}$$

Provided a cost function over states and inputs, state constraints, and torque bounds, we can construct the following finite-time optimal controller with horizon H for the aforementioned system. Here, all state constraints are projected to input constraints through prediction over the model (4.20):

$$\begin{aligned} \underset{\mathbf{u}=(u^0,u^1,\ldots,u^{H-1})\in\mathcal{U}^H}{\operatorname{argmin}} \quad & J(\mathbf{u}, x_k), \\ \text{subject to} \quad & \mathbf{u} \in \mathbb{U}(x_k) \subseteq \mathcal{U}^H. \end{aligned} \tag{FTOCP}$$

The aforementioned finite-time optimal controller (FTOCP) collapses to the form in (4.19) if we consider an optimality set $\mathbb{O}$ indexed by states $x \in \mathcal{X}$ — a specific form of indexing more generally referred to via $l \in L$ in (4.19).

Key Insight: The critical insight for this section then is as follows. If we were to randomly sample via a distribution π over $\mathbb{O}$, optimization problems of the form in (4.19) and calculate the optimality gap G^l of percentile solutions for that problem, the corresponding gap is a sample of some real-valued random variable. By taking multiple independent samples of this random variable, we can leverage Theorem 6 to provide a probabilistic upper bound on this random variable, *i.e.* a probabilistic upper bound on achievable optimality gaps. To do so, we require a definition.

Definition 19. Let $\mathcal{G}(N_p)$ be a real-valued random variable with distribution π_{G} and samples $\mathbf{g}$ defined as follows: (1) Uniformly sample an index $l \in L$, (2) Take a percentile approach to solve (4.19) corresponding to this sampled index l, producing the solution $\mathrm{s}^{l*}_{N_p}$, (3) Calculate and report as a sample $\mathbf{g}$, the optimality gap $\mathrm{G}^l\left(\mathrm{s}^{l*}_{N_p}\right)$.

Then, we can upper bound the optimality gaps of percentile solutions to all optimization problems formed in the set $\mathbb{O}$, to some minimum probability. The formal statement will follow.

Theorem 11. *Let $\mathcal{G}(N_p)$ be as per Definition 19, and let $\{\mathbf{g}_i\}_{i=1}^R$ be a set of R independent samples of $\mathcal{G}(N_p)$ with $\mathbf{g}^*_R$ the maximum such sample. Then $\forall\, \epsilon \in [0, 1]$, percentile solutions to optimization problems* (4.19) *will exhibit optimality gaps less than $\mathbf{g}^*_R$ with minimum probability $1 - \epsilon$ and confidence $1 - (1 - \epsilon)^R$, i.e.*

$$\mathbb{P}^R_{\pi_{\mathrm{G}}}\left[\mathbb{P}_{\pi_{\mathrm{G}}}[\mathbf{g} \leq \mathbf{g}^*_R] \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^R.$$

Proof: Stems via Theorem 6. ■

In short, if we wish to remove the secondary sampling requirement for the determination of optimality gaps, we need to be able to calculate the optimality gap for at least R independently chosen optimization problems of the form (4.19). Doing so permits us to make a general statement on the maximum achievable optimality gaps, to some minimum probability. Now, we will provide a few examples.

Validating Theorem 10 — Traveling Salesman

To validate Theorem 10 we can revisit the same traveling salesman problem referenced in the prior section. To repeat for the sake of completeness, the Traveling Salesman Problem (TSP) is a classic non-convex path planning problem referencing the identification of the path of the shortest distance traversing each node in a set once. Mathematically, consider a set of waypoints $W = \{w_1, w_2, \ldots, w_{|W|}\}$ $w_i \in \mathbb{R}^2$ and the set of all paths over these waypoints $P = \{(i_1, i_2, \ldots, i_{|W|}) \mid i_j \in \{1, 2, \ldots, |W|\},\ i_j \neq i_k,\ \forall\ j \neq k\}$. Then the Traveling Salesman Problem is to

$$\min_{p \in P} \sum_{i=0}^{|W|-1} \|p_i - p_{i+1}\| + \|p_0 - p_{|W|}\|.$$

For a graph with 10 nodes and 3628800 possible paths, evaluating $N_p = 5000$ paths and picking the best one identifies a path $p^*_{N_p}$ in the 99.9%-ile with at least 99% confidence according to Theorem 9. Using a subset D of the corresponding information set I_1 for the determination of such a percentile solution (see Assumption 10 for the definition of I_1), we define a variance function $\mathbb{V}$ as per equation (4.13). Finally, to validate the probabilistic results of Theorem 10, we can also calculate the true probability p of uniformly sampling paths that exhibit a variance higher than the optimality gap of our proposed solution $\mathrm{G}(p^*_{N_p})$ — this is the minimum probability assumed to exist via Assumption 11 and defined in equation (4.16). For this particular node set and percentile solution $p^*_{N_p}$ the probability $p = 0.1083$.

Figure Analysis: To validate the results of Theorem 10, we solved for the minimum number of samples N_v required to determine an upper bound $\mathbb{V}^*_{N_v}$ to $\mathrm{G}(p^*_{N_p})$ with minimum confidence 0.7 — top figure in Figure 4.8 requiring 11 samples— and minimum confidence 0.999 — middle figure in Figure 4.8 requiring 61 samples. Both of these minimum sample requirements were identified by setting $\epsilon = p$ in (4.18) and solving for the minimum integer N_v required to make the right-hand side greater than or equal to our desired confidence. For each confidence level, we performed 100 separate trials and reported the maximum variance $\mathbb{V}^*_{N_v}$ produced

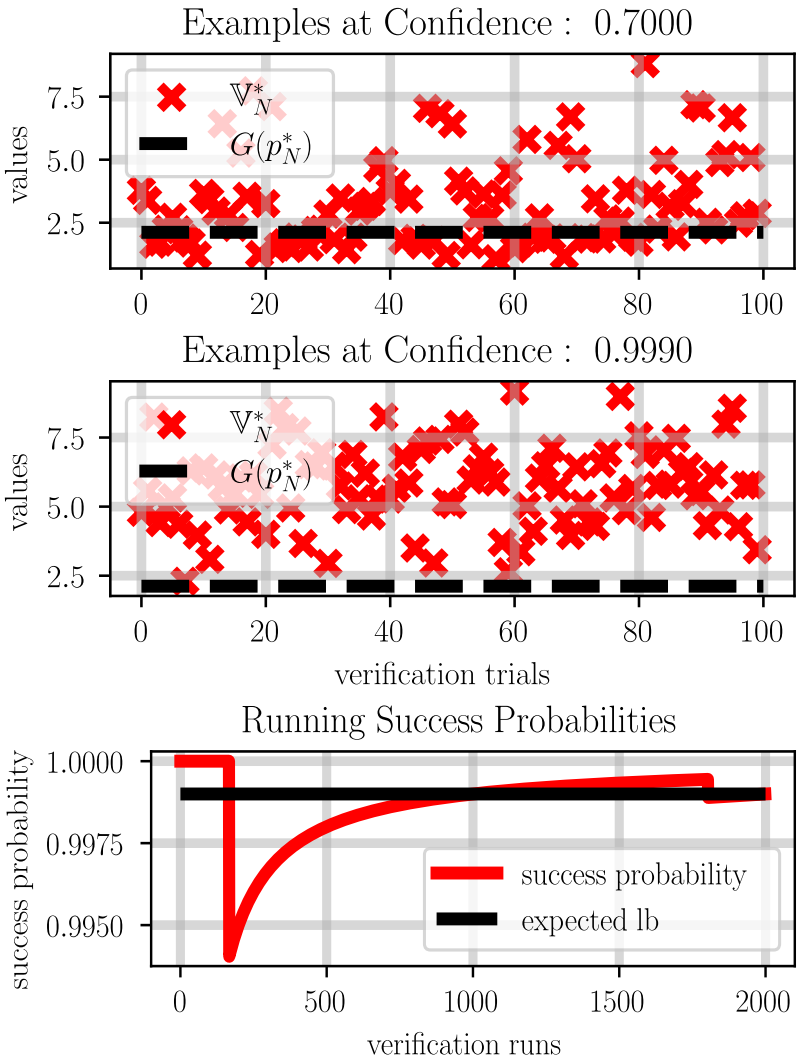

Figure 4.8: Validation Data for Section 4.5 corresponding to Theorem 10. (Top) 100 reported upper bounds $\mathbb{V}_{N_v}^*$ using Theorem 10 with desired confidence equal to 0.7. (Middle) 100 reported upper bounds $\mathbb{V}_{N_v}^*$ with confidence 0.999. (Bottom) Running fraction over 2000 trials of reported upper bounds $\mathbb{V}_{N_v}^*$ exceeding the true optimality gap $\mathrm{G}(p_{N_p}^*)$ at confidence level 0.999. Notice how the fraction of upper bounds exceeding the optimality gap increases as we increase confidence (top to middle), and the running fraction of upper bounds exceeding the optimality gap converges to our desired confidence (bottom), corroborating Theorem 10.

by each trial according to Theorem 10. As can be seen in the corresponding data, increasing the confidence increases the likelihood that the corresponding reported result $\mathbb{V}_{N_v}^*$ exceeds the true optimality gap $\mathrm{G}(p_{N_p}^*)$ — the red x'es in the middle figure are all above the black, dashed line, whereas a few dip below the same line in the top figure when we report solutions with lower confidence. Furthermore,

by repeating the procedure once more at confidence 0.999, taking 2000 separate verification runs, and recording whether $\mathbb{V}^*_{N_v} \geq \mathrm{G}(p^*_{N_p})$ per run, we can get a sense of the true, running probability that $\mathbb{V}^*_{N_v} \geq \mathrm{G}(p^*_{N_p})$. As can be seen in the bottom figure, this probability converges to 0.999 — the lower bound expected by Theorem 10. Notably, though, this result implies that we were able to identify a path in the 99%-ile that was no more than 1.12 times the length of the optimal path, by only evaluating 5061 paths, less than 0.14% of the overall decision space.

Increasing Success Probabilities — Benchmark Functions

In a brief remark after defining the variance function in equation (4.13), we mentioned that by specific choice of a subset D, one could increase the baseline probability p of uniformly choosing samples that exhibit a higher variance than the optimality gap of the reported percentile solution. This section provides evidence in support of that statement for a few benchmark optimization problems. The one referenced in Figure 4.9, the 2-d Rastigrin function [143], is as follows:

$$\min_{x \in [-5.12, 5.12]^2} \; 20 + \sum_{i=1}^{2} (x_i^2 - 10\cos(2\pi x_i)). \tag{4.21}$$

Both the decision space and objective function are bounded, permitting a percentile solution to (4.21). Following Theorem 9 and taking $N_p = 100$ samples for such an approach, we generate the information set I_1 and percentile solution $x^*_{N_p}$. Then, by choice of a subset D of I_1 for the definition of the variance function in (4.13), we claim we can vary the baseline probability p of sampling decisions that exhibit a higher variance than the true optimality gap. To show this, define χ to be the volume fraction the subset D in (4.13) occupies of I_1, *i.e.* $\chi = 1$ implies $D = I_1$ and $\chi = 0.05$ implies that D contains $1/20$-th as many elements as I_1.

Figure Analysis: Figure 4.9 portrays the results of varying the volume fraction χ of decisions D utilized to generate the variance function $\mathbb{V}$ per equation (4.13). The decisions highlighted in orange are those that exhibit a higher variance than the optimality gap of the reported solution. Notice that as χ decreases, p increases as indicated in the titles. This inverse relationship arises as if we consider two sets $D_1 \subset D_2$ utilized for generation of the variance functions $\mathbb{V}_1, \mathbb{V}_2$, respectively, then $\forall\ \mathrm{s} \in \mathbb{S}$, $\mathbb{V}_1(s) \geq \mathbb{V}_2(s)$ by definition of $\mathbb{V}$ as a minimization problem. In other words, decreasing the amount of information provided to the variance function increases the corresponding conservativeness of the resulting function, which increases the likelihood of sampling a, now more conservative, upper bound

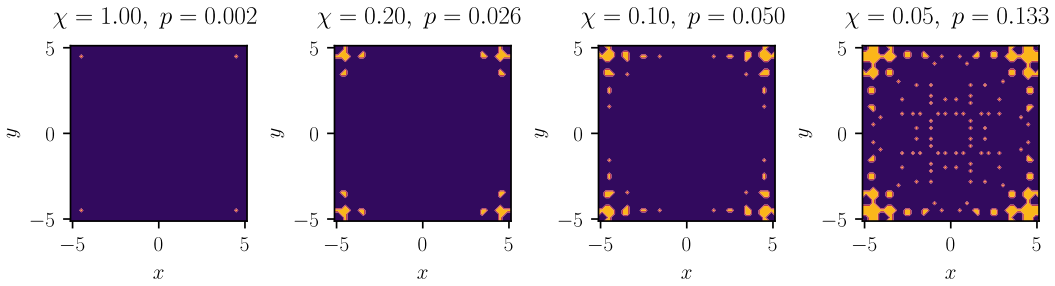

Figure 4.9: Validation data for Section 4.5. We claim that by varying the amount of information used to generate the variance function $\mathbb{V}$, we can change the baseline probability p of sampling a decision whose variance exceeds the optimality gap of a given percentile solution (such decisions are highlighted in orange). Notice that as the volume fractions χ occupied by the chosen information set D decreases, we see a corresponding increase in the baseline probability p. Section 4.5 discusses why this inverse relationship holds.

Name	N_p	N_v	expected success probability (4.18)	true success probability	average runtime (ms)
R-2	300	300	0.95	≈ 1	5.18
R-10	300	300	0.95	≈ 1	5.17
Ack	300	300	0.95	≈ 1	5.28
Ble	300	300	0.95	≈ 1	5.17
Levi	300	300	0.95	≈ 1	5.31
Himm	300	300	0.95	≈ 1	5.30

Table 4.1: Data for Section 4.5 for the (R-2) Rastigrin 2-D, (R-10) Rastigrin 10-D, (Ack) Ackley, (Ble) Beale, Levi, and (Himm) Himmelblau benchmark problems.

on the optimality gap. In practice, and in the examples provided in the prior section, using a dilation $\chi = 0.1$ proved most effective, though studying if there exists an optimal volume fraction remains an open problem and the subject of future work.

Table Description: By increasing the success probability p of sampling decisions whose variance exceeds the optimality gap, we can "blindly" use Theorem 10 to bound the optimality gap of percentile solutions to benchmark optimization problems. Table 4.1 shows our data in this vein. For each benchmark optimization problem, we produced a percentile solution using $N_p = 300$ samples, constructed a variance function $\mathbb{V}$ using 10% of the information set I_1 generated via the percentile method, and took $N_v = 300$ samples to identify the probabilistic maximum variance $\mathbb{V}^*_{N_v}$. Under the assumption that $p \geq 0.01$, Theorem 9 tells us that $\mathbb{V}^*_{N_v} \geq \mathrm{G}(x^*_{N_p})$

with 95% probability — column 4 in Table 4.1. Indeed, over 5000 trials following the above procedure for each optimization problem, we were successfully able to identify a valid upper bound every time. Additionally, as the sampling method only requires the evaluation of sampled points, which is relatively quick, the procedure takes very little time to implement, as seen in the rightmost column.

Validating Theorem 11 — Nonlinear MPC

To validate the results of Theorem 11, we require a series of optimization problems of the form in (4.19). Keeping with the example provided in Section 4.5, we aim to bound the optimality gap of percentile solutions for a Nonlinear MPC controller steering the Robotarium robots [144] — a collection of agents modelable via unicycle dynamics. Formally, let $x \in \mathcal{X} = [-1.6, 1.6] \times [-1.2, 1.2] \times [0, 2\pi]$ be the system state, and let $u \in \mathcal{U} = [-0.2, 0.2] \times [-\pi, \pi]$ be the control input space. Then,

$$x_{k+1} = x_k + \underbrace{\begin{bmatrix} \cos(x_k[3]) & 0 \\ \sin(x_k[3]) & 0 \\ 0 & 1 \end{bmatrix} u_k (\Delta t = 0.033)}_{f(x_k, u_k)}.$$

Furthermore, each agent has a Lyapunov controller U steering it to a provided waypoint $w \in \mathcal{W}$:

$$U : \mathcal{X} \times \mathcal{W} \triangleq [-1.6, 1.6] \times [-1.2, 1.2] \to \mathcal{U}.$$

We will use U to construct our MPC algorithm, which provides waypoints to steer the system around static and moving obstacles toward at least one goal. Formally, we overlay an 8×5 grid on $\mathcal{W}$ and define the space of operating environments $\mathcal{D}$ as those environments that: (1) have 8 static obstacles (SO) and 3 goals (g), (2) have controlled (x_a) and uncontrolled (x_o) agent starting positions outside static obstacles and goals, and (3) have at least one feasible path from the controlled agent's starting location to one of the three goals. A vector $d = [x_a, x_o, \text{SO}, g] \in \mathcal{D}$ corresponds to one such environmental setup. To account for collisions, consider the following barrier function h, where $P = [\mathbf{I}_{2\times 2}\ \mathbf{0}_{2\times 1}]$ projects system states to the plane [145]:

$$h(x_a, x_o, d) = \begin{cases} -5 & \text{in SO cell,} \\ \|P(x_a - x_o)\| - 0.18 & \text{else.} \end{cases}$$

Then the nominal NMPC algorithm minimizes $S : \mathcal{W} \to \mathbb{R}$ — a function outputting the shortest path distance from a waypoint to the closest goal — while ensuring that

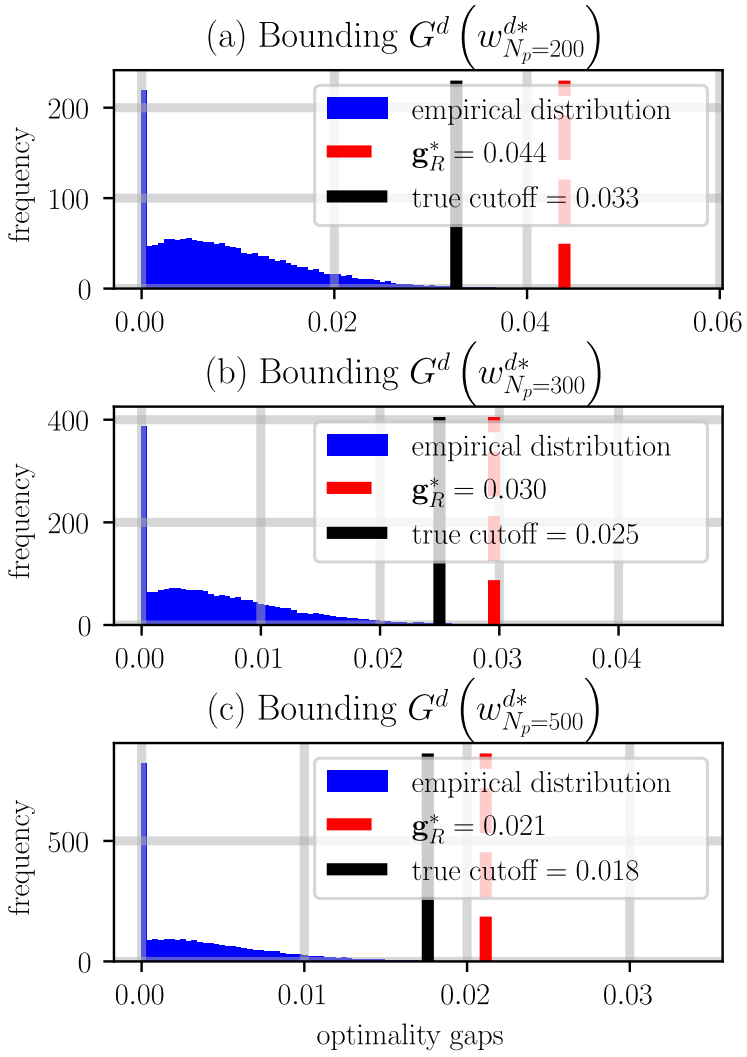

Figure 4.10: Validation data for Section 4.5 in support of Theorem 11. We claim that we can upper bound the optimality gap of successive applications of percentile methods to solve appropriate optimization problems. Shown above in red are the calculated upper bounds for the black lines corresponding to the 99% cutoff value of optimality gaps for percentile solutions to a nonlinear model predictive controller. For the three separate percentile methods shown, we're able to upper bound the true value every time, corroborating Theorem 11.

the existing Lyapunov controller U renders the barrier h positive for the next 5 time

steps, *i.e.*, $\forall\ j = 1, 2, \ldots, 5$,

$$
\begin{aligned}
w_k^* = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \quad & S(w), && \text{(NMPC-A)}\\
\text{subject to} \quad & x_k^j = f(x_k^{j-1}, u^{j-1}), && \text{(a)}\\
& x_k^0 = x_k, && \text{(b)}\\
& h(x_{k,a}^j, x_o, d) \geq 0 && \text{(c)}\\
& u^{j-1} = U\left(x_k^{j-1}, w\right), && \text{(d)}\\
& 0.05 \leq \|w - x_k\| \leq 0.2.
\end{aligned}
$$

To ease sampling requirements, we consider an augmented cost J that outputs 100 whenever a waypoint w fails to satisfy constraints (a)-(d) in (NMPC-A), yielding the following:

$$
\begin{aligned}
w^{d*} = \underset{w \in \mathcal{W}}{\operatorname{argmin}} \quad & J(w, d), && \text{(NMPC-B)}\\
\text{subject to} \quad & 0.05 \leq \|w - x\| \leq 0.2.
\end{aligned}
$$

Finally, we note that (NMPC-B) is equivalent to (4.19), if we consider as index set L, the environment set $\mathcal{D}$.

Figure Analysis: To validate the results of Theorem 11 we uniformly sample $R = 459$ different environments $d \in \mathcal{D}$ and calculate a percentile solution to (NMPC-B) with $N_p = 200$ samples, Figure 4.10-(a); $N_p = 300$ samples, Figure 4.10-(b); and $N_p = 500$ samples, Figure 4.10-(c). We calculate the optimality gap for each solution by performing gradient descent on the best out of 2000 uniformly chosen samples and reporting the final value as the true optimal value. According to Theorem 11, in each case we should produce an upper bound on optimality gaps $\mathbf{g}_R^*$ that exceeds sample-able optimality gaps $\mathbf{g}$ with at least 99% probability and 99% confidence. To validate this statement, we uniformly sampled 50000 more environments $d \in \mathcal{D}$ and followed the prior scheme for each percentile case to determine the distribution of sample-able optimality gaps. This data is reflected as the distributional data you see in each subfigure in Figure 4.10. As can be seen, in each case the reported upper bound $\mathbf{g}_R^*$ exceeds the true 99% cutoff. Furthermore, as the number of samples taken for the percentile solution increases, the upper bound decreases. This is expected as we are providing a solution in a higher percentile each time. To emphasize the utility of this result for controls, say we wished to implement a percentile procedure with $N_p = 300$ samples to provide "good" waypoints optimizing for (NMPC-B).

Offline calculation of this optimality gap would provide confidence that in practice, we would, with 99% probability and with 99% confidence, be choosing waypoints within $0.03m$ of the optimal waypoint at every iteration. Notably, we would not have to solve the non-convex program and repetitively sample variances at each time step to make this statement.

4.6 Conclusion

This chapter detailed our efforts in uncertainty quantification. Specifically, it detailed a procedure for upper bounding risk-measure evaluation of scalar random variables whose distributions are unknown for Value-at-Risk and g-entropic risk measures. Then, extending these results, it detailed a percentile optimization technique for non-convex optimization problems that rapidly produces solutions outperforming a large fraction of the decision space with respect to optimizing for a certain objective. Finally, it mentioned how utilizing the same percentile optimization technique on a variance function defined over the information set generated by taking a percentile approach permits upper bounding of the optimality gap of the reported percentile solution. Following chapters will leverage these mathematical results to formalize pipelines for risk-aware verification and controller synthesis and make probabilistic guarantees for optimal control.

Chapter 5

RISK-AWARE SYNTHESIS AND VERIFICATION

This chapter was adapted from:

[1] P. Akella, A. Dixit, M. Ahmadi, J. W. Burdick, and A. D. Ames, "Sample-Based Bounds for Coherent Risk Measures: Applications to Policy Synthesis and Verification," *The Artificial Intelligence Journal (Under Review)*, Apr. 2022. DOI: 10.48550/arXiv.2204.09833. arXiv: 2204.09833 [cs.AI],

[2] P. Akella, M. Ahmadi, and A. D. Ames, "A Scenario Approach to Risk-Aware Safety-Critical System Verification," *arXiv e-prints*, arXiv:2203.02595, Mar. 2022. DOI: 10.48550/arXiv.2203.02595. arXiv: 2203.02595 [eess.SY],

With the mathematical setup offered by the prior chapter, this chapter will focus on pipelines for both risk-aware safety-critical controller verification and synthesis. Specifically, we detail how we can phrase risk-aware controller verification as a risk-measure estimation problem for a random variable whose distribution is unknown. As such, we can utilize the bounding methods detailed in the prior chapter to facilitate data-efficient risk-aware verification that is provably dimensionally independent. Similarly, we can optimize to maximize the lower bound provided by such a bounding procedure to phrase risk-aware controller synthesis as an optimization problem solvable by percentile methods. As such, reported controllers have intuitive meaning insofar as a controller in the 90%-ile, is better than 90% of controllers we could have possibly developed for a given system, subject to the parameter set provided for controller synthesis. As both problems of verification and policy synthesis under uncertainty have been well-studied, this chapter will start with a brief review of existing work and mention how our contributions fit into the broader picture.

5.1 Introduction

As mentioned in the introduction for the prior chapter, the problem of optimal policy generation under uncertainty has been well-studied in both the learning and controls communities, most notably via Reinforcement Learning [108]–[112]. However, as motivated earlier, most of these works synthesize policies to maximize an expected reward. For safety-critical control applications, failing to consider variances in outcomes could lead to catastrophic behavior [42]. Indeed this is why

the community at large is advocating for a risk-aware approach utilizing the same risk measures popularized by the financial community [43], [45], and this advocacy has led to the widespread study of risk-aware policy synthesis in both communities.

As the prior chapter laid the groundwork for the calculation of risk measures for random variables whose distributions are unknown, it has removed the primary hindrance in our ability to take a risk-aware approach to safety-critical controller verification. As such we can now ask whether we can use these bounds to provide high-confidence statements on system performance in a risk-aware setting. Furthermore, the previously cited risk-aware controller generation works typically require *apriori* understanding of the underlying uncertainty — whether via direct knowledge or in a distributionally-robust sense. Granted, risk-aware Reinforcement Learning does not require such knowledge, and existing convergence bounds guarantee that repeated iteration will eventually identify a satisfactory policy [121]–[124]. However, if we can determine a basic sample requirement for our risk measure bounds, and if the result of policy synthesis is the identification of satisfactory policies, can we provide similar sample requirements for the synthesis of satisfactory risk-aware policies and the relative complexity in identifying better policies?

Summary of Chapter Contributions

Our contributions will be itemized as follows.

1. We rephrase risk-aware verification as a risk measure determination problem for a random variable whose distribution is unknown. This permits us to use our prior results to generate high-confidence verification statements for arbitrarily complex robotic systems with limited system information.

2. We rephrase risk-aware synthesis as an optimization problem solvable by percentile-based methods, whose objective is to maximize the lower bound offered by the prior risk-aware verification procedure.

3. To showcase the efficacy of our work, we verify and synthesize a controller for a cooperative three-agent robotic system that avoids self-collisions while each agent traverses to its goal. We also show that our synthesized controller outperforms the baseline controller with which the system is equipped by default, at least with respect to the risk measures utilized for our risk-aware verification and synthesis analyses.

Chapter Structure

Section 5.2 details our efforts in risk-aware safety-critical system verification, and Section 5.3 details our efforts in risk-aware safety-critical controller synthesis.

5.2 Risk-Aware Verification

An important application of the sample-based bounds derived in the prior chapter arises in safety-critical system verification where system evolution is partially stochastic due to unmodeled dynamics, noise, *etc*. In this section, we will detail how we can reformulate safety-critical system verification as a risk measure determination problem. This reformulation lets us use our prior results to easily bound system performance in a cooperative multi-agent system setting.

Notation and Problem Setting

For our system under study, $\mathcal{X}$ is the state space, $\mathcal{U}$ is the input space, and Θ is a known space of parameters θ influencing the system's controller U. Furthermore, we assume the system is subject to stochastic noise ξ with an unknown distribution $\pi_\xi(x, u, t)$ over $\mathbb{R}^n$.

$$\begin{aligned} &\dot{x} = f(x,u) + \xi, && x \in \mathcal{X} \subset \mathbb{R}^n,\ u \in \mathcal{U} \subset \mathbb{R}^m, \\ &u = U(x,\theta), && \theta \in \Theta \subset \mathbb{R}^p, \\ &\xi \sim \pi_\xi(x,u,t), && \int_{\mathcal{X}} \pi_\xi(x,u,t,s)\, ds = 1 \ \forall\, x,u,t. \end{aligned} \tag{5.1}$$

We denote x_t^θ as our closed-loop system solution at time t—note that the parameter θ does not change over a trajectory—and x^θ corresponds to our closed-loop solution:

$$\dot{x}_t^\theta = f\left(x_t^\theta, U\left(x_t^\theta, \theta\right)\right) + \xi, \quad x^\theta \in \mathcal{S}^{\mathbb{R}^n},\ \mathcal{S}^{\mathbb{R}^n} = \{s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n\}$$

Verification work typically assumes the existence of a robustness metric ρ—a function that maps state trajectory signals to the real line, with positive evaluations of the metric indicating system objective satisfaction [19].

Definition 20. A *robustness metric* ρ is a function that maps from the signal space to the reals, *i.e.* $\rho : \mathcal{S}^{\mathbb{R}^n} \to [-a, b],\ a, b \in \mathbb{R}_{++}$. Furthermore, a signal s is "accepted" by the robustness metric ρ, *i.e.* exhibits the desired properties encoded by the robustness metric ρ, if and only if $\rho(s) \geq 0$.

Remark 3. *Here, we note that Definition 20 differs from the traditional statement of robustness measures in Signal Temporal Logic as per Definition 5 in Chapter 2 —*

the former does not depend on time whereas the latter does. This notational change reflects the notion that for robotic systems, we always evaluate signal satisfaction of a behavior from time 0 *onwards, i.e. our robot, once it has been turned on, should start satisfying the criteria required of it.*

Examples of robustness metrics ρ include the minimum value of a control barrier function h over some pre-specified time horizon [145], [146], or the robustness metrics of Signal Temporal Logic [19], [96] as provided in Chapter 2. As the existence and construction of these functions have been well-studied, we will simply assume their existence for the time being [61], [147], [148].

For risk-aware verification, the uncertainty arises through the uncertainty ξ entering the system dynamics in (5.1). Hence, the robustness ρ of a closed loop trajectory x^θ, *i.e.* $\rho(x^\theta)$, is a scalar random variable $R(x_0, \theta)$ with some distribution $\pi_R(x_0, \theta)$ dependent on the initial condition x_0 and parameter θ.

Definition 21. The *trajectory-specific robustness*, $R(x_0, \theta)$, is a scalar-valued random variable with distribution $\pi_R(x_0, \theta)$ corresponding to the closed-loop robustness $\rho(x^\theta)$ of trajectories x^θ emanating from the initial condition $(x_0, \theta) \in \mathcal{X}_0 \times \Theta \triangleq \Phi$.

By further uniformly sampling initial conditions and parameters from their respective spaces *i.e.* sampling (x_0, θ) from $\mathrm{U}[\mathcal{X}_0 \times \Theta]$, we generate the scalar randomized robustness variable R with distribution π_R that was studied in [149].

Definition 22. The *holistic system robustness*, R, is a scalar random variable with distribution π_R and samples r denoting the closed-loop robustness $\rho(x^\theta)$ of trajectories x^θ whose initial condition and parameter (x_0, θ) were sampled uniformly from $\mathcal{X}_0 \times \Theta$, *i.e.* $(x_0, \theta) \sim \mathrm{U}[\mathcal{X}_0 \times \Theta \triangleq \Phi]$.

The goal of risk-aware verification then would be to identify risk measures for this holistic system robustness random variable R as per Definition 22, *i.e.* identify $\mathrm{VaR}_\alpha(R)$, $\mathrm{CVaR}_\alpha(R)$ or $\mathrm{EVaR}_\alpha(R)$ for some $\alpha \in (0, 1]$. Since we do not know the distribution π_R of R, however, direct identification of these risk measures is difficult. Therefore, we will instead identify upper bounds for these risk measures— r_V^* for $\mathrm{VaR}_\alpha(-R)$, r_C^* for $\mathrm{CVaR}_\alpha(-R)$ and r_E^* for $\mathrm{EVaR}_\alpha(-R)$ —that are upper bounds with some minimum probability—$\epsilon_V, \epsilon_C, \epsilon_E$ respectively.

Problem 4. *For the scalar random variable R with distribution π_R as per Definition 22, devise a method to determine upper bounds $r_V^*, r_C^*, r_E^* \in \mathbb{R}$ with corresponding probabilities $\epsilon_V, \epsilon_C, \epsilon_E \in [0,1]$ such that for some $\alpha \in (0,1]$,*

$$\begin{aligned}\mathbb{P}_{\pi_R}[r_V^* \geq \text{VaR}_\alpha(-R)] &\geq 1-\epsilon_V,\\ \mathbb{P}_{\pi_R}[r_C^* \geq \text{CVaR}_\alpha(-R)] &\geq 1-\epsilon_C,\\ \mathbb{P}_{\pi_R}[r_E^* \geq \text{EVaR}_\alpha(-R)] &\geq 1-\epsilon_E.\end{aligned}$$

Here, $\text{VaR}_\alpha(-R), \text{CVaR}_\alpha(-R), \text{EVaR}_\alpha(-R)$ *are the risk measures defined in Definitions 15-17 respectively.*

Upper Bounds for Risk-Aware Verification

To start, we will first motivate why bounding robustness risk measures is useful, as the procedure is non-standard when compared with existing literature. It is easiest to see this utility in bounding robustness risk evaluation for Value-at-Risk. Here, we aim to identify the probabilistic cutoff r_V^* for the random variable $-R$, as it will let us bound the weighted volume of the initial conditions and parameters $(x_0, \theta) \in \Phi$ which have the potential of yielding trajectories whose robustness $r < -r_V^*$. A motivating example of this is shown in Figure 5.1, and we can show this volume bounding as follows. First, we define a function $B : \mathbb{R} \to \mathbb{R}$ outputting the total probability of sampling a trajectory whose robustness is strictly less than a cutoff $y \in \mathbb{R}$. In what follows, $\pi_R(x_0, \theta)$ is the distribution of $R(x_0, \theta)$ as per Definition 21:

$$\begin{aligned}B(y) &= \int_\Phi \frac{1}{\beta} \int_{-\infty}^{y} \pi_R(x_0, \theta, s)\ ds\ d(x_0, \theta), \qquad (5.2)\\ \beta &= \int_\Phi 1\ d(x_0, \theta).\end{aligned}$$

As such, r_V^* is a holistic characterization of system behavior.

Proposition 1. *Let R be the holistic robustness random variable as per Definition 22, let B be as per equation* (5.2)*, and let $\alpha \in (0,1]$. Then,*

$$0 \geq r_V^* \geq \text{VaR}_\alpha(-R) \implies B(0) \leq \alpha$$

Proof: To start, we have the following integral inequality:

$$\int_\Phi \frac{1}{\beta} \int_{-r_V^*}^{\infty} \pi_R(x_0, \theta, s)\ ds\ d(x_0, \theta) \geq 1-\alpha.$$

Furthermore, as π_R is a valid probability distribution,

$$\int_\Phi \frac{1}{\beta} \int_{-r_V^*}^{\infty} \pi_R(x_0, \theta, s)\ ds\ d(x_0, \theta) + B(-r_V^*) = 1.$$

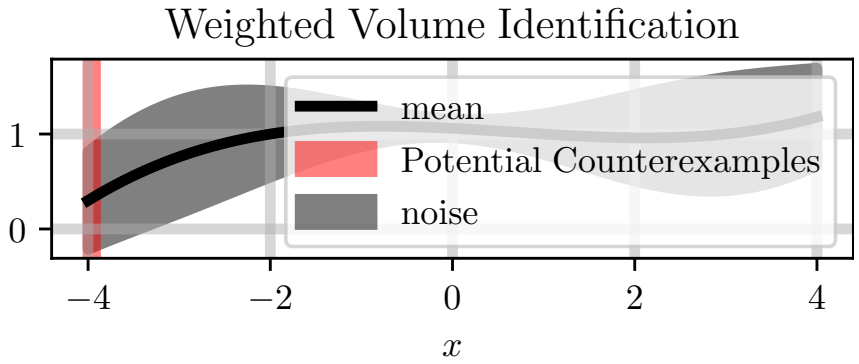

Figure 5.1: The above figure provides context for why we choose to take a randomized, risk-aware approach to verification. Doing so lets us upper bound by an $\epsilon \in [0, 1]$ the weighted volume of the states in the red region shown. For verification purposes, this bounds the total risk of sampling trajectories whose robustness $r < -r_V^*$ as stated in Proposition 1.

As a result, $B(-r_V^*) \leq \alpha$. Then, as $\pi_R(x_0, \theta, s) \in [0, 1]$, $\forall\ s \in \mathbb{R}$, for all $a \leq b$, $B(a) \leq B(b)$. Since $-r_V^* \geq 0$ and $B(-r_V^*) \leq \alpha$ then, so to is $B(0) \leq \alpha$. ■

Therefore, if we fix a minimum probability by choosing an $\alpha \in (0, 1]$ and find that the corresponding probabilistic robustness lower bound $-r_V^* \geq 0$, then the total probability $B(0)$ of sampling a trajectory whose robustness $r < 0$ is bounded above by α. As we uniformly sample initial conditions and parameters (x_0, θ), this total probability $B(0)$ also corresponds to the weighted volume of initial conditions and parameters $(x_0, \theta) \in \Phi$ which could yield trajectories x^θ whose robustness $\rho(x^\theta) < 0$. The weights $w(x_0, \theta)$ for this weighted volume is the probability of that initial condition and parameter pair (x_0, θ) realizing such a trajectory, *i.e.*

$$B(0) = \int_\Phi \frac{w(x_0, \theta)}{\beta}\, d(x_0, \theta),$$
$$w(x_0, \theta) = \int_{-\infty}^{0} \pi_R(x_0, \theta, s)\, ds.$$

If there were no uncertainty in our system, *i.e.* if (5.1) was deterministic, then $B(0)$ directly corresponds to an upper bound on the volume fraction of those initial condition and parameter pairs that yield trajectories whose robustness $\rho(x^\theta) < 0$.

Simplicity in Finding $\mathbf{r_V^*}$: Now that we have motivated why we might want to find such a probabilistic bound r_V^*, it remains to find such a bound. Here we can leverage our prior results in the following corollary.

Corollary 15. *Let ζ_N^* be the solution to* (UB-RP-N) *for an N-sample set $\{x_k = -r_k\}_{k=1}^N$ of the random variable R with distribution π_R as per Definition 22. Then,*

$$\forall\, \alpha \in (0,1],\ \mathbb{P}_{\pi_R}^N \left[r_V^* \triangleq \zeta_N^* \geq \mathrm{VaR}_\alpha(-R) \right] \geq 1 - (1-\alpha)^N.$$

Proof: This results from Theorem 6. ∎

Corollary 15 tells us that we can identify an estimate ζ_N^* to our desired statistic r_V^* for any confidence level $1-\alpha$ if we take a sufficiently large number of samples N of the random variable R. However, it does not state how many samples N are required to determine this estimate with high probability. Theorem 12 formalizes this sample requirement. Specifically, Theorem 12 states that the number of samples N required to achieve high confidence γ in our estimate ζ_N^* is only a function of the desired risk level α and confidence γ.

Theorem 12. *Let $\alpha, \gamma \in (0,1)$, and let ζ_N^* be the solution to* (UB-RP-N) *for an N-sample set $\{x_k = -r_k\}_{k=1}^N$ of the random variable R with distribution π_R as per Definition 22. Then,*

$$N \geq \frac{\log(1-\gamma)}{\log(1-\alpha)} \implies \mathbb{P}_{\pi_R}^N \left[r_V^* \triangleq \zeta_N^* \geq \mathrm{VaR}_\alpha(-R) \right] \geq \gamma.$$

Proof: Via Corollary 15 we have the following inequality:

$$\mathbb{P}_{\pi_R}^N \left[\mathbb{P}_{\pi_R}[r \geq -\zeta_N^*] \geq 1-\alpha \right] \geq 1 - (1-\alpha)^N.$$

As $1-\alpha \in (0,1)$, if $N \geq \frac{\log(1-\gamma)}{\log(1-\alpha)}$ then substituting and simplifying the right-hand side of the above inequality provides the desired result. ∎

Notably, Theorem 12's result is independent of the dimension of the system's state and parameter space, *i.e.* independent of ℓ where $\mathcal{X}_0 \times \Theta \subseteq \mathbb{R}^\ell$. This is why we claim we have made a step towards sample-efficient risk-aware safety-critical system verification, as independent of system complexity, Theorem 12 identifies the minimum number of samples required to verify system behavior. These results on dimensional scaling also hold for our procedures to upper bound coherent risk measure evaluation of the randomized robustness variable, as will be described.

Specializing to CVaR and EVaR: First, we note that via Definitions 22 and 20, the negation of the holistic system robustness R has a probabilistic upper bound $a \in \mathbb{R}$, i.e. $\mathbb{P}_{\pi_R}[-r \leq a] = 1$. Therefore, we can directly apply Corollaries 13 and 14 to solve Problem 4. To formally state our results in this vein, we will redefine the loss

function L and upper bounding function u_b for each risk measure in this specific application. Here, a is the upper bound for $-R$:

$$L(x,\mu,t) = t\left(\mu + \frac{1}{\alpha}\max\left\{\frac{x}{t} - \mu, 0\right\}\right), \qquad u_b(\mu,t) = L(a,\mu,t), \qquad \text{(CVaR)}$$

$$L(x,\mu,t) = t\left(\mu + e^{\frac{x}{t}-\mu-\ln(\alpha)-1}\right), \qquad u_b(\mu,t) = L(a,\mu,t). \qquad \text{(EVaR)}$$

First, we have that we can upper bound $\text{CVaR}_\alpha(-R)\ \forall\ \alpha \in (0,1]$.

Corollary 16. *Let R be a scalar R.V. as per Definition 22, let $\alpha \in (0,1]$, let L, u_b be as defined in equation* (CVaR) *with respect to this α, let $\epsilon \in (0,1)$ and $\gamma \in [0,1)$, and let $\zeta_N^*(\mu,t)$ be the solution to* (UB-RP-N) *for a set of N-samples $\{y_k = L(-r_k,\mu,t)\}_{k=1}^N$ of the random variable $Y = L(-R,\mu,t)$, where $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$. Then,*

$$\mathbb{P}_{\pi_R}^N\left[r_C^* \triangleq \inf_{\mu\in\mathbb{R},\ t>0}\ \zeta_N^*(\mu,t)(1-\epsilon) + u_b(\mu,t)\epsilon \geq \text{CVaR}_\alpha(-R)\right] \geq \gamma.$$

Proof: This is an application of Corollary 13 with L and u_b as per equation (CVaR). The R.V. $-R$ has as its upper bound $a \in \mathbb{R}$ as per Definitions 20 and 22. Finally, as $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$, $1-(1-\epsilon)^N \geq \gamma$. ■

Likewise, we can also upper bound $\text{EVaR}_\alpha(-R)\ \forall\ \alpha \in (0,1]$.

Corollary 17. *Let R be a scalar random variable as per Definition 22, let $\alpha \in (0,1]$, let L, u_b be as defined in equation* (EVaR) *with respect to this α, let $\epsilon \in (0,1)$ and $\gamma \in [0,1)$, and let $\zeta_N^*(\mu,t)$ be the solution to* (UB-RP-N) *for a set of N-samples $\{y_k = L(-r_k,\mu,t)\}_{k=1}^N$ of the random variable $Y = L(-R,\mu,t)$, where $N \geq \frac{\log(1-\gamma)}{\log(1-\epsilon)}$. Then,*

$$\mathbb{P}_{\pi_R}^N\left[r_E^* \triangleq \inf_{\mu\in\mathbb{R},\ t>0}\ \zeta_N^*(\mu,t)(1-\epsilon) + u_b(\mu,t)\epsilon \geq \text{EVaR}_\alpha(-R)\right] \geq \gamma.$$

Proof: This is an application of Corollary 14 and follows in the footsteps of the proof for Corollary 16. ■

Examples

To showcase the results of Corollary 16 and 17, we will verify a cooperative multi-agent robotic system. Specifically, we will identify lower bounds on expected worst-case system performance, *i.e.* upper bounds on both $\text{CVaR}_{0.1}(-R)$ and $\text{EVaR}_{0.1}(-R)$, for a multi-agent system that is to avoid self-collisions while

each agent reaches their respective goal. As a case study, we will use the Georgia Tech Robotarium, wherein all the robots can be modeled as unicycles [150]:

$$x = \begin{bmatrix} x, \\ y, \\ \theta \end{bmatrix}, \ \dot{x} = \begin{bmatrix} v\cos(\theta), \\ v\sin(\theta), \\ \omega, \end{bmatrix}, \ u = [v, \omega]^T, \tag{5.3}$$
$$\mathcal{X} = [-1, 1] \times [-0.6, 0.6] \times [0, 2\pi], \ M = [I_2, \ \mathbf{0}_{2x1}].$$

As mentioned in [150], a Lyapunov-based controller drives each agent x^i to their desired orientation $x_d^i \in \mathcal{X}$, and control inputs are filtered in a barrier-based quadratic program to ensure that robots do not collide when multiple robots are moving simultaneously [145]. As a result, this barrier-based filter provides a natural robustness measure as per Definition 20. Let the state vector for all 3 robots $\mathbf{x}^T = [x^{1T}, x^{2T}, x^{3T}]$. Then, we can generate two, key functions. One function, h_g, that the system hopes to keep positive, and another function, h_f, that the system hopes to make positive over its trajectory. Here, x^i, x_d^i are the states and desired poses for robot i:

$$h_g(\mathbf{x}) = \min_{i \neq j, \ i,j \in [1,2,3]} \|M(x^i - x^j)\| - 0.15. \tag{5.4}$$

$$h_f(\mathbf{x}) = \max_{i \in [1,2,3]} 0.1 - \|M(x^i - x_d^i)\|. \tag{5.5}$$

Intuitively, $h_g(\mathbf{x}) \geq 0$ implies that the robots are sufficiently far apart, and $h_f(\mathbf{x}) \geq 0$ means that all robots are sufficiently close to their goal. From these two functions, we construct our robustness measure ρ for a state-signal $\mathbf{x}^\theta$ where $\theta^T = [x_d^{1T}, x_d^{2T}, x_d^{3T}]$:

$$\rho_g(\mathbf{x}^\theta) = \min_{t \in [0,30]} h_g\left(\mathbf{x}_t^\theta\right), \quad \rho_f(\mathbf{x}^\theta) = \max_{t \in [0,30]} h_f\left(\mathbf{x}_t^\theta\right),$$
$$\rho(\mathbf{x}^\theta) = \begin{cases} \rho_g(\mathbf{x}^\theta), & \text{if } \rho_g(\mathbf{x}^\theta), \rho_f(\mathbf{x}^\theta) \geq 0, \\ \max\left\{\rho_g(\mathbf{x}^\theta), -0.1\right\}, & \text{if } \rho_g(\mathbf{x}^\theta) < 0, \\ \max\left\{\rho_f(\mathbf{x}^\theta), -0.1\right\}, & \text{else.} \end{cases} \tag{5.6}$$

If $\rho(\mathbf{x}^\theta) \geq 0$ then, all 3 robots stayed at least 0.15 meters from each other for 30 seconds—as $h_g(\mathbf{x}_t^\theta) \geq 0, \ \forall \ t \in [0, 30]$—and reached within 0.1 meters of their goal within 30 seconds—as $h_f(\mathbf{x}_t^\theta) \geq 0$, for some $t \in [0, 30]$. Furthermore, we also know this robustness measure ρ outputs values that are always greater than or equal to -0.1. This robustness measure ρ also lets us specifically define our holistic system robustness R in this case. For this example, samples r of R arise when we first uniformly randomly sample initial locations and parameters $(\mathbf{x}_0, \theta)$ from their

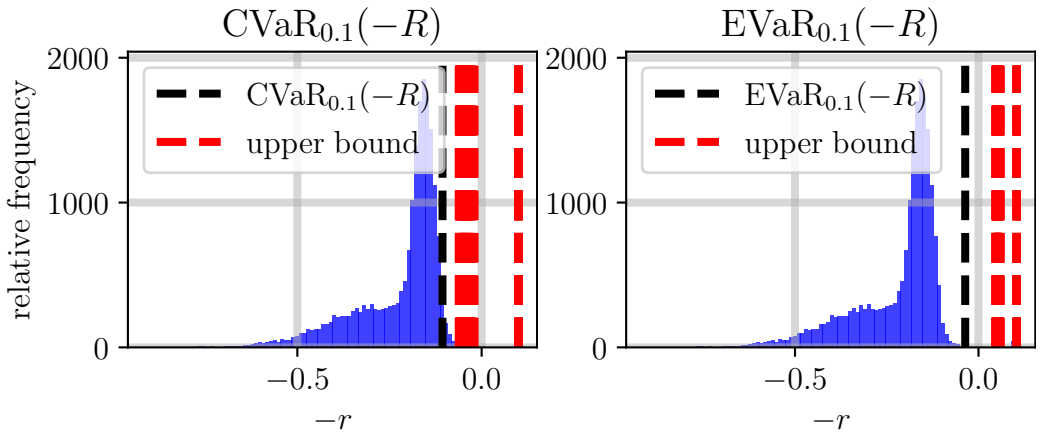

Figure 5.2: We upper bound the risk measures of a multi-agent robotic system when its state trajectory is evaluated through a robustness metric. Shown above is this upper bounding procedure for both CVaR (left) and EVaR (right). For each of the 50 trials, the upper bounds (red) for both risk measures are indeed greater than or equal to their "true" counterparts (black). These "true" counterparts were calculated by taking 20000 samples of the randomized system robustness R (Definition 22), and the distribution of samples is shown (blue). The fact that the upper bounds are indeed upper bounds over all trials serves as a numerical confirmation of Corollaries 16 and 17. They also support the repeatability of our procedure in identifying upper bounds to g-entropic risk measures with high probability.

respective spaces below:

$$\mathcal{X}_0 = \{\mathbf{x} \in \mathcal{X}^3 \mid h_g(\mathbf{x}) \geq 0.3\}, \quad \Theta = \{\mathbf{x} \in \mathcal{X}^3 \mid h_g(\mathbf{x}) \geq 0.3\}.$$

Then, we record the corresponding state trajectory of the multi-agent system $\mathbf{x}^\theta$ for at-least 30 seconds. The holistic system robustness sample $r = \rho(\mathbf{x}^\theta)$ then, with ρ as defined in (5.6).

Verifying a 3 **Robot System:** Our goal is to determine an upper bound on both the $\mathrm{CVaR}_\alpha(-R)$ and $\mathrm{EVaR}_\alpha(-R)$ with $\alpha = 0.1$ for the three robots as they carry out their task. Furthermore, we hope to determine this upper bound with 95% confidence, *i.e.* $\gamma = 0.95$, and we require that $\epsilon = 0.02$. For this case, Corollaries 16 and 17 require that we take at minimum $N \geq 149$ samples of our holistic system robustness R to determine this upper bound. As a result, we uniformly sampled $N = 149$ initial conditions and parameters $(\mathbf{x}_0, \theta)$ from $\mathcal{X}_0 \times \Theta$, recorded the corresponding state trajectories $\mathbf{x}^\theta$, and recorded their robustnesses $r = \rho(\mathbf{x}^\theta)$. We also repeated this data collection procedure 50 times to ensure that our results are repeatable, and to identify the true $\mathrm{CVaR}_\alpha(-R)$ and $\mathrm{EVaR}_\alpha(-R)$ we repeated this data collection procedure once more but took $N = 20000$ samples instead. If Corollaries 16 and 17

are correct, then with 95% confidence, we expect that our sampled upper bounds for the 50 trials performed are greater than or equal to their true counterparts. All this data is shown in Figure 5.2, and as prior, the black lines indicate the true risk measure evaluation and all red lines are the upper bounds generated per trial. As can be seen, all generated upper bounds are indeed upper bounds for their counterparts which serves as a numerical confirmation of Corollaries 16 and 17 and also of the repeatability of our procedure in identifying upper bounds to these risk measures.

Brief Remark on Relative Bound Tightness: Figure 5.2 does prompt one question, however. Specifically, it appears the upper bounds for CVaR are tighter than those for EVaR. We anticipate this might arise as CVaR is a less conservative risk measure than EVaR. As a result, for a given confidence level γ in our upper bound, the upper bound for EVaR will likely be looser on account of this conservatism.

5.3 Risk-Aware Policy Synthesis

Finally, we can now combine the sample-based approaches developed in all prior sections and chapters. Specifically, we will phrase risk-aware policy synthesis as an optimization problem where the objective function accounts for uncertainty in initial system states, controller inputs, and system evolution, and the design space accounts for the uncertainty during optimal policy synthesis. To "solve" this optimization problem, we will use the sample-based approach to identifying "good" policies as developed in Chapter 4. Here, each policy sample taken is the result of our sample-based verification approach detailed in the prior section. As a result, we detail a risk-aware policy synthesis technique with tractable sample requirements that easily scales to high-dimensional systems and potentially non-convex policy objectives.

Setting and Problem Statement

Specifically, we still focus on a general class of systems with the state space, $\mathcal{X}$; the input space, $\mathcal{U}$; a space of problem parameters Θ that the controller U must account for. The discrepancy with (5.1) arises in the parameterization of the controller U via the parameter space P of controller parameters. As before, we assume the system is subject to some stochastic noise ξ with an unknown distribution $\pi_\xi(x, u, t)$ over $\mathbb{R}^n$.

$$\begin{aligned} &\dot{x} = f(x,u) + \xi, && x \in \mathcal{X} \subset \mathbb{R}^n,\ u \in \mathcal{U} \subset \mathbb{R}^m, \\ &u = U(x,\theta,p), && \theta \in \Theta \subset \mathbb{R}^l,\ p \in P \subset \mathbb{R}^s, \\ &\xi \sim \pi_\xi(x,u,t), && \int_{\mathcal{X}} \pi_\xi(x,u,t,s)\, ds = 1 \ \forall\, x,u,t. \end{aligned} \tag{5.7}$$

As before, we will define $x_t^{\theta,p}$ as the solution to our closed-loop system at time t, given the initial condition $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$, an accounting parameter $\theta \in \Theta$, and a set of controller parameters $p \in P$.

$$\dot{x}_t^{\theta,p} = f\left(x_t^{\theta,p}, U\left(x_t^{\theta,p}, \theta, p\right)\right) + \xi,\ x^{\theta,p} \in \mathcal{S}^{\mathbb{R}^n},\ \mathcal{S}^{\mathbb{R}^n} = \{s : \mathbb{R}_{\geq 0} \to \mathbb{R}^n\}.$$

Then, our goal will be to identify a set of parameters $p \in P$ such that the ensuing controller U is better than, say, 90% of any possible controller we could develop by changing parameters $p \in P$. To delineate between controllers, we aim to maximize the lower bound on the robustness risk of the closed-loop system—the lower bound outputted by the sample-based risk-aware verification approach detailed in the prior section. Specifically, choosing a parameter $p \in P$ defines a closed-loop system—that system for which the controller U is parameterized via this parameter p. Then for this system, we can calculate a lower bound on the Conditional-Value-at-Risk of the system's robustness ρ over the entire space of initial conditions and accounting parameters $\mathcal{X}_0 \times \Theta$—this is Corollary 16. We could follow a similar process for upper bounding the Entropic-Value-at-Risk as well. However, we will focus on CVaR for brevity as the procedure for any other risk measures is effectively the same. As such, the natural risk-aware policy synthesis aim is to maximize this lower bound.

Formally stating this problem first requires a modification of Definition 22 for the parameterized setting.

Definition 23. The *parameterized holistic system robustness*, R_p, is a scalar random variable with distribution π_{R_p} and samples r_p denoting the closed-loop robustness $\rho(x^{\theta,p})$ of trajectories $x^{\theta,p}$ whose initial condition and parameter (x_0, θ) were sampled uniformly from $\mathcal{X}_0 \times \Theta$ and whose controller U is parameterized via the controller parameter $p \in P$ as per equation (5.7)

Per Corollary 16, if we fix the parameter p for the controller U, we can generate an upper bound $r^*_{C,p}$ on $\text{CVaR}_\alpha(-R_p)$ for any $\alpha \in (0, 1]$ with high-confidence. As we study only CVaR in this section, we will drop the subscript C for $r^*_{C,p}$ to simplify notation. Mathematically, this lets us generate a mapping $\mathcal{R}$ from controller parameter p, confidence γ, and risk-level α, to the upper bound r^*_p generated with confidence γ for $\text{CVaR}_\alpha(-R_p)$. Implicit in this function definition will be the number of samples $N_\mathcal{R}$ we take of the initial conditions and accounting parameters (x_0, θ) from the uniform distribution over $\mathcal{X}_0 \times \Theta$:

$$\mathcal{R}(p, \gamma, \alpha) = r^*_p,\ \text{s.t.}\ \mathbb{P}^{N_\mathcal{R}}_{\pi_{R_p}}\left[r^*_p \geq \text{CVaR}_\alpha(-R_p)\right] \geq \gamma. \tag{5.8}$$

Then, by Definitions 15 and 16, we have the following proposition indicating that the risk map $\mathcal{R}$ serves as a high-confidence lower bound on the expected worst-case system performance.

Proposition 2. *Let $\mathcal{R}$ be as defined in equation* (5.8) *for some $\gamma \in [0, 1)$ and $\alpha \in (0, 1]$. The following inequality holds, with π_{R_p} the distribution of the parameterized holistic system robustness R_p defined in Definition 23:*

$$\mathbb{P}^{N_{\mathcal{R}}}_{\pi_{R_p}} \left[-\mathcal{R}(p, \gamma, \alpha) \leq \mathbb{E}_{\pi_{R_p}} [r \mid r \leq \text{VaR}_{1-\alpha}(R_p)] \right] \geq \gamma.$$

Proof: By definition of CVaR_α for any risk-level $\alpha \in (0, 1]$ in Definition 16, we have the following equivalency:

$$\text{CVaR}_\alpha(-R_p) = \mathbb{E}_{\pi_{R_p}} [r_p \mid -r_p \geq \text{VaR}_\alpha(-R_p)].$$

Then by definition of VaR and a symmetry argument when flipping the random variable of interest, we have the following equality:

$$-\text{CVaR}_\alpha(-R_p) = \mathbb{E}_{\pi_{r_p}} [r_p \mid r_p \leq \text{VaR}_{1-\alpha}(R_p)].$$

Then the desired inequality stems from the definition of $\mathcal{R}$ in equation (5.8). ■

As a result, we state that the goal of policy synthesis should be to minimize $\mathcal{R}(p, \gamma, \alpha)$. Specifically, minimization of $\mathcal{R}(p, \gamma, \alpha)$ over P for some fixed γ, α corresponds to the identification of a set of controller parameters $p \in P$ such that in the worst $100\alpha\%$ of cases, the expected minimum system robustness is maximized. Since the robustness measure ρ outputs positive numbers for those trajectories that satisfy the desired behavior (Definition 20), if this maximum lower bound were positive, then the expected performance in the worst $100\alpha\%$ of cases would still exhibit satisfactory behavior. This leads to the nominal, risk-aware synthesis problem:

$$\min_{p \in P} \ \mathcal{R}(p, \gamma, \alpha), \text{ for some } \gamma, \alpha \in (0, 1), \tag{5.9}$$

which is similar to the general problem (4.9) studied in the prior chapter. The decision space $D = P$ and the reward function $R = -\mathcal{R}(\gamma, \alpha)$. We can also rewrite the volume fraction function $\mathcal{V}$ (4.10) and "better" policy map F (4.11):

$$\mathcal{V}(A) = \frac{\int_A 1 \, dx}{\int_P 1 \, dx}, \quad F(p, \gamma, \alpha) = \{p' \in P \mid \mathcal{R}(p', \gamma, \alpha) < \mathcal{R}(p, \gamma, \alpha)\}. \tag{5.10}$$

This lets us formally define the problem under study for this section.

Problem 5. *Let $\mathcal{R}$ be as defined in* (5.8) *with respect to some $\alpha \in (0, 1]$ and $\gamma \in (0, 1)$. Let $\epsilon \in (0, 1)$ as well. Find a set of controller parameters $p \in P$ such that the corresponding controller U is at-least in the $100(1-\epsilon)$-th percentile of all possible controllers with respect to minimization of $\mathcal{R}$, i.e. find $p \in P$ such that $\mathcal{V}(F(p)) \leq \epsilon$ where $\mathcal{V}, F$ are defined in* (5.10).

Identification of "Good" Risk-Aware Policies

To start, Problem 5 is a refinement of Problem 2. As such, we start by constructing a similar scenario program to (RP-G):

$$\begin{aligned} \zeta_N^* = \underset{\zeta \in \mathbb{R}}{\text{argmin}} \quad & \zeta, \\ \text{subject to} \quad & \zeta \geq y_i,\ y_i \in \{y_k = \mathcal{R}(p_k, \gamma, \alpha)\}_{k=1}^N, \\ & p_k \text{ are drawn uniformly from } P. \end{aligned} \tag{RP-PS}$$

Then, identification of "good" policies is a direct consequence of Theorem 9, as stated in the following corollary.

Assumption 12. $\mathcal{R}$ is as defined in equation (5.8) with respect to some $\alpha \in (0, 1]$ and $\gamma_1 \in [0, 1)$. $\epsilon \in (0, 1)$, $\gamma_2 \in [0, 1)$, ζ_N^* is the solution to (RP-PS) for a set of N-samples $\{y_k = \mathcal{R}(p_k, \gamma_1, \alpha)\}_{k=1}^N$ where each p_k was drawn uniformly from P, and $\mathcal{V}, F$ are as defined in equation (5.10).

Corollary 18. *Let Assumption 12 hold. If $N \geq \frac{\log(1-\gamma_2)}{\log(1-\epsilon)}$, then with minimum probability γ_2 there exists at least one parameter set $p_i \in \{p_k\}_{k=1}^N$ whose associated controller U is at-least in the $100(1-\epsilon)$-th percentile of controllers possible with respect to minimizing $\mathcal{R}$, i.e.*

$$\exists\ p_i \in \{p_k\}_{k=1}^N,\ \text{s.t. } \mathbb{P}_{\mathrm{U}[P]}^N \left[\mathcal{V}(F(p_i)) \leq \epsilon\right] \geq \gamma_2, \text{ and } \zeta_N^* = \mathcal{R}(p_i, \gamma_1, \alpha).$$

Proof: This is a direct application of Theorem 9. ∎

To summarize, for each uniformly sampled controller parameterization $p \in P$, we evaluate the corresponding controller U against $N_{\mathcal{R}}$ uniformly sampled initial conditions and accounting parameters $(x_0, \theta) \in \mathcal{X}_0 \times \Theta$ and evaluate the robustness $\rho(x^{\theta,p})$ of the corresponding trajectories. By evaluating enough such trajectories, we generate—via Corollary 16 and Proposition 2—a lower bound $-r^*$ on the expected worst-case system performance in the worst $100\alpha\%$ of cases. By repeating this procedure for enough samples—$N \geq \frac{\log(1-\gamma_2)}{\log(1-\epsilon)}$ samples—we are guaranteed

with minimum probability γ_2 to find at least one parameter set p_i in the sampled set $\{p_k\}_{k=1}^N$ such that the corresponding controller U is at-least in the $100(1-\epsilon)$-th percentile of all possible controllers with respect to maximizing this lower bound. This probabilistic guarantee is due to Theorem 9. As we also provide a minimum sample requirement on the number of controllers required to be tested in this procedure, we state that we have a fundamental sample requirement to generate "good" risk-aware policies with high confidence. Furthermore, as this sample requirement is based on both our desired confidence γ and desired percentile level $1-\epsilon$, this procedure also identifies the relative complexity in the identification of a "better" policy, insofar as it provides a sample requirement for doing so.

Examples

In this section, we will synthesize a risk-aware controller for the cooperative multi-agent system whose controller was verified in Section 5.2, and we will show that the synthesized controller outperforms the baseline controller when accounting for worst-case system performance. To reiterate, we will use the robotarium as a case study again, where the robots can be modeled as unicycles [150]. The dynamics are the same as in (5.3) in Section 5.2.

Different from the verification case in Section 5.2, however, we will now assume that the hybrid Lyapunov-barrier controller the robots are equipped with [150] is now to be optimized. Since we only assume knowledge of the controller parameters p and not the specific controller form U, we will refrain from specifying the controller in favor of stating that the controller parameters we can vary are the gains for the Lyapunov controller: approach angle gain p_1, desired angle gain p_2, rotation error gain p_3, and decay constant p_4 for the barrier filter. As such, the space of parameters

$$P = [0.2, 5]^3 \times [0.1, 200],\ p_{1,2,3} \in [0.2, 5]^3,\ p_4 \in [0.1, 200].$$

To determine a "good" parameter set $p \in P$ without knowledge of how these parameters affect the controller U, we require a robustness measure ρ (Definition 20) to act as a determiner of satisfactory system performance. As such, we will use the same robustness measure ρ in equation (5.6) in Section 5.2 to facilitate a risk-aware verification comparison between our to-be-calculated controller and the baseline controller for the Robotarium. To be clear, we will reproduce this robustness measure ρ in its application to our parameterized system trajectories $\mathbf{x}^{\theta,p}$, where

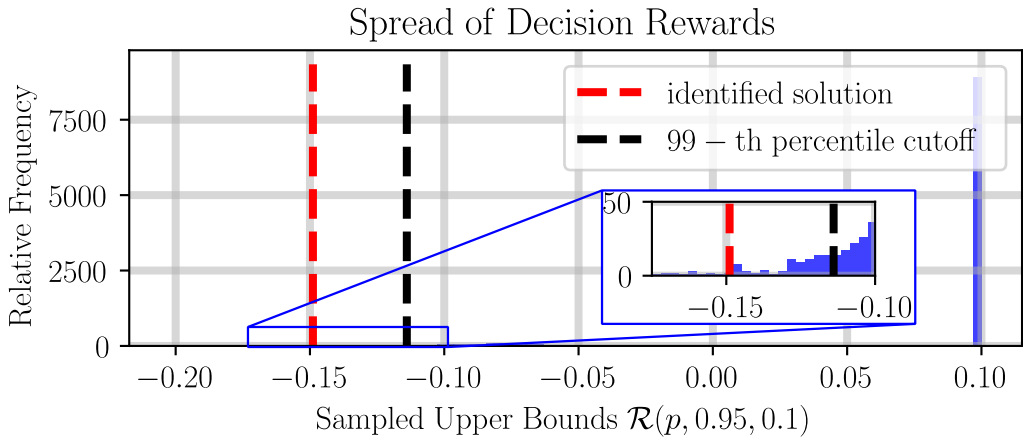

Figure 5.3: We can utilize all sample-based bounds developed in Chapters 4 and 5 to formalize a pipeline for risk-aware controller synthesis, the results for which are shown here. Our goal is to identify a parameterized controller that maximizes a lower bound on worst-case system performance, *i.e.* minimizes an upper bound, $\mathcal{R}(p, 0.95, 0.1)$, over a set of controller parameters $p \in P$. Shown above in blue is the distribution of this upper bound for all controller parameters $p \in P$ and was generated by taking 20000 uniform parameter samples p and evaluating $\mathcal{R}(p, 0.95, 0.1)$. As per the decision selection process detailed in Chapter 4, our goal is to identify a controller in the 99-th percentile with respect to minimization of this upper bound with the true 99-th percentile cutoff shown in black and all controllers yielding upper bounds to its left lying in the 99-th percentile. As can be seen, our identified solution (red) achieves an upper bound in at least the 99-th percentile. This serves as a numerical confirmation of both Theorem 9 and Corollary 18 insofar as we evaluated the minimum number of controllers prescribed, $N = 459$ controllers, to calculate our solution which meets our desired criteria.

h_g, h_f are as defined in equations (5.4) and (5.5) respectively:

$$\rho_g(\mathbf{x}^{\theta,p}) = \min_{t\in[0,30]} h_g\left(\mathbf{x}_t^{\theta,p}\right), \quad \rho_f(\mathbf{x}^{\theta,p}) = \max_{t\in[0,30]} h_f\left(\mathbf{x}_t^{\theta,p}\right),$$

$$\rho(\mathbf{x}^{\theta,p}) = \begin{cases} \rho_g(\mathbf{x}^{\theta,p}), & \text{if } \rho_g(\mathbf{x}^{\theta,p}), \rho_f(\mathbf{x}^{\theta,p}) \geq 0, \\ \max\left\{\rho_g(\mathbf{x}^{\theta,p}), -0.1\right\}, & \text{if } \rho_g(\mathbf{x}^{\theta,p}) < 0, \\ \max\left\{\rho_f(\mathbf{x}^{\theta,p}), -0.1\right\}, & \text{else.} \end{cases}$$

Then, to take a sample r_p of our randomized robustness R_p as per Definition 23, we first uniformly sample (x_0, θ) from the spaces below:

$$\mathcal{X}_0 = \{\mathbf{x} \in \mathcal{X}^3 \mid h_g(\mathbf{x}) \geq 0.3\}, \quad \Theta = \{\mathbf{x} \in \mathcal{X}^3 \mid h_g(\mathbf{x}) \geq 0.3\}.$$

Then we evaluate the robustness of the corresponding state trajectory, *i.e.* $r_p = \rho(\mathbf{x}^{\theta,p})$. Our goal is to identify a parameter set $p \in P$ that minimizes the upper bound

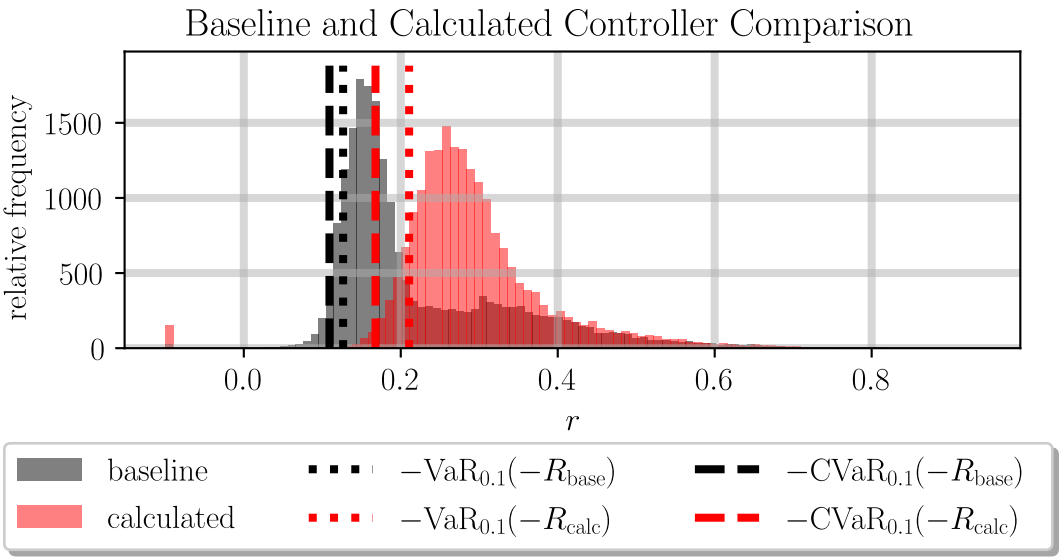

Figure 5.4: A comparison between the baseline controller provided with the robotarium—the controller that was probabilistically verified in Section 5.2—and our calculated controller identified in Figure 5.3. Our risk-aware policy synthesis goal is to identify a controller in the 99-th percentile with respect to maximizing the lower bound on the expected worst-case system performance in the worst 10% of cases—*i.e.* maximizes a lower bound on $-\text{CVaR}_{0.1}(-R)$ as expressed in Proposition 2. Shown above is the distribution of this randomized robustness R for the calculated controller (R_{calc} in red) and the baseline controller provided with the robotarium (R_{base} black). As can be seen, the calculated controller outperforms the baseline controller insofar as the worst-case robustness value for 10% of cases $-\text{VaR}_{0.1}(-R_{\text{calc}}) \geq -\text{VaR}_{0.1}(-R_{\text{base}})$, and the expected value in the worst 10% of cases $-\text{CVaR}_{0.1}(-R_{\text{calc}}) \geq -\text{CVaR}_{0.1}(-R_{\text{base}})$ as well.

r_p^* for $\text{CVaR}_{\alpha=0.1}(-R_p)$ identified with 95% confidence with $N_{\mathcal{R}} = 149$ samples. By definition of $\mathcal{R}$ in equation (5.8), this results in the following optimization problem, reminiscent of (5.9)

$$\min_{p \in P} \ \mathcal{R}(p, 0.95.0.1), \ N_{\mathcal{R}} = 149.$$

As such, we hope to identify a policy that is in the 99−th percentile with 99% confidence, *i.e.* $\gamma = 0.99$ and $\epsilon = 0.01$.

Figure Analysis: According to Corollary 18, to generate a parameter set $p_i \in P$ whose controller U is in the 99-th percentile of all controllers with minimum probability $\gamma = 0.99$, we are required to test $N \geq 459$ uniformly randomly sampled parameters $p \in P$. After doing so, we identified a parameter set $p_i \in P$ that realized an upper bound $\mathcal{R}(p_i, 0.95, 0.1) = -0.1489$. As can be seen in Figure 5.3, the controller corresponding to this policy is indeed in at least the 99-th percentile with respect to all controllers achievable via this parameterization. This serves as a

numerical confirmation of Corollary 18 and Theorem 9. Figure 5.4 shows data for the comparison between the distribution of the robustness of trajectories realized by this controller U_{calc} with the baseline controller with which the robotarium comes equipped U_{base}. Notice that our identified controller outperforms the baseline robotarium controller for our robustness measure of interest, indicating that our synthesized policy was safer — in a risk-aware sense — than the on-board policy.

5.4 Conclusion

This chapter detailed our efforts in the formalization of risk-aware controller verification and synthesis pipelines. Specifically, it detailed how risk-aware controller verification can be posed as a risk-measure estimation problem and leveraged mathematical results from a prior chapter to identify the estimate. Similarly, it detailed how risk-aware controller synthesis can be expressed as an optimization problem maximizing the lower bound offered by the prior risk-measure estimation procedure. Finally, it detailed the application of these techniques on a few simulated examples. However, the success of these techniques on simulated problems motivates questions as to their applicability in all aspects of the controller synthesis paradigm, including hardware implementations of verified controllers. Describing these results is the focus of the next chapter.

Chapter 6

PROBABILISTIC GUARANTEES FOR OPTIMAL CONTROL

This chapter was adapted from:

[1] P. Akella, W. Ubellacker, and A. D. Ames, "Safety-Critical Controller Verification via Sim2Real Gap Quantification," *2023 International Conference on Robotics and Automation (ICRA) (Accepted)*, arXiv:2209.09337, Sep. 2022. DOI: 10.48550/arXiv.2209.09337. arXiv: 2209.09337 [eess.SY].

[2] P. Akella, S. X. Wei, J. W. Burdick, and A. D. Ames, "Learning Disturbances Online for Risk-Aware Control: Risk-Aware Flight with Less Than One Minute of Data," *Conference on Learning for Dynamics and Control (L4DC) (Accepted)*, arXiv:2212.06253, arXiv:2212.06253, Dec. 2022. DOI: 10.48550/arXiv.2212.06253. arXiv: 2212.06253 [eess.SY],

[3] P. Akella, W. Ubellacker, and A. D. Ames, "Probabilistic Guarantees for Nonlinear Safety-Critical Optimal Control," *2023 International Conference Intelligent Robots and Systems (IROS) (Submitted)*, arXiv:2303.06258, Mar. 2023. DOI: 10.48550/arXiv.2303.06258. arXiv: 2303.06258 [math.OC],

Finally, this last chapter of the thesis will detail how we can utilize all the prior methods—developed to provide guarantees for safety-critical system verification — to facilitate controller synthesis across all aspects of the nominal synthesis paradigm. Effectively, this chapter "closes the loop" on the duality between verification and synthesis, at least with respect to the provided methods. To that end, it will describe our efforts in model verification, controller synthesis (both offline and online) against stochastic models generated from the prior verification procedure, and the utility of percentile methods to rapidly generate "good" control inputs for finite-time optimal controllers and facilitate guarantee determination for the same optimal controllers. To start, we will provide a more in-depth introduction of related works corresponding to the contributions detailed in the chapter.

6.1 Introduction

The nominal controller synthesis process for safety-critical systems follows a well-worn path: (1) develop a model for the system of interest (from first principles, system identification, or otherwise); (2) develop a controller in simulation based on

this model; (3) implement the controller on the safety-critical system of interest; and (4) tune controller parameters until the system exhibits the desired behavior. This chapter details how we can leverage the prior results in test synthesis and verification to facilitate rapid, verifiable controller synthesis for safety-critical systems. In addition, not only can we streamline the controller synthesis and verification processes, but we will also be able to rapidly synthesize control inputs for optimal controllers using our sample-based percentile optimization procedures as well.

To that end, the first of our results will offer a method that aims to mitigate the need for extensive—and perhaps expensive—hardware testing and verification, by simultaneously verifying the simulator used for controller synthesis and the controller itself. Our desire to augment existing model generation techniques stems from a desire to leverage extensive prior work in system identification and reduced-order-model control. More specifically, System identification specifically deals with techniques aimed at generating models that more closely align with the system-to-be-modeled and is a very well-studied field [151]–[159]. Despite the capability of these existing methods to generate good models, they oftentimes fail to capture rarer system behavior, *i.e.* corner cases, stochastic disturbances, *etc*. Moreover, prior work indicates that reduced order models are oftentimes sufficient to express underlying system physics [160]–[162]. These simpler models are also often leveraged in autonomy stacks for complex systems. Additionally, augmenting these reduced order models with uncertainty bounds and accounting for these bounds through robust control [161], [163], input-to-state stabilizing barrier functions [146], [164], [165], or other techniques, *i.e.* learning [166]–[168], tends to yield safe and reliable controllers. This prompts the question then: how should the "correct" uncertainty bound be determined? If we use too large a bound, the controller might be too conservative, and if we use too small a bound, then the controller could be unsafe. Our first set of results will provide an offline method to answer this question, whereas the second set will provide an online, more conservative method.

The third set of results will dive deeper into the control loop and provide probabilistic guarantees on percentile approaches to input generation for finite-time optimal controllers. The interest in this vein stems from the fact that optimal controllers provide a natural way of expressing and segmenting disparate control objectives, as can be easily seen in works regarding model predictive control (MPC) [169]–[171], control barrier functions [145], [146], [172], and optimal path planning [173]–[175], among others. However, optimization problems becoming central to controller syn-

thesis resulted in newer problems such as determining whether solutions exist, *e.g.* recursive feasibility in MPC, determining the efficiency with which solutions can be identified to inform control loop rates, and determining the optimality of identified solutions in non-convex optimization settings. Recent years have seen tremendous strides in answering these questions, but areas of improvement still exist. For example, advances in Nonlinear MPC still require assumptions on the existence of control invariant terminal sets and stabilizing controllers for recursive feasibility, though identification of such items for general nonlinear systems remains a difficult problem [49]–[53]. In general, determination of solution optimality for MPC problems is equivalent to solving the Hamilton-Jacobi-Bellman equation which is known to be difficult [54]. For path-planning problems, RRT* and other, sampling-based methods are known to be probabilistically complete, *i.e.* they will produce the optimal solution given an infinite runtime, though sample-complexity results for sub-optimal solutions are few [174], [176], [177]. Finally, there are similarly few theoretical results on the time complexity of these controllers on hardware systems, as such an analysis is heavily dependent on the specific hardware.

Summary of Chapter Contributions

Our contributions in this vein will be itemized as follows:

1. We provide an offline procedure to calculate a norm bound on the uncertainty between a provided model and its corresponding system. This bound effectively reads as: with minimum probability $1 - \epsilon$, the evolution of the system at one time-step will lie within a calculated polytopic set centered on the state prescribed by model evolution.

2. We synthesize and verify controllers against an uncertain model generated in the prior step, resulting in a pipeline for safety-critical controller synthesis and verification that translates to hardware performance.

3. We define a new type of disturbance model, a Surface-at-Risk, and provide an online procedure to calculate such a surface using the method detailed in contribution (1). By augmenting the model for a drone online, we leverage existing input-to-state-stable control-Lyapunov results to demonstrate the ability of this procedure to robustify a controller mid-flight with limited system data.

4. We provide theoretical guarantees on the provable sub-optimality of percentile-based optimization procedures on producing input sequences for general,

finite-time optimal control problems.

5. We provide an algorithm for determining the probability with which a black-box controller is successively feasible on existing system hardware.

6. We provide an algorithm to determine a probabilistic upper bound on hardware-specific controller runtimes.

Chapter Structure

First, Section 6.2 details our offline model augmentation approach addressing contributions (1) and (2) detailed above. Second, Section 6.3 details the online version of the same procedure, referencing contribution (3) in the prior section. Third and finally, Section 6.4 details the application of percentile approaches to facilitate "optimal" input selection and guarantee generation for finite-time optimal controllers. This references contributions (4)-(6) as mentioned in the prior section.

6.2 Offline Model Augmentation via Sim2Real Gap Calculation

Our method for sim2real gap quantification will express the identification of such a gap as a convex optimization problem with (perhaps) infinite constraints, and we will take a percentile approach to solve this problem. This approach will yield a robust result—a large enough sim2real gap—that holds with some minimum probability.

Defining the Gap

First, we denote our true system via x and our nominal model via $\hat{x}$, *i.e.* $\forall\ k, j = 0, 1, 2, \ldots,$

$$\begin{aligned} &\textbf{True:} \quad x_{k+1} = f(x_k, u_k),\ x_{k,k+1} \in \mathcal{X},\ u_k \in \mathcal{U}, \\ &\textbf{Sim:} \quad \hat{x}_{j+1} = \hat{f}(\hat{x}_j, \hat{u}_j),\ \hat{x}_{j,j+1} \in \hat{\mathcal{X}},\ \hat{u}_j \in \hat{\mathcal{U}}. \end{aligned} \tag{SYS}$$

As an example consistent with the demonstrations to follow, the true system could be a quadruped with the representative nominal model a unicycle system.

To provide a method of comparing the evolution of the two systems in (SYS), we will define two maps—M_x which projects the true system state x to the model state $\hat{x}$, and M_u which extends the model input $\hat{u}$ to the true system input u:

$$M_x : \mathcal{X} \to \hat{\mathcal{X}}, \quad M_u : \hat{\mathcal{U}} \times \mathcal{X} \to \mathcal{U}. \tag{MAPS}$$

These maps in (MAPS) let us formalize the gap we aim to identify between the two systems. First, we assume that we can command an input u to the true system by prescribing an input $\hat{u}$ that we would provide to our associated model. Then, we

define via O an observation function which measures the projected true system state $M_x(x_k)$ at some time-step K, *i.e.*, $\forall\ k = 0, 1, \ldots, K$,

$$x_{k+1} = f(x_k, M_u(\hat{u}, x_k)),\ O(x_0, \hat{u}) = M_x(x_K). \tag{OBS}$$

To put all these maps in the context of the quadruped/unicycle model example, the underlying control loop operates at 1 kHz and provides a natural discrete abstraction at that time step. Since, we desire our unicycle model to update at 10 Hz, $K = 100$, and we observe projected true system evolution every 10 Hz. M_x is just the projection of the quadruped state to its unicycle components. Likewise, M_u is the underlying control loop that runs at 1 kHz to realize the commanded forward walking speed and rotation. While these maps seem abstract, we will provide examples in a section to follow.

This observation map O in (OBS) permits us to quantify a discrepancy between model evolution and observed true system evolution. However, we only want to make this comparison when the projection of the initial state $M_x(x_0) \in \hat{\mathcal{X}}$—as otherwise, the projected initial state is not addressed by our representative model. This results in the following space definition and problem statement:

$$\Pi(M_x) = \left\{x \in \mathcal{X} \mid M_x(x) \in \hat{\mathcal{X}}\right\} \tag{6.1}$$

Definition 24. Let O be as defined in (OBS), $\hat{f}, \mathcal{X}, \hat{\mathcal{U}}$ be as defined in (SYS), and $\Pi(M_x)$ be as defined in (6.1). The *sim2real gap* $\Lambda \in \mathbb{R}$ is such that,

$$\Lambda = \sup_{(x_0,\hat{u})\in\Pi(M_x)\times\hat{\mathcal{U}}} \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right)\right\|. \tag{6.2}$$

Quantifying the Gap

We express identification of the sim2real gap Λ in (6.2) as an optimization problem:

$$\begin{aligned} \Lambda = \underset{r\in\mathbb{R}}{\operatorname{argmin}} \quad & r, \\ \text{subject to} \quad & r \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right)\right\|, \\ & \qquad \ldots \forall\ (x_0, \hat{u}) \in \Pi(M_x) \times \hat{\mathcal{U}} \end{aligned} \tag{6.3}$$

This problem is (likely) impossible to solve as posed, as we have no knowledge of all the constraints. So, taking inspiration from the prior chapters, we aim instead to find a "good" solution to (6.3) via a percentile approach. Such a solution will

only hold with some minimum probability $\epsilon \in [0, 1)$. Although, we can make ϵ arbitrarily close to 1 with enough samples.

Description of our Approach: Our approach hinges on the ability to independently draw state and input samples from a static distribution π over the combined state and input spaces $\mathcal{X} \times \hat{\mathcal{U}}$. In the experimental demonstrations to follow, we will argue and show evidence that our chosen method produces independent samples from an unknown distribution π. However, the specifics of crafting such a distribution will likely be different for different system/model pairs — this is where we anticipate a large portion of future work in this vein to lie. So, for the moment, we will simply assume the existence of such a distribution π formalized as follows:

Definition 25. The *comparison distribution* π maps subsets of the combined state and model input space $\mathcal{X} \times \hat{\mathcal{U}}$—as defined in (SYS)—to $[0, 1]$ *i.e.* $\forall\ A \subseteq \mathcal{X} \times \hat{\mathcal{U}}$, $\pi(A) \in [0, 1]$. Furthermore, π "covers" the space of states and inputs generating constraints for (6.3), *i.e.*, $\pi\left(\Pi(M_x) \times \hat{\mathcal{U}}\right) = 1$.

Using this comparison distribution π, we can construct a scenario program for sets of N samples $\left\{\left(x_0^l, \hat{u}^l\right)\right\}_{l=1}^N$ of state and input pairs $(x_0, \hat{u})$ distributed by π:

$$\begin{aligned} \Lambda_N^* = \underset{r \in \mathbb{R}}{\operatorname{argmin}} \quad & r, \\ \text{subject to} \quad & r \geq \left\| O(x_0^j, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}^j\right) \right\|, \\ & \ldots \forall\ (x_0^j, \hat{u}^j) \in \left\{\left(x_0^l, \hat{u}^l\right)\right\}_{l=1}^N. \end{aligned} \tag{6.4}$$

The resulting solution Λ_N^* approaches the true sim2real gap Λ as the number of samples N increases, as stated in the following theorem.

Theorem 13. *Let* Λ_N^* *be the solution to* (6.4) *with* O *as defined in* (OBS)*,* $\hat{f}$ *as defined in* (SYS)*, and* π *as defined in Definition 25. Then,* $\forall\ \epsilon \in [0, 1]$

$$S_1 \triangleq \mathbb{P}_\pi\left[\Lambda_N^* \geq \left\| O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right) \right\|\right],$$
$$\mathbb{P}_\pi^N\left[S_1 \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^N.$$

In other words, Λ_N^* *is larger than the sim2real gap for any sampled state and input pair* $(x_0, \hat{u})$ *from* π *with minimum probability* $1 - \epsilon$ *and confidence* $1 - (1 - \epsilon)^N$.

Proof: This proof follows directly from Theorem 4, noting that the violation probability $V(\Lambda_N^*) = 1 - S_1$. ∎

In other words, Theorem 13 states that the solution Λ_N^* to (6.4) is a decent approximation of the sim2real gap Λ.

Verifying against Uncertain Models

Now, we can leverage this approximate sim2real gap Λ_N^* to facilitate controller synthesis and verification in simulation. This section details those efforts.

Constructing a Valid Uncertain Model: To start, we can use the resulting probabilistic sim2real gap Λ_N^* from (6.4) to augment our nominal model $\hat{x}$ in (SYS) and define an uncertain system denoted via $\tilde{x}$. Specifically, we will first define a feasible space of disturbances, with 2-norm $\|\cdot\|$:

$$D = \{d \in \mathbb{R}^n \mid \|d\| \leq \Lambda_N^* \text{ (as per (6.4))}\}, \tag{6.5}$$

and use D to define our uncertain system:

$$\tilde{x}_{j+1} = \hat{f}(\tilde{x}_j, \hat{u}_j) + d_j,\ d_j \sim \mathrm{U}[D], \tag{6.6}$$

with $\mathrm{U}[D]$ the uniform distribution over D. If we define a one-step reachable space provided a model state and input $(\hat{x}, \hat{u}) \in \hat{\mathcal{X}} \times \hat{\mathcal{U}}$,

$$\mathcal{R}(\hat{x}, \hat{u}) = \left\{\hat{f}(\hat{x}, \hat{u}) + d,\ \forall\ d \in D\right\}, \tag{6.7}$$

then we have the following result regarding the evolution of the true system and this reachable space.

Corollary 19. *Let O be as defined in* (OBS)*, $\mathcal{R}$ be as defined in* (6.7)*, M_x be as defined in* (MAPS)*, π be as per Definition 25, and Λ_N^* be as defined in* (6.4)*. Then,* $\forall\ \epsilon \in [0, 1]$*,*

$$S_2 \triangleq \mathbb{P}_\pi\left[O(x_0, \hat{u}) \in \mathcal{R}(M_x(x_0), \hat{u})\right],$$
$$\mathbb{P}_\pi^N\left[S_2 \geq 1 - \epsilon\right] \geq 1 - (1 - \epsilon)^N.$$

In other words, the probability that the observed evolution of the true system $O(x_0, \hat{u})$ lies in the reachable space of our uncertain model $\mathcal{R}(M_x(x_0), \hat{u})$ is at least $1 - \epsilon$ with confidence $1 - (1 - \epsilon)^N$.

Proof: This is a direct application of Theorem 13, as for any state and input pair $(x_0, \hat{u}) \in \mathcal{X} \times \hat{\mathcal{U}}$, $\Lambda_N^* \geq \left\|O(x_0, \hat{u}) - \hat{f}\left(M_x(x_0), \hat{u}\right)\right\|$ iff $O(x_0, \hat{u}) \in \mathcal{R}(M_x(x_0), \hat{u})$, by definition of $\mathcal{R}$ in (6.7) and D in (6.5). ■

In other words, Corollary 19 tells us that even though a single step of our uncertain model (6.6) may not be an accurate representation of our true system's evolution, the space of all possible single-step evolutions does, to high probability, contain the evolution of our true system at the next time-step.

For controller synthesis and verification then, Corollary 19 tells us that our uncertain model is a decent approximator of true system behavior. Therefore, any controller that exhibits good performance on the uncertain model, should likewise exhibit good performance on the true system. Quantification of "good" performance and verifying a controller's ability to realize "good" performance is the subject of the next subsection, which follows from the risk-aware probabilistic verification procedure detailed in Section 5.2 in Chapter 5.

Verifying against Safety Metrics: First, provided a parameterized controller $\hat{\mathrm{U}} : \hat{\mathcal{X}} \times \Theta \to \hat{\mathcal{U}}$ with parameter $\theta \in \Theta$ and noise sequence ξ where $\xi_j = d \sim \mathrm{U}[D]$, we define ϕ to be the closed-loop trajectory to our uncertain model (6.6), *i.e.* with $J > 0$ and $\tilde{x}_0 = \hat{x}_0$,

$$\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta, J) = \tilde{x}_J,\ \tilde{x}_{j+1} = \hat{f}\left(\tilde{x}_j, \hat{\mathrm{U}}(\tilde{x}_j, \theta)\right) + \xi_j. \tag{6.8}$$

Second, inspired by traditional safety measures, *e.g.* barrier functions over the system state, we define safety metrics h to be functions over system trajectories that only output positive numbers for trajectories exhibiting the desired safe behavior:

$$\mathrm{h}\left(\phi^{\hat{\mathrm{U}}}\left(\hat{x}_0, \xi, \theta\right)\right) \geq 0 \iff \begin{array}{l}\phi^{\hat{\mathrm{U}}}\left(\hat{x}_0, \xi, \theta\right) \text{ exhibits} \\ \text{desired safe behavior.}\end{array} \tag{6.9}$$

Examples of safety metrics include robustness measures from Signal Temporal Logic [19] or the minimum value of a barrier function over a finite-time horizon [145].

Ideally, we would like for our controller $\hat{\mathrm{U}}$ to only ever realize trajectories $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ with a positive evaluation under this safety metric h. To check for this positivity, we will implement a value-at-risk approach to robustness estimation, as detailed in Section 5.2. Specifically, we will first draw $(\hat{x}_0, \theta)$ uniformly from $\hat{\mathcal{X}} \times \Theta$. Then, we will evaluate the safety of one trajectory emanating from that initial condition $\hat{x}_0$ with that parameter θ, *i.e.* record $s = \mathrm{h}\left(\phi^{\hat{\mathcal{U}}}\left(\hat{x}_0, \xi, \theta\right)\right)$ for some valid noise-sequence ξ[1]. Repeating this procedure N times to take N such samples s_i and create the dataset $\{s_i\}_{i=1}^N$, we then define $s_N^* = \min\{s_i\}_{i=1}^N$. Then, via Theorem 6, we have the following result:

Corollary 20. *Let the uncertain system trajectory* $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ *be as defined in* (6.8)*, the safety metric* h *be as defined in* (6.9)*,* $\{s_i\}_{i=1}^N$ *be the safety values of N sampled*

[1]Note that according to the general risk-aware pipeline presented in Section 5.2, taking samples s in this fashion corresponds to taking a sample of the holistic system robustness measure R as per Definition 22.

trajectories $\phi^{\hat{\mathrm{U}}}(\hat{x}_0, \xi, \theta)$ *with initial conditions and parameters* (x_0, θ) *drawn from* $\mathrm{U}[\hat{\mathcal{X}} \times \Theta]$, *and* $s_N^* = \min\{s_i\}_{i=1}^N$. *Then,* $\forall\, \epsilon \in [0, 1]$ *and abbreviating* $\mathrm{U}[\hat{\mathcal{X}} \times \Theta] = \pi_0$,

$$S_3 \triangleq \mathbb{P}_{\pi_0,\, \xi_j \sim \mathrm{U}[D]\ \forall\ j=1,2,\ldots} \left[s \geq s_N^* \right],$$
$$\mathbb{P}^N_{\pi_0,\, \xi_j \sim \mathrm{U}[D]\ \forall\ j=1,2,\ldots} \left[S_3 \geq 1 - \epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

In other words, the probability that s_N^* *will be smaller than any sample-able safety value* s *is at minimum* $1 - \epsilon$ *with confidence* $1 - (1 - \epsilon)^N$.

Proof: This is an application of Theorem 6. ∎

This completes the theoretical statement of our proposed pipeline for safety-critical controller verification via sim2real gap quantification. For the intermediate synthesis step, one can use any method they like, *e.g.* robust control methods [178], [179], input-to-state-stable Lyapunov or barrier functions [146], [164], [180], [181], *etc*. Practitioners could even parameterize controllers and implement the risk-aware synthesis step outlined in Section 5.3. The emphasis here is on verifying the resulting controller against the uncertain model and using Corollary 20 to discriminate between better controllers (those with a higher minimum probability of realizing safe trajectories) and worse controllers (those with a lower minimum probability). By Theorem 13 and Corollary 20 we know that our uncertain model is a decent representor of system behavior. Therefore, those controllers with a high probability of exhibiting desired safe behavior in the uncertain simulator should likely have a high probability of exhibiting safe behavior on hardware. We will demonstrate this procedure on two hardware platforms: the Robotarium [144] and a quadruped.

Experimental Demonstrations

We will describe our procedure's implementation on both systems simultaneously, as we aim to represent both systems with the same model abstraction, a unicycle:

$$x_{k+1} = f(x_k, u_k),\ x_{k,k+1} \in \mathcal{X},\ u_k \in \mathcal{U},\ t_{k+1} - t_k = \Delta t$$
$$\hat{x}_{j+1} = \hat{x}_j + \Delta\hat{t} \begin{bmatrix} \cos(\hat{x}[3]) & 0 \\ \sin(\hat{x}[3]) & 0 \\ 0 & 1 \end{bmatrix} \hat{u}_j,\ \hat{x}_{j,j+1} \in \hat{\mathcal{X}},\ \hat{u}_j \in \hat{\mathcal{U}}.$$

The parameters for each system are as follows (Robotarium (R) and Quadruped (Q)):

(R) $\hat{\mathcal{X}} = [-1.6 \times 1.6] \times [-1, 1] \times [0, 2\pi]$; $\hat{\mathcal{U}} = [-0.2, 0.2] \times [-\pi, \pi]$; $\Delta t, \Delta\hat{t} = 0.033$.

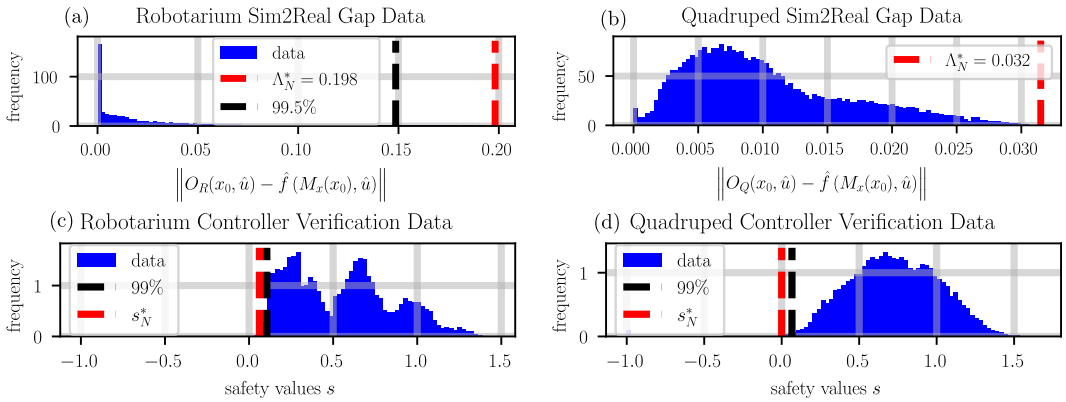

Figure 6.1: All data for our experimental pipeline. (Top Left) Data for calculation of a probabilistic sim2real gap Λ_N^* for the Robotarium. Per Theorem 13, we expect that after observing 600 randomly sampled errors, our reported sim2real gap $\Lambda_N^* = 0.198$ is greater than any sample-able gap with minimum probability 99.5% with 95% confidence. Comparing Λ_N^* to the true cutoff after taking 2400 samples verifies this inequality and supports Theorem 13. (Top Right) Data for sim2real gap calculation for the quadruped. True cutoffs are not shown as we did not exhaustively sample gaps. (Bottom) Verification data for both controllers against their respective uncertain models. Note that in both cases, we sampled 300 trajectories to calculate a minimum safety value s_N^* which, according to Corollary 20, should be less than any sample-able safety value with minimum probability 99% with 95% confidence. Taking 20000 safety samples and calculating the true cutoffs against the sampled data shows that this inequality holds verifying Corollary 20.

(Q) $\hat{\mathcal{X}} = [-2.5, 2.5]^2 \times [0, 2\pi]$; $\hat{\mathcal{U}} = [-0.15, 0.15] \times [-0.3, 0.3]$; $\Delta t = 0.001$, and $\Delta \hat{t} = 0.1$.

For both systems, we can read true state data x to recover the idealized unicycle state $\hat{x}$. Therefore, M_x is just a projection for both systems. The Robotarium permits unicycle-like commands to their agents resulting in $M_u = I_{2\times 2}$. On the other hand, the quadruped has a lower-level walking controller that operates at 1 kHz to realize commanded forward walking and yaw angular velocities [182]. As a result, M_u for the quadruped is this pre-built walking controller. Finally, our observation maps:

(R) $O_R(x_0, \hat{u}) = M_x(x_1)$ - read the projected true state after one time-step, and,

(Q) $O_Q(x_0, \hat{u}) = M_x(x_{100})$ - read the projected true state after 100 time-steps.

Sampling from the Comparison Distribution: To calculate the discrepancy between the system and our chosen model, we will sample from the comparison distribution π (Definition 25) with the steps listed below:

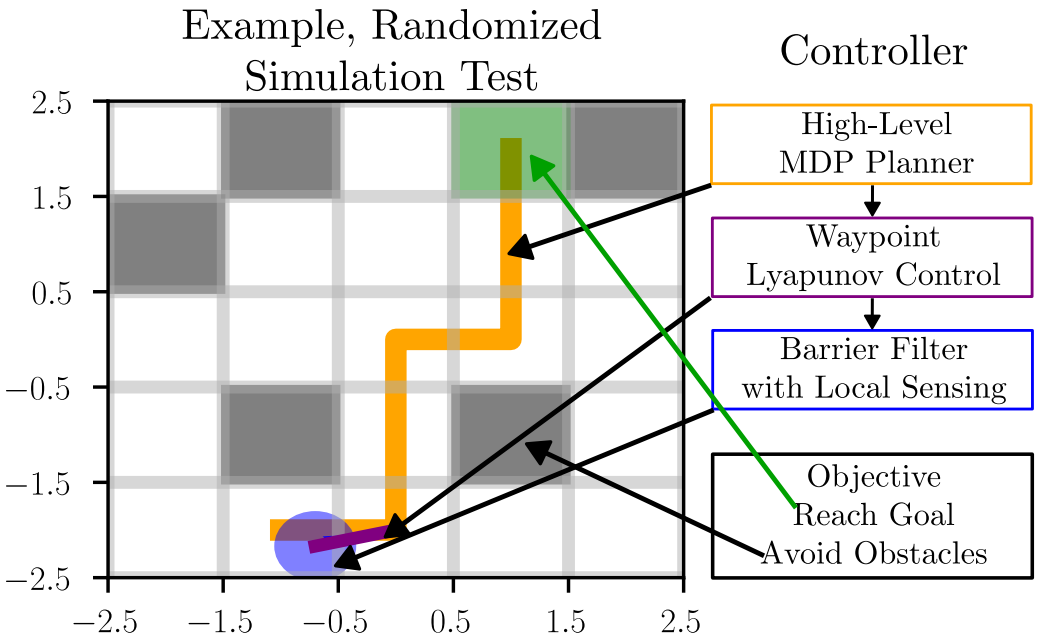

Figure 6.2: As part of our risk-aware synthesis pipeline, we verify controllers against an uncertain model. Shown above is an example randomized test scenario for controller development. The test scheme remains the same for the two different systems (Robotarium and quadruped) as their controller objectives are similar.

(x_0) We uniformly randomly sample a planar position x from $\hat{\mathcal{X}}$. Then, we send both agents to the waypoint x using a Lyapunov controller built on top of the input maps M_u for both systems. Once the system reaches a ball of 0.1 m around the desired waypoint, we stop and record the resulting state as x_0.

($\hat{u}$) We uniformly randomly sample an input $\hat{u}$ from $\hat{\mathcal{U}}$. Then, we command the system with this input for 50 true-system time-steps for the Robotarium and 1000 true-system time-steps for the quadruped. We command this input for an extended period to approximate the randomized initial location that we had before sampling x_0 so that subsequent samples drawn from this procedure are drawn effectively independently.

Probabilistic Sim2Real Gap Calculation: For the Robotarium, we collected 2400 observations and used the first 600 to calculate constraints for (6.4). This provided a sim2real gap constant $\Lambda_N^* = 0.198$ which, according to Theorem 13, is greater than any sampled sim2real gap with minimum probability 99.5% with minimum confidence 95%. Figure 6.1 (a) shows the probabilistic sim2real gap Λ_N^* overlaid on a histogram of sampled sim2real gaps to approximate the underlying distribution. Notice that since Λ_N^* is greater than the 99.5% cutoff, this corroborates Theorem 13.

Hardware Demonstrations of Controllers Synthesized and Verified only in Simulation – All Successes

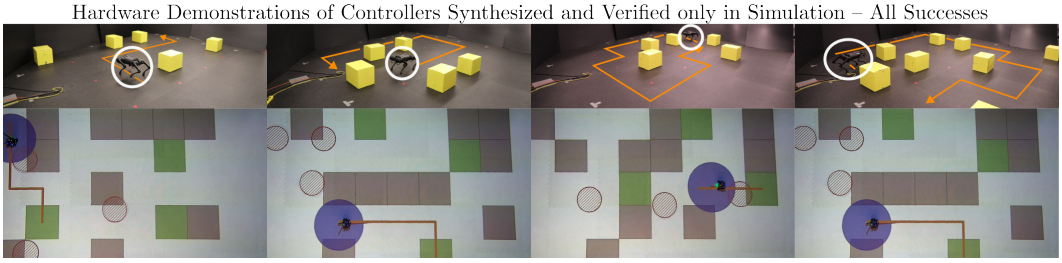

Figure 6.3: Since we verified our controllers against the uncertain model produced by our procedure, we expect that the closed-loop hardware systems should realize similar, satisfactory behavior. Indeed, for the first 10 runs on the quadruped and the first 40 runs on the Robotarium, the agents were able to avoid static/moving obstacles and navigate to their goals successfully, despite a wide variety of randomized test scenarios. The first four runs for both systems are depicted above. This ability to synthesize and verify controllers in simulation, with confidence that similar behaviors will manifest in the true system without requiring additional testing, is the main benefit of our proposed approach. Paths for all tests are shown in orange, and the quadruped is highlighted in white. The multi-level control architecture is depicted in Figure 6.2.

Figure 6.1 (b) shows similar results for the quadruped after taking 100 observations. The true cutoff value is not shown as we did not exhaustively sample observations to approximate the underlying distribution. Although, with Theorem 13 and the Robotarium results, we are confident that the calculated sim2real gap is greater than any sample-able gap with minimum probability 97% and with confidence 95%.

Safety-Critical Controller Verification: Figure 6.2 shows the general controller architecture for which we aim to identify parameters such that the resulting controller $\hat{U}$ has a high probability of rendering satisfactory behavior on the uncertain model. To synthesize such a controller, we use a safety metric h as per (6.9) that outputs -1 if the agent crashes into either a static or moving obstacle and outputs the Manhattan distance traveled along the shortest feasible path to a goal—the orange line in Figure 6.2—if it successfully avoids crashes within 200 time-steps. We do not formally define this metric as it is not central to the paper's concept.

For both systems, we iterated through at least 10 different sets of controllers $\hat{\mathrm{U}}$ with parameter spaces, Θ defined as follows:

(R) All possible setups of 10 static obstacles, 3 goals, and 1 initial condition cell in an 8×5 grid such that a feasible path exists between the initial cell and at least one goal. This is in addition to the starting locations and movement

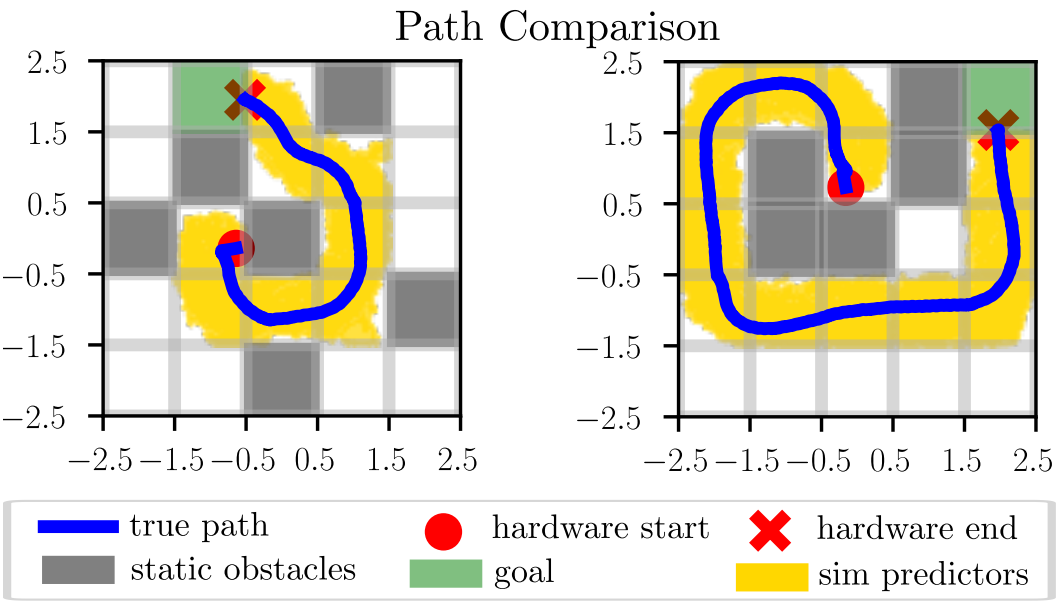

Figure 6.4: Our procedure detailed in Section 6.2 increases the confidence that those controllers that pass the verification step will exhibit similar performance on hardware as they did in simulation, even if we did not directly verify the controller on hardware. This increased confidence arises through our verification of the uncertain model, whose reachable set we prove encapsulates true system evolution to high probability. This can be seen in the figures above, as the quadruped's evolution (blue) lies within its associated uncertain simulator's predictions (gold).

directions of 3, uncontrolled moving obstacles.

(Q) All possible setups of 5 static obstacles, 1 goal, and 1 initial condition cell in a 5×5 grid such that a feasible path exists between the initial cell and the goal.

Once we found controllers that, according to Corollary 20, had a minimum satisfaction probability of at least 99% with 95% confidence, we implemented these controllers $\hat{\mathrm{U}}_R, \hat{\mathrm{U}}_Q$ on their respective systems, the Robotarium and the quadruped. In each verification step, we evaluated the controllers under 500 randomized scenarios—500 test scenarios we would have otherwise had to run on real systems were we not using our uncertain model to approximate true-system behavior. Figures 6.1 (c) and (d) show the verification data for the finalized controllers $\hat{\mathrm{U}}_R, \hat{\mathrm{U}}_Q$, respectively, with the distribution approximated by evaluating 20000 randomized scenarios.

According to our pipeline then, for the fact that we verified our controllers—Corollary 20—against an uncertain model which has a high probability of rep-

resenting true system behavior—Theorem 13 and Corollary 19—these controllers should similarly exhibit satisfactory behavior on their true systems when implemented. Figure 6.3 depicts the first four tests undergone by both systems—they completed their tasks in these tests as expected. We ran 40 more randomized tests for the Robotarium, drawing from the same parameter space Θ as described earlier—all successes. Similarly, we ran 10 more tests for the Quadruped drawing from its respective parameter space Θ—all successes. We expected this level of performance since the controllers exhibited a high probability of realizing desired safe behaviors on their respective uncertain models that encapsulated true system behavior. Additionally, we can see that encapsulation of true system behavior in Figure 6.4, as the quadruped's trajectory (blue) lies within the gold simulation prediction regions. Furthermore, we only had to run tests on hardware to calculate the sim2real gap and did not have to run any more tests to make these statements on system performance.

6.3 Online Disturbance Model Learning

Whereas the prior section detailed an offline method for uncertain model generation and controller synthesis, this section offers an online method to learn a disturbance model during operation, that can be leveraged to robustify controllers using existing input-to-state-stable and input-to-state-safe techniques. Succinctly, we utilize a scenario approach with a bounded variance assumption to identify points on a *Surface-at-Risk* — a concept we will define. Then, we fit, via Gaussian Process Regression (GPR), a model of the Surface-at-Risk based on these sampled points. As such, we will first briefly overview Gaussian Process Regression.

A Brief Overview of Gaussian Process Regression

Let $f : X \to \mathbb{R}$ be an unknown function that we aim to represent by taking noisy samples y of f at points $x \in X$, where the noise ξ is assumed to be sub-Gaussian [183]–[185]. Let $\mathbb{X} = \{x_i\}_{i=1}^N$ be a set of N points $x \in X$ and $\mathbb{Y}$ be the corresponding set of noisy observations, *i.e.* $\mathbb{Y} = \{y_i = f(x_i) + \xi,\ \forall\ x_i \in \mathbb{X}\}$. Furthermore, let $k : X \times X \to \mathbb{R}$ be a positive-definite *kernel function*. Then, a *gaussian process* is uniquely defined by its mean function $\mu : X \to \mathbb{R}$ and its variance function $\sigma : X \to \mathbb{R}$. These functions are defined as follows, with $k_N(x) = [k(x, x_i)]_{x_i \in \mathbb{X}}$, $\mathbb{K} = [k(x_i, x_j)]_{x_i, x_j \in \mathbb{X}}$, $y_{1:N} = [y_i]_{y_i \in \mathbb{Y}}$, and $\lambda = (1 + \frac{2}{N})$ [184]:

$$\mu_N(x) = k_N(x)^T \left(\mathbb{K} + \lambda I_N\right)^{-1} y_{1:N}, \quad \sigma_N(x) = k_N(x, x), \tag{6.10}$$
$$k_N(x, x') = k(x, x') - k_N(x)^T \left(\mathbb{K}_N + \lambda I\right)^{-1} k_N(x').$$

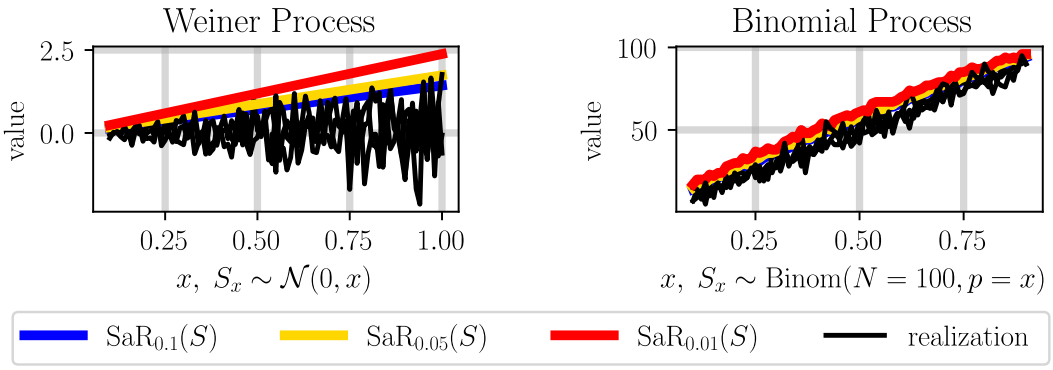

Figure 6.5: Example Surfaces-at-Risk at risk-levels $\epsilon \in [0.1, 0.05, 0.01]$ for a Weiner Process (Left) and Binomial Process (Right). Distributions for the indexed scalar random variables S_x comprising each process S are provided on the axes. Sample realizations of the stochastic processes are shown in black, with Surfaces-at-Risk shown via colored lines.

Lastly, each kernel function has a space of functions it can reproduce to point-wise accuracy, it's Reproducing Kernel Hilbert Space (RKHS). Under the assumption that the function to-be-fitted f has bounded norm in the RKHS of the chosen kernel k, GPR guarantees high-probability representation of f as formalized in the theorem below, taken from [184]:

Theorem 14. *Let $f : X \to \mathbb{R}$, $\mathbb{X} = \{x_i\}_{i=1}^N$ be a set of N points $x \in X$, $\mathbb{Y} = \{y_i = f(x_i) + \xi\}_{x_i \in \mathbb{X}}$ be a set of noisy observations y_i of $f(x_i)$ with R sub-gaussian noise ξ, and $k : X \times X \to \mathbb{R}$ be a positive-definite kernel function. If f has B-bounded RKHS norm for some $B > 0$, i.e. $\|f\|_{RKHS} \leq B$, then, with μ_N and σ_N as per* (6.10) *and with minimum probability $1 - \delta$,*

$$|\mu_N(x) - f(x)| \leq \left(B + R\sqrt{2 \ln \frac{\sqrt{\det\left((1 + \frac{2}{N})I_N + \mathbb{K}_N\right)}}{\delta}} \right) \sigma_N(x),\ \forall\, x \in X.$$

Surfaces-at-Risk for Scalar Stochastic Processes

This section formally defines a *Surface-at-Risk* for a scalar stochastic process — the specific structure we aim to fit via GPR. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with Ω a sample space, $\mathcal{F}$ a σ-algebra over Ω defining events, and $\mathbb{P}$ a probability measure, we define a scalar *stochastic process* S over the indexed space $\mathcal{X}$ as a collection of scalar random variables $S_x : \Omega \to \mathbb{R}$, *i.e.* $S = \{S_x\}_{x \in \mathcal{X}}$. Here, each scalar

random variable S_x has a (perhaps) different distribution $\pi_x : \mathbb{R} \to [0, 1]$ such that probability of S_x taking values in $A \subseteq \mathbb{R}$, *i.e.* $\mathbb{P}_{\pi_x}[S_x \in A \subseteq \mathbb{R}]$, is well-defined.

Then, the *Surface-at-Risk* for a scalar stochastic process is a similar collection of the Values-at-Risk of the underlying scalar random variables constituting the scalar stochastic process. Note that Value-at-Risk was defined in a prior chapter (see Definition 15 in Chapter 4).

Definition 26. The *Surface-at-Risk* level $\epsilon \in [0, 1]$ of a scalar stochastic process S indexed by the set $\mathcal{X}$ is the indexed collection of the Values-at-Risk level ϵ of each random variables S_x comprising S (Value-at-Risk is defined in Definition 15):

$$\mathrm{SaR}_\epsilon(S, x) = \mathrm{VaR}_\epsilon(S_x).$$

Figure 6.5 shows a few examples of Surfaces-at-Risk for varying risk-levels ϵ overlaid on realizations of common stochastic processes.

The Risk-Aware Disturbance-Norm Identification Problem

From a risk-aware standpoint, we aim to identify a *Surface-at-Risk* as per Definition 26 for a scalar stochastic process S indexed over the model state-space $\hat{\mathcal{X}}$. Sample realizations of this process correspond to disturbance norms the system might experience at any given model state $\hat{x} \in \hat{\mathcal{X}}$. To formally state this problem, we will first denote our true system via x and sim model via $\hat{x}$, *i.e.* $\forall\ k, j = 0, 1, 2, \ldots$, (perhaps) different state and input spaces, and process noise ξ with (unknown and perhaps) state-dependent distribution π. Note that this is the same true and model system discrepancy studied in the offline version of this problem in Section 6.2:

$$\begin{aligned} &\textbf{True:} \quad x_{k+1} = f(x_k, u_k) + \xi, \quad x_k \in \mathcal{X},\ u_k \in \mathcal{U},\ \xi \sim \pi, \\ &\textbf{Sim:} \quad \hat{x}_{j+1} = \hat{f}(\hat{x}_j, \hat{u}_j), \qquad\qquad\quad \hat{x}_j \in \hat{\mathcal{X}},\ \hat{u}_j \in \hat{\mathcal{U}}. \end{aligned} \tag{SYS}$$

As an example consistent with the demonstration to follow, the true system would be a drone, with our reduced-order simulator model a single integrator. The true state would be the drone's position and orientation, and the true input would be the rotor torques. Meanwhile, the model state would be the drone's position in 3-space, and the model input would be the desired velocity.

To identify the discrepancy between the systems in (SYS), we define two maps: M_x, which projects the true state x to the model state $\hat{x}$, and M_u, which extends the model input $\hat{u}$ to the true input u, *e.g.* M_u provides rotor torques to realize the desired

velocity in 3-space. Note, these are the same maps as defined in the prior section:

$$M_x : \mathcal{X} \to \hat{\mathcal{X}}, \quad M_u : \hat{\mathcal{U}} \times \mathcal{X} \to \mathcal{U}. \tag{MAPS}$$

To note, we only assume the existence of these maps and the ability to use them, we do not assume that they are unique, we know their analytic form, *etc*. To put these maps in the context of our drone example, the drone's underlying controller operates at 1 kHz making the true-system time step 1 ms. Since we aim to provide model inputs at 50 Hz, $K = 20$. M_x is just the projection of our drone's position in 3-space, and M_u is the on-board controller that takes in a commanded 3-space velocity — model input $\hat{u}$ — and updates rotor speeds at 1 kHz to achieve that velocity. These maps will be further explained in an example section to follow. Finally, we assume that after some amount of true system time-steps $K > 0$, we can observe projected true system evolution. We denote K as the *time-dilation parameter* and the observation function O is defined as follows:

$$x_{k+1} = f(x_k, M_u(\hat{u}, x_k)), \; O(x_0, \hat{u}) = M_x(x_K). \tag{OBS}$$

These maps let us define the projected evolution of our true system, *i.e.* evolution of $\hat{x}_j = M_x(x_{Kj})$, when driven by a feedback controller $U : \hat{\mathcal{X}} \to \hat{\mathcal{U}}$. Comparing projected and sim model evolution results in the discrepancy d we aim to learn:

$$\begin{aligned} \hat{x}_{j+1} &= \hat{f}(M_x(x_{Kj}), U(M_x(x_{Kj})) + d, \\ d &= O(x_{Kj}, U(M_x(x_{Kj})) - \hat{f}(M_x(x_{Kj}), U(M_x(x_{Kj})), \\ \delta &= \|d\|, \text{ and is a random sample with distribution } \pi_{\hat{x}} : \mathbb{R} \to [0, 1]. \end{aligned} \tag{6.11}$$

Then, inspired by the input-to-state-safe barrier and input-to-state-stable Lyapunov works whose robust controllers only require information on the 2-norm of this disturbance d, we aim to learn a probabilistic upper bound on $\|d\|$ by taking samples of indexed random variables $S_{\hat{x}}$ comprising a disturbance-norm stochastic process S indexed by $\hat{\mathcal{X}}$ as in (SYS).

Definition 27. The *disturbance-norm stochastic process* $S = \{S_{\hat{x}}\}_{\hat{x} \in \hat{\mathcal{X}}}$ where samples of each random variable $S_{\hat{x}}$ correspond to norms δ of disturbances d as defined in equation (6.11). The variability in norm samples δ arises through the assumed process noise ξ in the true system dynamics in (SYS).

Remark on Residuals: If we only consider a deterministic discrepancy between the true and sim models, then the disturbances d as per (6.11) would correspond to

residual dynamics, and our procedure would fit a surface to the norm of the residual dynamics (learning residual dynamics has a well-studied history, see [186]–[189] and citations within). The discrepancy between these approaches and ours is that we also learn a probabilistic bound on the norm of any stochastic, model-state-dependent disturbances that affect the system during operation. This is why we represent the discrepancies as a stochastic process and fit a Surface-at-Risk, which provides a natural way to reason about risk-aware disturbance rejection in a context including model errors and stochastic uncertainty.

Furthermore, we assume our disturbance-norm stochastic process is indexed over the model state space $\hat{\mathcal{X}}$ as opposed to the true state space $\mathcal{X}$ as we only assume the ability to measure the projected state $\hat{x}_j = M_x(x_{Kj})$. Therefore, we can only correspond sampled disturbance norms δ to points in the projected state space $\hat{\mathcal{X}}$. Then, our goal is to identify a "close" upper bound to the *Surface-at-Risk* for this disturbance-norm stochastic process at some risk-level $\epsilon \in [0, 1]$.

Problem 6. *Identify an upper bound to the Surface-at-Risk at some risk-level* $\epsilon \in [0, 1]$ *for the disturbance-norm stochastic process S as per Definition 27 with Surfaces-at-Risk as defined in Definition 26. Specifically, identify an estimate* $\mathbb{SR}_\epsilon$ *such that,*

$$\mathbb{SR}_\epsilon(S, \hat{x}) \geq \mathrm{SaR}_\epsilon(S, \hat{x}),\ \forall\, \hat{x} \in \hat{\mathcal{X}}. \tag{6.12}$$

While the aforementioned upper bound $\mathbb{SR}_\epsilon$ could be arbitrarily large and satisfy (6.12), we aim to find a "close" upper bound to the true Surface-at-Risk level ϵ to facilitate risk-aware control.

Fitting a Disturbance-Norm Surface-at-Risk

For identifying such an upper bound $\mathbb{SR}_\epsilon$, we first note that even for stochastic processes whose sample realizations are non-differentiable, their Surfaces-at-Risk are relatively smooth — see Figure 6.5 for examples. Intuitively, we expect the disturbance norms δ_i, δ_j at "close" model states $\hat{x}_i, \hat{x}_j \in \hat{\mathcal{X}}$ are similarly "close":

Assumption 13. For the disturbance-norm stochastic process S in Definition 27, the *Surface-at-Risk* at a given risk-level $\epsilon \in [0, 1]$ has bounded discrepancy. In other words, $\exists\ \alpha, \beta \in \mathbb{R}_{\geq 0}$ such that,

$$\forall\, \hat{x}_i, \hat{x}_j \in \hat{\mathcal{X}},\ \|\hat{x}_i - \hat{x}_j\| \leq \alpha \implies |\,\mathrm{SaR}_\epsilon(S, \hat{x}_i) - \mathrm{SaR}_\epsilon(S, \hat{x}_j)| \leq \beta.$$

Notably, this assumption only implies a bounded discrepancy, and not continuity, *e.g.* a bounded piecewise continuous function would have bounded variance as per our assumption. We will verify that this assumption holds for the data set we collect in the experimentation section to follow.

Second, we need to take (perhaps noisy) unbiased samples of $\mathbb{SR}_\epsilon(S, \hat{x})$ for a given model state $\hat{x} \in \hat{\mathcal{X}}$. By equation (6.12), $\mathbb{SR}_\epsilon(S, \hat{x}) \geq \text{VaR}_\epsilon(S_{\hat{x}})$, and we can define one sample δ_j of $S_{\hat{x}_j}$ as follows, where O is as per (OBS), and M_x is as per (MAPS):

$$\delta_j = \|O(x_{Kj}, U(M_x(x_{Kj})) - \hat{f}(M_x(x_{Kj}), U(M_x(x_{Kj}))\|, \quad \hat{x}_j = M_x(x_{Kj}). \tag{6.13}$$

Then, we can group multiple samples δ_j for sequential model states visited during operation, *i.e.* $\delta_j, \delta_{j+1}, \ldots$ for $\hat{x}_j, \hat{x}_{j+1}, \ldots$ to produce an upper bound to at least one Value-at-Risk level ϵ of a sampled random variable, *i.e.* $\text{VaR}_\epsilon(S_{\hat{x}_j}), \text{VaR}_\epsilon(S_{\hat{x}_{j+1}}), \ldots$. To do so, we require the following theorem, stated for N scalar random variables X with (perhaps) different distributions π.

Theorem 15. *Let $\{X_i\}_{i=1}^N$ be a collection of N scalar random variables with (perhaps) different distributions $\{\pi_i\}_{i=1}^N$, and let $\{x_i\}_{i=1}^N$ be a set of N samples of these random variables, one sample per each random variable, i.e. x_i is a sample of X_i. Then, for any $\epsilon \in [0, 1]$, the probability that at least one sample $x_\ell \in \{x_i\}_{i=1}^N$ is greater than the Value-at-Risk level ϵ of its corresponding random variable X_ℓ is at least $1 - (1 - \epsilon)^N$, i.e. with* VaR *as per Definition 15 and $\forall\ \epsilon \in [0, 1]$,*

$$\mathbb{P}_{\pi_1, \pi_2, \ldots, \pi_N} \left[\exists\, x_\ell \in \{x_i\}_{i=1}^N \text{ s.t. } x_\ell \geq \text{VaR}_\epsilon(X_\ell) \right] \geq 1 - (1 - \epsilon)^N.$$

Proof: Consider a random variable $X_\ell \in \{X_i\}_{i=1}^N$. The probability of taking a sample x_ℓ of X_ℓ such that $x_\ell \geq \text{VaR}_\epsilon(X_\ell)$ is less than or equal to ϵ by Definition 15. The same line of reasoning holds $\forall\ X_\ell \in \{X_i\}_{i=1}^N$. As such, the probability that no sample $x_\ell \in \{x_i\}_{i=1}^N$ is greater than the corresponding Value-at-Risk level ϵ is less than or equal to $(1 - \epsilon)^N$, yielding our result. ■

Our procedure for generating unbiased samples of the upper bound $\mathbb{SR}_\epsilon$ stems directly from Theorem 15 and Assumption 13. First, we let the system evolve for N_{RV} model time-steps and collect one norm sample δ_j per model state $\hat{x}_j$ visited during operation. This norm sample δ_j is calculated as per (6.13). Second, Theorem 15 guarantees that the largest norm sample δ_j^* is greater than the Value-at-Risk level ϵ for its corresponding indexed random variable $S_{\hat{x}_j^*}$ with some minimum probability. Third, if all norm samples were drawn from indexed random variables

Algorithm 1 Fitting a Disturbance-Norm Surface-at-Risk

Data: α, β for Assumption 13, an integer $\mathrm{N_{RV}} > 0$ for Theorem 15 corresponding to the number of random variables to sample, time-step dilation parameter $K > 0$ between true system evolution and model evolution as per (OBS), and $k : \hat{\mathcal{X}} \times \hat{\mathcal{X}} \to \mathbb{R}$ a kernel function

1 **Initialize:** $\mathrm{s} = 0, \mathbb{X} = [], \mathbb{Y} = []$

2 **References:** Disturbance Norm samples δ_j as per (6.13) and projector M_x as per (MAPS)

3 **while** *True* **do**

4 Initialize empty data-set, *i.e.* $\mathcal{D}_\mathrm{s} = [\,]$

5 **for** $j = \mathrm{N_{RV}} \cdot \mathrm{s}, \mathrm{N_{RV}} \cdot \mathrm{s}+1, \ldots, \mathrm{N_{RV}}(\mathrm{s}+1) - 1$ **do**

6 Collect state-indexed disturbance norm samples, *i.e.* $\mathcal{D}_\mathrm{s} \leftarrow \mathcal{D}_\mathrm{s} \cup (\delta_j, \hat{x}_j = M_x(x_{Kj}))$

7 **end**

8 Augment GP state dataset with $\mathcal{D}_\mathrm{s}$: $\mathbb{X} \leftarrow \mathbb{X} \cup \hat{x}_{\mathrm{N_{RV}}(\mathrm{s}+1)-1}$

9 Augment GP norm dataset with $\mathcal{D}_\mathrm{s}$: $\mathbb{Y} \leftarrow \mathbb{Y} \cup \max\{\delta_\ell \in \mathcal{D}\} + \beta$

10 Fit $\mu_\mathrm{s}, \sigma_\mathrm{s}$ as per (6.10) with data sets $\mathbb{X}, \mathbb{Y}$. s ++

11 **end**

$S_{\hat{x}_j}$ whose indices $\hat{x}_j$ were "close", *i.e.* $\|\hat{x}_s - \hat{x}_r\| \leq \alpha \;\forall\; \hat{x}_r \neq \hat{x}_s \in \{\hat{x}_{j+i}\}_{i=0}^{N-1}$ and for some $\alpha > 0$, we can use Assumption 13 to augment the largest norm sample δ_j^* by a constant $\beta > 0$. The sum is, with minimum probability $1 - (1-\epsilon)^N$, an unbiased, non-noisy sample of $\mathbb{SR}_\epsilon(S, \hat{x}_j)$. Algorithm 1 formalizes this procedure and our main theoretical result follows.

Theorem 16. *Let* $\alpha, \beta, \mathrm{N_{RV}}, \mathrm{s}, \mu_\mathrm{s}, \sigma_\mathrm{s}$, *and* k *be as defined in Algorithm 1, let* $B > 0$, *let* SaR *be the Surface-at-Risk measure as per Definition 26 for some risk-level* $\epsilon \in [0, 1]$, *let* S *be the disturbance-norm stochastic process as per Definition 27, and let Assumption 13 hold for each data set* $\mathcal{D}_\mathrm{s}$ *in lines 5-7 of Algorithm 1 with respect to the given parameters* α, β. *If* $\| \mathbb{SR}_\epsilon(S)\|_{RKHS} \leq B$, *then with minimum probability* $\left(1 - (1-\epsilon)^{\mathrm{N_{RV}}}\right)^\mathrm{s}$ *the following holds* $\forall\, \hat{x} \in \hat{\mathcal{X}}$ *and* $\forall\ \mathrm{s} = 1, 2, \ldots$:

$$|\mu_\mathrm{s}(\hat{x}) - \mathbb{SR}_\epsilon(S, \hat{x})| \leq B\sigma_\mathrm{s}(\hat{x}), \qquad \mu_\mathrm{s}(\hat{x}) + B\sigma_\mathrm{s}(\hat{x}) \geq \mathrm{SaR}_\epsilon(S, \hat{x}).$$

Proof: First, by the assumptions above, we know that for each data set $\mathcal{D}_\mathrm{s}$ in lines 5-7 of Algorithm 1, we have taken one sample δ_j of $\mathrm{N_{RV}}$ (potentially) different random variables $S_{\hat{x}_j}$. By Theorem 15, we know that with minimum probability $1 - (1-\epsilon)^{\mathrm{N_{RV}}}$, the maximum sample $\delta_j^* \triangleq \max\{\delta_\ell \in \mathcal{D}\}$ is greater than the Value-at-Risk of its corresponding random variable $\mathrm{VaR}_\epsilon(S_{\hat{x}_j^*})$ (VaR is defined in Definition 15). Since we assume Assumption 13 holds for each such set of random

variables, then we know that with minimum probability $1 - (1-\epsilon)^{N_{RV}}$, the sum $\delta_j^* + \beta$ is greater than the value-at-risk level ϵ of any sampled random variable, *i.e.* the sum $\delta_j^* + \beta$ is a non-noisy estimate of $\mathbb{SR}_\epsilon(S, \hat{x}),\ \forall\ \hat{x} \in \mathcal{D}_s$. Hence, repeating this same argument for each data point in $\mathbb{X}, \mathbb{Y}$ and setting $R = 0$, as each sampled point is a non-noisy sample of our upper-bounding surface, we recover the results of Theorem 14 with minimum probability $(1 - (1-\epsilon)^{N_{RV}})^s$:

$$|\mu_s(\hat{x}) - \mathbb{SR}_\epsilon(S, \hat{x})| \leq B\sigma_s(\hat{x}),\ \forall\ \hat{x} \in \hat{\mathcal{X}}. \tag{6.14}$$

Our final result holds by unraveling the absolute-value inequality in (6.14), as $\mathbb{SR}_\epsilon(S)$ is an upper-bounding surface for $\mathrm{SaR}_\epsilon(S)$. ■

Learning Disturbance Models Mid-Flight

Finally, to showcase the efficacy of our algorithm that learns a disturbance model online, we aimed to robustify a drone's controller mid-flight by recording discrepancies between model and true system evolution. All flight tests are performed at the Caltech Center for Autonomous Systems and Technology arena which is equipped with an Optitrack motion capture system that samples and streams the rotor-craft pose at 190 Hz. We belay a safeguard tether to the drone (weights 2.46 kg) with a ~200 g passive weight attached on the other end to partially eliminate tether slack, which is another source of uncertainty. Figure 6.6 depicts the two types of flight paths taken, wherein we aimed to realize complex behaviors commonly asked of drones, *e.g.* ascent and descent with both headwind and tailwind, circulating low to the ground, and taking off vertically in the presence of transverse wind. All disturbing winds were realized by The Caltech Real Weather Wind Tunnel, and windspeed information was not made available to the baseline controller. This baseline controller was developed against a single integrator model, and as such, it outputs 3-space velocities at 50 Hz for the drone to follow. The velocities provided by this controller are tracked by the drone's onboard flight controller, a Hex Cube Orange running a PX4 autopilot [190].

With respect to the mathematical setting in Section 6.3 then, we do not know our true system dynamics, though we model the system as a single integrator:

$$\hat{x}_{j+1} = \hat{x}_j + \hat{u}_j(\Delta t = 0.02),\ \hat{x}_j \in \underbrace{[-2,2]^2 \times [1.2,2]}_{\hat{\mathcal{X}}},\ \hat{u}_j \in \underbrace{[-0.8,0.8]^2 \times [-0.5,0.5]}_{\hat{\mathcal{U}}}.$$

The state projection map M_x as in (MAPS) reads the drone's position in 3-space. The input map M_u corresponds to the onboard PX4 controller that maps true drone

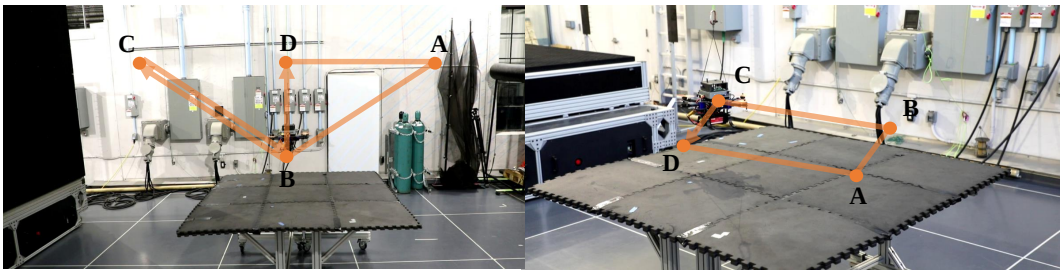

Figure 6.6: Depictions of the two types of periodic trajectories implemented in our drone experiments described in the experimentation subsection of Section 6.3. These trajectories approximate difficult types of behaviors commonly asked of drones,

states $x \in \mathcal{X}$ and commanded 3-space velocities $\hat{u} \in \hat{\mathcal{U}}$ to rotor speeds at 1 kHz. As we update these desired velocities at 50 Hz, our time-dilation parameter $K = 20$ for Algorithm 1. Finally, our observation function O as per (OBS) outputs the projected true-system 3-space position after K true-system time-steps, and our disturbance-norm samples δ as per (6.13) are defined as follows:

$$\delta_j = \|O(x_{Kj}, U(M_x(x_{Kj})) - (\hat{x}_j + U(M_x(x_{Kj}))\Delta t\|, \quad \hat{x}_j = M_x(x_{Kj}).$$

The baseline controller $U : \hat{\mathcal{X}} \to \hat{\mathcal{U}}$ is a discrete-time Lyapunov controller designed to send the single-integrator system to a provided waypoint and does not take into account complex aerodynamic effects, *e.g.* ground effects, transverse wind, and tethered disturbances, which are challenging to model and can degrade flight performance when ignored [191], [192]. Furthermore, the number of random variables sampled per data-collection step $\mathrm{N_{RV}} = 60$, and we used the squared-exponential kernel function with length-scale parameter $\ell = 1.0$ for all experiments.

Our desired outcomes were twofold. First, we fit an upper bound to the disturbance-norm *Surface-at-Risk* level $\epsilon = 0.05$ over the course of one traversal of the desired flight path. In this initial flight path, we only implement the baseline controller and augment this controller if the system takes longer than 10 seconds to reach within 0.1 m of the subsequent waypoint along the desired path. As each path comprises fewer than 6 waypoints, this ensures that our learned model considers less than a minute of data for all experiments on both flight paths. These cutoff times were specifically chosen to highlight the efficiency of our method with limited data. Second, on all subsequent flight paths, we provide from our fitted surface the norm of disturbances that the Lyapunov controller should reject while providing velocity commands. As such, we expect performance improvements from our augmented controller in the form of traversal time speedups through the series of waypoints,

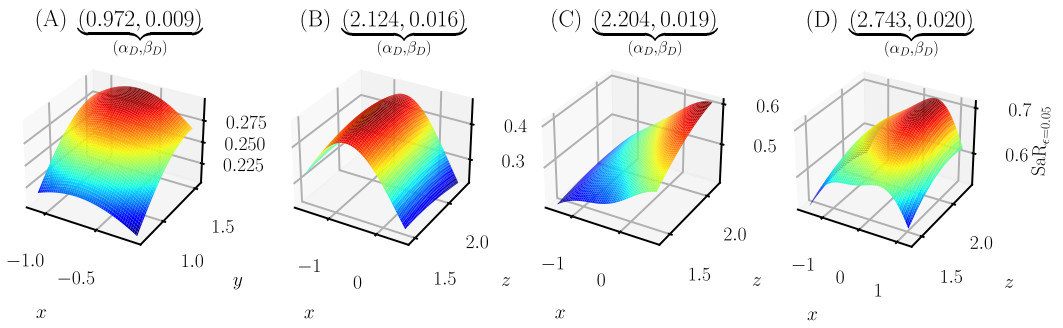

Figure 6.7: Fitted $\mathrm{SaR}_{\epsilon=0.05}$ for the four experiments depicted in Figure 6.6, with α_D the maximum distance between two sampled states for GPR, and β_D the maximum discrepancy between two sampled disturbance norms. Over all four experiments, we see a consistent 2× speedup in flight path times after implementation of the augmented controller — a qualitative result we expect as per Theorem 16, as we fit an upper bound to disturbance norms at 95% probability.

as subsequent waypoints are provided once the drone reaches within 0.1 m of the current, commanded waypoint, and the drone's controller should account for the vast majority of disturbances caused by wind, ground, and tether effects as we fitted an upper bound to the disturbance-norm *Surface-at-Risk* level $\epsilon = 0.05$.

Discussion of Results

We performed four sets of experiments: (A) Hovering and moving while maintaining a 0.15 m height above ground (see right in Figure 6.6); (B) Ascent, descent, and vertical take-off without any wind (see left in Figure 6.6); (C) The same flight path as (B) but with a 0.6 m/s transverse wind; (D) The same flight path as (B) and (C) but with a 2 m/s transverse wind. The wind flows from left to right when looking at the setup in Figure 6.6.

Figure 6.7 shows the fitted $\mathrm{SaR}_{\epsilon=0.05}$ for each of the four experiments (A)-(D) ran on the drone, as labeled prior. As mentioned, in all cases we see at least a 2× speedup in flight path times when implementing the augmented controller, with as much as a 5× speedup in the hovering case (A). Furthermore, we were also able to verify Assumption 13 with respect to the data sets we collected for each experiment. Specifically for (A), we assumed that for states within $\alpha = 1$m, their Values-at-Risk level $\epsilon = 0.05$ would not change by more than $\beta = 0.05$. As can be seen in the title of the associated subfigure in Figure 6.7, the reported values from data are smaller than their assumed counterparts, indicating that Assumption 13 held over this experiment, at least with respect to the collected data. For the remaining

experiments, the assumed α, β values were as follows: (B) $\alpha = 3$m, $\beta = 0.05$, (C) $\alpha = 3$m, $\beta = 0.1$, and (D) $\alpha = 3$m, $\beta = 0.2$. Therefore, as we can see from the associated titles in Figure 6.7, we are similarly able to verify that Assumption 13 held over each of these cases as well — at least with respect to the data collected. As such, we expected a significant increase in performance according to Theorem 16 as was realized in all four cases with respect to flight path time speedups. All experiments can also be seen in our supplementary video here [193].

6.4 Probabilistic Guarantees for Finite-Time Optimal Controllers

The prior two sections in this chapter leveraged the prior, risk-aware verification results to provide procedures to validate models both online and offline. Then, using existing work in input-to-state-safe and input-to-state-stable control, we robustified controllers either before or during operation by leveraging these validated, augmented models. That being said, both of these procedures aimed to streamline risk-aware controller synthesis and verification by validating the models we use in either procedure. This section takes it one step further and describes how similar techniques can facilitate "optimal" input selection and guarantee generation for finite-time optimal controllers. Our desire to do so arises as optimal controllers provide a natural way of expressing and segmenting disparate control objectives, as can be easily seen in works regarding model predictive control (MPC) [169]–[171], control barrier functions [145], [146], [172], and optimal path planning [173]–[175], among others. Therefore, the ability to rapidly "solve" and provide guarantees for these non-convex optimal controllers, would facilitate their everyday use and the translation-to-practice of the ease of expressing these control objectives. To keep this section self-contained then, we will start with a brief motivation.

General Motivation and Problem Statements

We assume the existence of a nonlinear discrete-time system whose dynamics f are (potentially) unknown:

$$x_{k+1} = f(x_k, u_k, d),\ x \in \mathcal{X},\ u \in \mathcal{U},\ d \in \mathcal{D}. \tag{6.15}$$

Here, $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{U} \subseteq \mathbb{R}^m$ is the input space, and $\mathcal{D} \subseteq \mathbb{R}^p$ is the space of variable objects in our environment that we can control, *e.g.* center locations of obstacles and goals for path-planning examples, variable wind-speeds for a drone, *etc*. Provided this dynamics information, a cost J, state constraints, and input constraints, one could construct a Nonlinear Model Predictive Controller of

the following form (with $j \in [0, 1, \ldots, H-1]$):

$$\begin{aligned} \mathbf{u}^* = \underset{\mathbf{u}=(u^0,u^1,\ldots,u^{H-1})\in\mathcal{U}^H}{\operatorname{argmin}} \quad & J(\mathbf{u}, x_k, d), \\ \text{subject to} \quad & x_k^{j+1} = f(x_k^j, u^j, d), \\ & x_k^0 = x_k, \\ & x_k^{j+1} \in \mathcal{X}_k^{j+1}, \\ & u^j \in \mathcal{U}. \end{aligned} \tag{NMPC}$$

For the analysis to follow, however, we note that the general NMPC problem posed in (NMPC) can be posed as the following Finite-Time Optimal Control Problem.

$$\begin{aligned} \underset{\mathbf{u}=(u^0,u^1,\ldots,u^{H-1})\in\mathcal{U}^H}{\operatorname{argmin}} \quad & J(\mathbf{u}, x_k, d), \\ \text{subject to} \quad & \mathbf{u} \in \mathbb{U}(x_k, d) \subseteq \mathcal{U}^H. \end{aligned} \tag{FTOCP}$$

Here, J is a bounded (perhaps) nonlinear cost function, and $\mathbb{U}$ is a set-valued function outputting a constraint space for input sequences that (potentially) depends on the initial system and environment states (x_k, d), respectively. Specific examples following this general form will be provided in the experimentation sections to follow. Finally, $H > 0$ is the horizon length for the finite-time optimal control problem. Then, the three problem statements to-be-considered will follow.

Problem 7. *Develop a procedure to identify input sequences* $\mathbf{u}$ *that are in the* $100(1-\epsilon)$%*-ile for some* $\epsilon \in (0, 1]$ *with respect to solving* (FTOCP).

Problem 8. *Develop a procedure to determine whether* (FTOCP) *is recursively feasible.*

Problem 9. *Develop a procedure to upper bound maximum controller runtimes for optimal controllers of the form in* (FTOCP) *on their respective hardware instantiations, i.e. the maximum elapsed time for producing an input (sequence) on the physical robot during operation.*

Percentile-Based Input Selection

Problem 7 references the development of an efficient method to solve (FTOCP). To that end, we aim to take a percentile method as described in Chapter 4. As a result, our corollary in this vein stems directly from Theorem 9, though we will make one clarifying assumption.

Assumption 14. Let J and $\mathbb{U}$ be as per (FTOCP), let $\mathcal{V}$ be as per (4.10) with respect to the decision space $\mathbb{U}(x_k, d)$, and let F be as per (4.11) with respect to this cost J and $\mathbb{U}(x_k, d)$. Furthermore, let J be bounded over $\mathbb{U}(x_k, d)$, and let $\mathbb{U}(x_k, d)$ be a set of bounded volume (or finitely many elements if a discrete set) for any choice of $(x_k, d) \in \mathcal{X} \times \mathcal{D}$ (these sets defined in (6.15)). Finally, let $\{(\mathbf{u}_i, J(\mathbf{u}_i, x_k, d))\}_{i=1}^N$ be a set of N uniformly sampled sequences $\mathbf{u}_i$ from $\mathbb{U}(x_k, d)$ with their corresponding costs, and let $\mathbf{u}_N^*$ be the (potentially) non-unique sequence with minimum cost.

Corollary 21. *Let Assumption 14 hold and let* $\epsilon \in (0, 1]$. *Then,* $\mathbf{u}_N^*$ *is in the* $100(1-\epsilon)$%*-ile with respect to minimizing* J *at the current system and environment state* (x_k, d) *with minimum confidence* $1 - (1-\epsilon)^N$*, i.e.,*

$$\mathbb{P}^N_{\mathrm{U}[\mathbb{U}(x_k,d)]}\left[\mathcal{V}(F(\mathbf{u}_N^*)) \leq \epsilon\right] \geq 1 - (1-\epsilon)^N.$$

Proof: Stems directly via Theorem 9. ■

In short, Corollary 21 tells us that if we have a finite-time optimal control problem of the form in (FTOCP), where for some system and environment state (x_k, d), the cost function J is bounded over a bounded decision space $\mathbb{U}(x_k, d)$, then we can take a percentile approach to identify input sequences that are better than a large fraction of the space of all feasible input sequences. Notably, this statement is made independent of the convexity, or lack thereof, of (FTOCP), making it especially useful for non-convex MPC. Furthermore, as is done in the quadrupedal experimentation section to follow, one can further optimize over the outputted percentile solution $\mathbf{u}_N^*$ via gradient descent — should gradient information be available. The resulting solution then retains the same confidence on existing within the same percentile, while also being efficient to calculate. This does introduce new questions. Namely, will a percentile solution always exist, and how efficiently can we calculate these sequences on hardware? These questions will be answered in the sections to follow.

Determining Recursive Feasibility

Problem 8 references the development of an algorithm to efficiently determine the recursive feasibility of (FTOCP). To ease the statement of the theoretical results to follow, we indicate via $|\mathbb{U}(x_k, d)|$ the "size" of the constraint space $\mathbb{U}(x_k, d)$ for (FTOCP), with $|\varnothing| = 0$. Additionally, we will assume that there exists some controller U that either utilizes the aforementioned percentile method in Section 6.4 or some other technique to produce (approximate) solutions to (FTOCP), *i.e.*

$$\exists\, U : \mathcal{X} \times \mathcal{D} \to \mathcal{U} \text{ s.t. } U(x, d) = u \in \mathcal{U}. \tag{6.16}$$

Furthermore, we will indicate via the following notation, the evolution of our system under this controller U, provided an initial system and environment state:

$$x^+[x, d] = f(x, U(x, d), d).$$

This allows us to formally define recursive feasibility.

Definition 28. An optimal controller of the form in (FTOCP) is *recursively feasible* if and only if for all system and environment states, the feasible space for (FTOCP) is non-empty for successive timesteps, *i.e.*

$$\forall\ (x, d) \in \mathcal{X} \times \mathcal{D},\ |\mathbb{U}(x, d)| > 0 \implies |\mathbb{U}(x^+[x, d], d)| > 0.$$

As motivated earlier, we can express recursive feasibility determination as an optimization problem. Specifically, let our cost function C be as follows:

$$C(x, d) = \begin{cases} 1 & \text{if } |\mathbb{U}(x, d)| > 0 \implies |\mathbb{U}(x^+[x, d], d)| > 0, \\ -1 & \text{else.} \end{cases} \tag{6.17}$$

We can generate a minimization problem provided this cost function C over the joint state space $\mathcal{X} \times \mathcal{D}$:

$$\min_{x \in \mathcal{X},\ d \in \mathcal{D}} C(x, d). \tag{6.18}$$

If the solution to (6.18) were positive, then (FTOCP) is recursively feasible. Likewise, if the solution were negative, then there exists a counterexample. As a result, not only can we express recursive feasibility determination as an optimization problem, but this problem is also of the same form as in (4.9), permitting a probabilistic solution approach as expressed in the following assumption and corollary.

Assumption 15. Let C be as per (6.17), let $\mathcal{X}, \mathcal{D}$ be as per (6.15) and also be spaces of bounded volume, let $\{C(x_i, d_i)\}_{i=1}^N$ be a set of N cost evaluations of decision tuples (x_i, d_i) sampled independently via $\mathrm{U}[\mathcal{X} \times \mathcal{D}] \triangleq \mu$, let ζ_N^* be the minimum cost evaluation, and let $\epsilon \in [0, 1]$.

Corollary 22. *Let Assumption 15 hold. Then if* $\zeta_N^* = 1$, (FTOCP) *is successively feasible with minimum probability* $1 - \epsilon$ *and with minimum confidence* $1 - (1 - \epsilon)^N$.

Proof: By Theorem 9 we have that:

$$\mathbb{P}_\mu^N \left[\mathbb{P}_\mu \left[C(x, d) \geq \zeta_N^* \right] \geq 1 - \epsilon \right] \geq 1 - (1 - \epsilon)^N.$$

By definition of C in (6.17), if $\zeta_N^* = 1$, then with minimum probability $1 - \epsilon$, $|\mathbb{U}(x, d)| > 0 \implies |\mathbb{U}(x^+[x, d], d)| > 0$. In other words, with minimum probability $1 - \epsilon$, if (FTOCP) were feasible at the prior time step, then it will also be feasible at the next time step, *i.e.* successively feasible. ■

In other words, Corollary 22 tells us that to probabilistically determine whether a given finite-time optimal control problem is successively feasible, it is sufficient to identify at least one input in the constraint space for successive optimization problems starting at N randomly sampled state pairs (x, d). Determining at least one such input could be achieved by querying the corresponding controller U or some other desired method. Notably, this does not guarantee recursive feasibility as that would correspond to the optimal value of (6.18) being positive. However, with arbitrarily high probability, we can provide guarantees that even hardware controllers will be successively feasible for sampled state pairs $(x, d) \in \mathcal{X} \times \mathcal{D}$, which is the underlying requirement for recursive feasibility as per Definition 28.

Determining Hardware-Specific Controller Runtimes

Lastly, Problem 9 references the development of an algorithm to efficiently identify maximum controller runtimes on existing system hardware. To address this from a probabilistic perspective, we will first define some notation. To start, we will use the same controller U as per equation (6.16). We also denote via T a timing function that outputs the evaluation time for querying the controller U at a given state pair (x, d), *i.e.* $T : \mathcal{X} \times \mathcal{D} \to \mathbb{R}_{++}$. Then we can nominally express maximum controller runtime determination as an optimization problem:

$$\max_{x \in \mathcal{X},\, d \in \mathcal{D}} \; T(x, d). \tag{6.19}$$

Under the fairness assumption that the controller does have a bounded runtime, however, identification of a probabilistic maximum runtime is solvable via probabilistic optimization procedures as outlined by Theorem 9. As prior, we will state a clarifying assumption and the formal corollary statement will follow.

Assumption 16. Let T be as per (6.19), let $\mathcal{X}, \mathcal{D}$ be as per (6.15) and be of bounded volume, let $\{T(x_i, d_i)\}_{i=1}^N$ be a set of N controller runtimes for state pairs (x_i, d_i) sampled independently via $\mathrm{U}[\mathcal{X} \times \mathcal{D}] \triangleq \mu$, let ζ_N^* be the maximum runtime, and let $\epsilon \in [0, 1]$.

Corollary 23. *Let Assumption 16 hold. The probability of sampling a state pair*

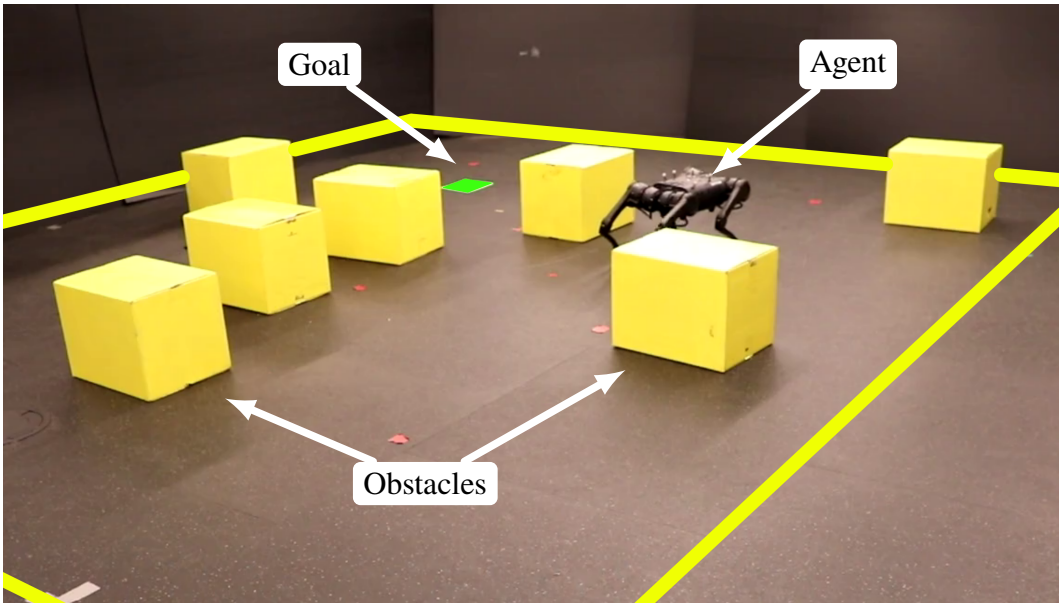

Figure 6.8: Experimental setup for Quadruped reach-avoid tests.

whose controller runtime is at most ζ_N^* *is at-least* $1-\epsilon$ *with confidence* $1-(1-\epsilon)^N$*:*

$$\mathbb{P}_\mu^N\left[\mathbb{P}_\mu\left[T(x,d) \leq \zeta_N^*\right] \geq 1-\epsilon\right] \geq 1-(1-\epsilon)^N.$$

Proof: Via Theorem 9,

$$\mathbb{P}_\mu^N\left[\mathbb{P}_\mu\left[-T(x,d) \geq -\zeta_N^*\right] \geq 1-\epsilon\right] \geq 1-(1-\epsilon)^N,$$

and flipping the innermost inequality provides the result. ■

In short then, Corollary 23 tells us that probabilistic determination of maximum controller runtimes stems easily by recording controller runtimes for N randomly sampled scenarios identified through N randomly sampled system and environment state pairs (x,d) from $\mathcal{X}\times\mathcal{D}$. Now, we will showcase the efficacy of these probabilistic approaches to optimal input selection and guarantee generation, on a few hardware examples. We will start with a quadrupedal example.

Quadrupedal Walking

Reach Avoid Navigation Task: In the quadruped example, the agent is tasked to reach a specific goal location (green) while avoiding static obstacles (yellow) within a 5m by 4m space. The agent and obstacles move and can be placed continuously within this space. The set of all environments $\mathcal{D}$ corresponds to the set of all setups, including goals, robot starting locations, and obstacles, that satisfy the

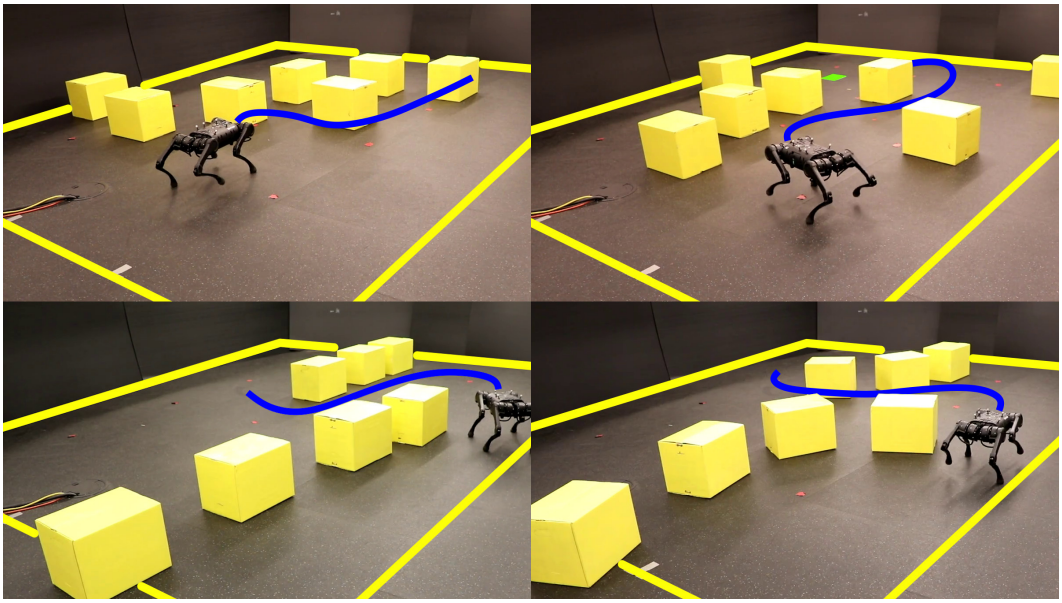

Figure 6.9: Depictions of the randomized environments $\mathcal{D}$ for the Quadruped experiments. Yellow boxes are static obstacles, the goal is shown in green (not visible in all images), and a cartoon of the computed plan is shown in blue.

aforementioned conditions while allowing for at-least one feasible path to the goal. Figure 6.8 depicts an example setup, with Figure 6.9 showing multiple examples of viable environments in $\mathcal{D}$.

FT-OCP formulation: We formulated quadrupedal navigation as an optimal control problem of the form in (FTOCP). We consider as states, the position of the robot within a bounded rectangle $\mathcal{X} = [0, 5]\times[0, 4]$. Individual inputs are discrete changes in position with bounded magnitude, with corresponding H-length input sequence $\mathbf{u}$ a finite horizon of positional waypoints. Mathematically, the state-dependent subset of permissible sequences $\mathcal{U}_p^H(x)$ is as follows, with $j \in [0, 1, \ldots, H-2]$:

$$\mathcal{U}_p^H(x) = \left\{ \mathbf{u} \in \mathcal{U}^H \;\middle|\; \begin{matrix} \|u^0 - x\| \leq 0.03, \text{ and }, \\ \|u^{j+1} - u^j\| \leq 0.03. \end{matrix} \right\}$$

$\mathbb{U}(x_k, d)$ then further constrains $\mathbf{u}$ to remain within a feasible set of states via a discrete barrier-like condition. To define that feasible state set, for D obstacle positions let $d = [d_1^T, d_2^T, \ldots, d_D^T]^T \in \mathbb{R}^{2\times D}$. Then with a collision radius r, the feasible state set is:

$$\mathcal{F}(d) = \{x \in \mathcal{X} \mid ||x - d_j|| \geq r \; \forall j = 1, ..., D\}.$$

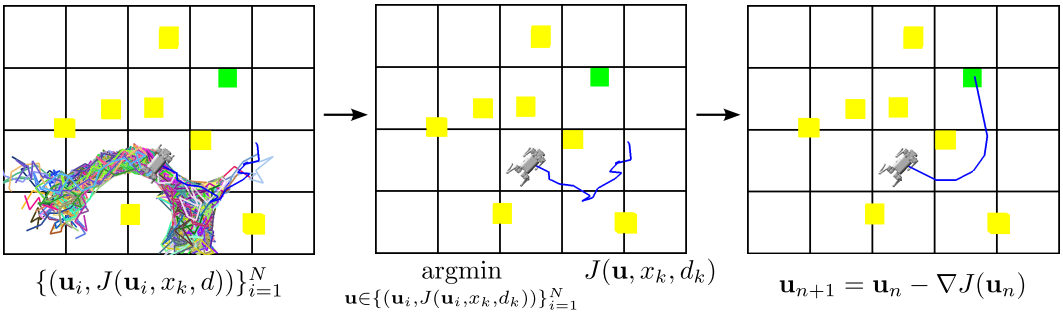

Figure 6.10: Solving the FT-OCP for the quadruped reach-avoid experiment. (a) generates uniformly random feasible input sequence samples. (b) selects the best sample according to cost function $J(\mathbf{u}, x_k, d_k)$. Finally, (c) leverages the differentiability of J to further improve the choice of $\mathbf{u}$ via constrained gradient descent.

Then we can define the overall constrained input space $\mathbb{U}(x, d)$ as follows, with $x^0 = x$, $x^{j+1} = f(x^j, u^j, d)$, and $\forall\, \ell \in 0, 1, \ldots, H$:

$$\mathbb{U}(x, d) = \left\{\mathbf{u} \in \mathcal{U}_p^H(x) \mid x^\ell \in \mathcal{F}(d)\right\}. \tag{6.20}$$

Here, the discrete-time dynamics are simply $f(x, u, d) = x + u$. Finally, with goal state x_d, we have our cost function J as follows, again with $x^0 = x$ and $x^{j+1} = f(x^j, u^j, d)$:

$$J(\mathbf{u}, x, d) = 10||x^H - x_d|| + \sum_{i=0}^{H-1} ||x^{i+1} - x^i||. \tag{6.21}$$

This cost simultaneously rewards the final waypoint when closer to the goal and rewards a shorter overall path length. As a result, the overall finite-time optimal control problem is:

$$\begin{aligned} \mathbf{u}^* = \underset{\mathbf{u}\in\mathcal{U}^H}{\text{argmin}} \quad & J(\mathbf{u}, x_k, d) \text{ as per (6.21)}, \\ \text{subject to} \quad & \mathbf{u} \in \mathbb{U}(x_k, d) \text{ as per (6.20)}. \end{aligned} \tag{6.22}$$

Solving the FT-OCP: To solve (6.22), we employ the procedure described in Section 6.4. We employ rejection sampling over the input space $\mathcal{U}^H$ to generate samples $\mathbf{u} \in \mathbb{U}(x_k, d)$ until we collect 1000 such samples. From this collection of samples, we choose the minimum cost sample by evaluating $J(\mathbf{u}, x_k, d)$. This sample meets our guarantees as described in Corollary 21. However, we recognize that our cost function is differentiable in $\mathbf{u}$, and we can employ constrained gradient descent [194] to further improve the solution. This process is illustrated in Figure 6.10.

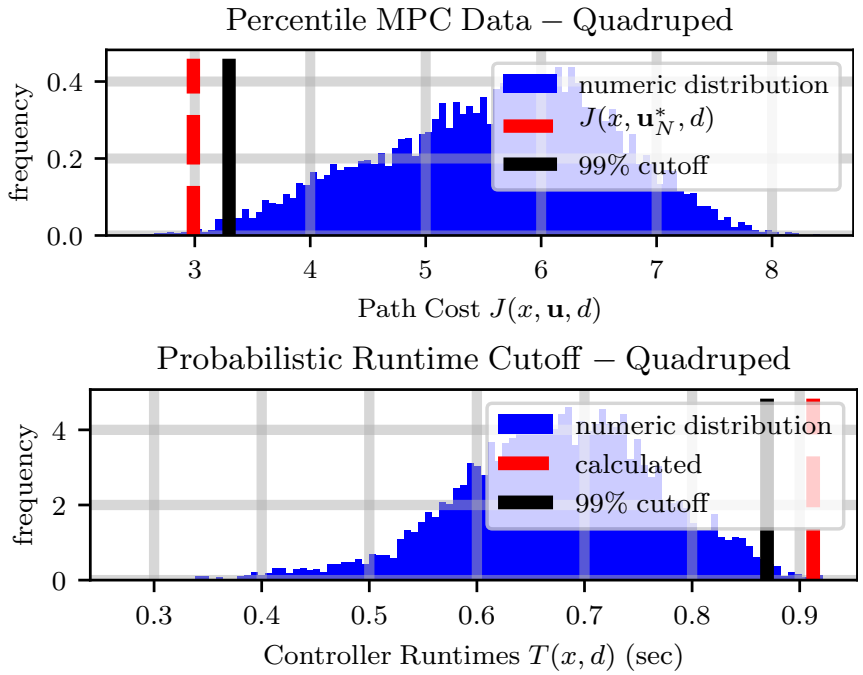

Figure 6.11: Quadruped Hardware data when (top) taking a percentile method to solve the quadruped's finite-time-optimal controller, and (bottom) calculating a probabilistic cutoff on maximum controller runtime for the same controller. In both cases, the red lines corresponding to (top) the identified path and (bottom) the reported maximum controller runtime are to the left and right, respectively, of their corresponding, true probabilistic cutoffs. This affirms Corollaries 21 and 23 insofar as the identified values satisfy their corresponding probabilistic statements. Numeric distributions were calculated by evaluating 5000 random samples.

Experiments and Results: Tests were performed for both random and curated obstacle locations, with care taken to reject samples without a feasible path to the goal. The quadruped was given a random start position and orientation, and a fixed goal, x_d. (6.22) was solved using a Python implementation of the above procedure at ~1.5 Hz, taking x_k to be the position of the quadruped as measured by an Optitrack motion capture system. An IDQP-based walking controller [182] tracked the computed plan with tangent angles along the desired path used as desired quadruped heading for tracking purposes.

By Corollary 21, choosing the best out of 1000 uniformly chosen waypoint sequences implies that the best sequence $\mathbf{u}_N^*$ should be in the 99%-ile with 99.995% confidence. This is indeed the case as can be seen in the data portrayed at the top of Figure 6.11, corroborating Corollary 21. Both Corollaries 22 and 23 were also corroborated by recording successive feasibility and controller runtimes for 1000 randomized

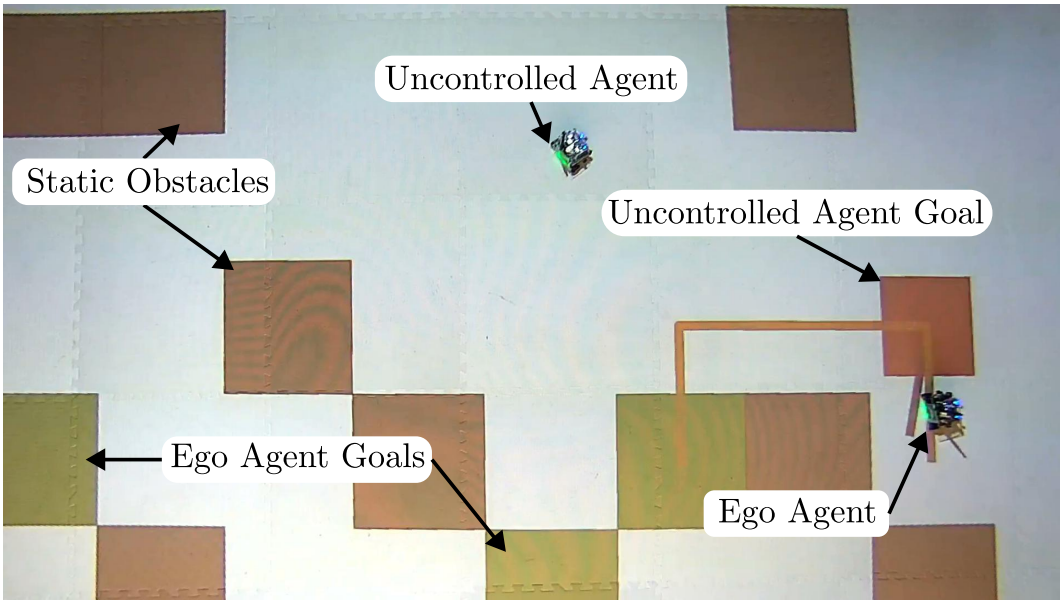

Figure 6.12: Experimental setup for Robotarium reach-avoid tests.

instances of the percentile method applied to (6.22). In all cases, the controller was successively feasible, and the maximum controller runtime was 0.92 seconds. Comparing against another 5000 random samples affirms that the reported maximum runtime exceeded the 99%-ile cutoff, while the controller was successively feasible in all instances as well. The data for runtimes is shown at the bottom of Figure 6.11. Qualitatively speaking, however, the proposed procedure produces a valid, collision-free plan in all tested scenarios. This plan ultimately leads to the quadruped reaching the desired goal in many scenarios. However, some obstacle placements lead to local minima that cannot be escaped, as this is a finite-time method. Increasing the horizon H allows for success in these conditions but requires a trade-off in execution time. These results are elucidated in the supplemental video.

Multi-Agent Verification

Figure 6.12 depicts the reach-avoid scenario for the Robotarium [144] agents which can be modeled as unicycle systems, *i.e.* with $x_k \in \mathcal{X},\ u_k \in \mathcal{U}$:

$$x_{k+1} = \underbrace{x_k + (\Delta t = 0.033) \begin{bmatrix} \cos(x_k[3]) & 0 \\ \sin(x_k[3]) & 0 \\ 0 & 1 \end{bmatrix} u_k}_{f(x_k, u_k, d)} . \tag{6.23}$$

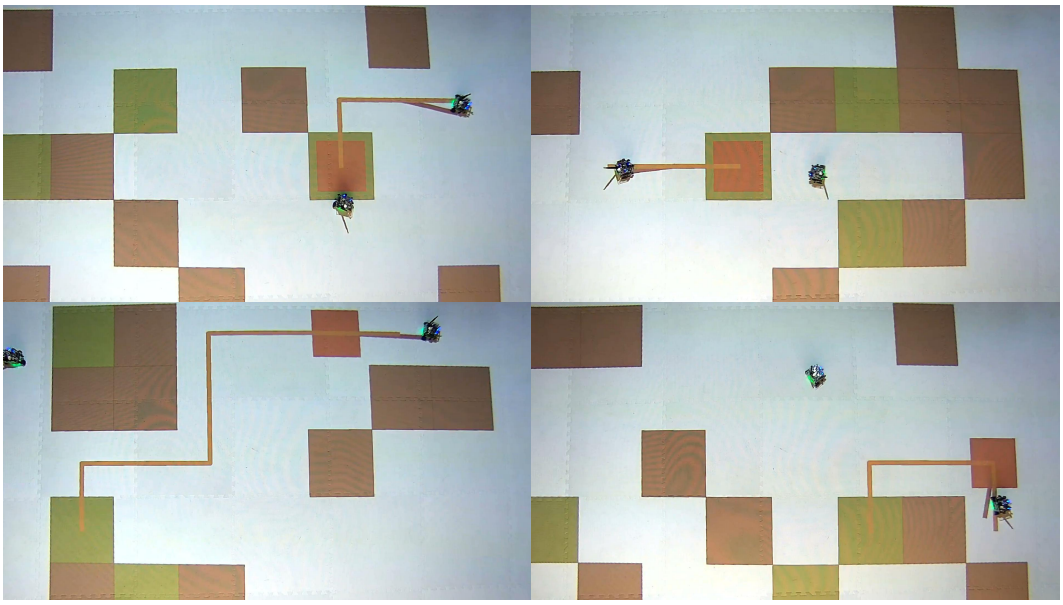

Figure 6.13: Experimental depictions of the randomized environments $\mathcal{D}$ for the Robotarium as described in Section 4.5. The black squares correspond to static obstacles, the green squares correspond to goals for the ego-agent whose shortest path from its starting cell is shown in orange, and the red squares correspond to the un-controlled agent's goal.

Here, $\mathcal{X} = [-1.6, 1.6] \times [-1.2, 1.2] \times [0, 2\pi]$ and $\mathcal{U} = [-0.2, 0.2] \times [\frac{-\pi}{2}, \frac{\pi}{2}]$. Additionally, each agent comes equipped with a Lyapunov controller U that steers the agent to a provided waypoint $w \in \mathcal{W}$:

$$U : \mathcal{X} \times \mathcal{D} \times \mathcal{W} \triangleq [-1.6, 1.6] \times [-1.2, 1.2] \to \mathcal{U}.$$

The environment space $\mathcal{D}$ consists of the grid locations of 8 static obstacles on an 8×5 grid overlaid on the state space $\mathcal{X}$, the cells of 3 goals on the same grid, the starting position in $\mathcal{X}$ of another, un-controlled moving agent that is at-least 0.3 meters away from the ego agent of interest, and the un-controlled agent's goal cell on the same grid. No static obstacles are allowed to overlap with any of the goals, though the un-controlled agent's goal may overlap with at least one of the goals of the ego agent, and the setup of static obstacles must always allow for there to exist at least one path to one of the ego agent's goals. Figure 6.13 shows multiple examples of environment setups within $\mathcal{D}$.

NMPC Formulation: Based on the setup of static obstacles and goal locations on the grid, we define a function $S : \mathcal{W} \to \mathbb{R}_+$ that outputs the length of the shortest feasible path to a goal from a provided planar waypoint. Should no feasible path

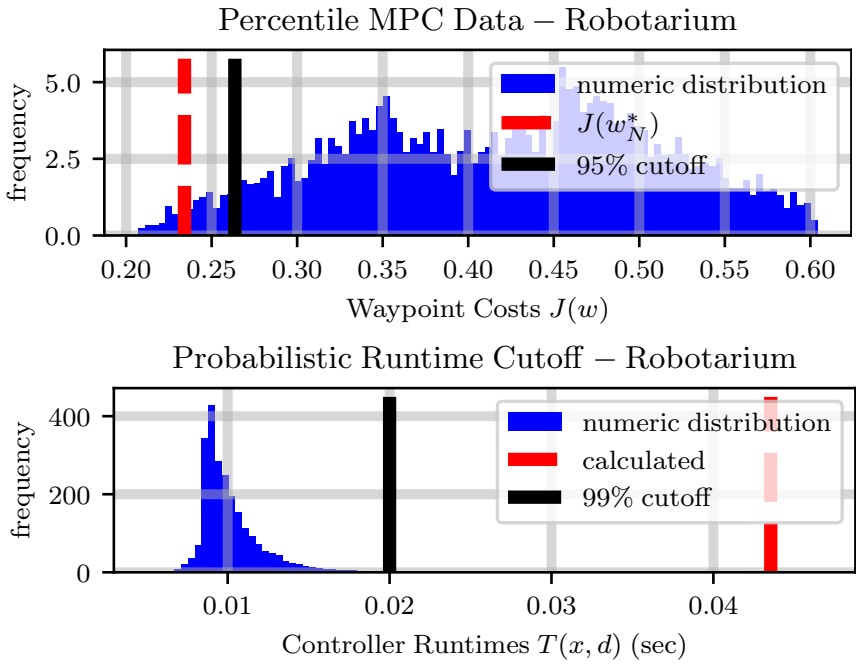

Figure 6.14: Robotarium Hardware data when (top) taking a percentile method to solve the multi-agent finite-time optimal controller, and (bottom) calculating a probabilistic cutoff on maximum controller runtime for the same controller. In both cases, the red lines corresponding to (top) the identified waypoint and (bottom) the reported maximum controller runtime are to the left and right, respectively, of their corresponding, true probabilistic cutoffs. In other words, the identified values satisfy their corresponding probabilistic statements, affirming Corollaries 21 and 23. Numeric distributions were calculated by evaluating 5000 random samples.

exist from a waypoint $w \in \mathcal{W}$, $S(w) = 100$ to indicate infeasibility. Inspired by discrete control barrier function theory [59], we define a control barrier function h which accounts for both the ego agent state x_a and the un-controlled agent state x_o (with $P = [I_{2\times 2}\ \mathbf{0}_{2\times 1}]$):

$$h(x_a, x_o) = \begin{cases} -5 & \text{in static obstacle cell,} \\ \|P(x_a - x_o)\| - 0.18 & \text{else.} \end{cases}$$

Then, provided $h(x_a, x_o) \geq 0$, the ego agent hasn't crashed into a static obstacle and is maintaining at least a distance of 0.18 m from the un-controlled agent.

This permits us to define an NMPC problem as follows with the dynamics f as

per (6.23) and $\forall\ j \in [1, 2, 3, 4, 5]$:

$$
\begin{aligned}
w_k^* = \underset{w \in \mathcal{W}}{\text{argmin}} \quad & S(w), && \text{(NMPC-A)}\\
\text{subject to} \quad & x_k^j = f(x_k^{j-1}, u^{j-1}, d), && \text{(a)}\\
& x_k^0 = x_k, && \text{(b)}\\
& h(x_{k,a}^j, x_o) \geq 0 && \text{(c)}\\
& u^{j-1} = U\left(x_k^{j-1}, d, w\right), && \text{(d)}\\
& 0.05 \leq \|w - x_k\| \leq 0.2.
\end{aligned}
$$

To ease sampling then, we will consider an augmented cost J that outputs 100 whenever a waypoint w fails to satisfy constraints (a)-(d) in (NMPC-A). Then we define the NMPC problem to-be-solved as follows:

$$
\begin{aligned}
w_k^* = \underset{w \in \mathcal{W}}{\text{argmin}} \quad & J(w), && \text{(NMPC-B)}\\
\text{subject to} \quad & 0.05 \leq \|w - x_k\| \leq 0.2.
\end{aligned}
$$

Results: By Corollary 21, if we wish to take a percentile approach to determine a waypoint w_N^* in the 95%-ile with 99.4% confidence we need to evaluate $N = 100$ uniformly chosen waypoints from the constraint space for (NMPC-B). Figure 6.14 shows the cost of the outputted waypoint sequence compared against 5000 randomly sampled values, and as can be seen, the outputted waypoint w_N^* is indeed in the 95%-ile, confirming Corollary 21. Calculating this controller's runtime in 460 randomly sampled initial state and environment scenarios yielded a probabilistic maximum $\zeta_N^* = 0.043$ seconds. According to Corollary 23, this maximum runtime should be an upper bound on the true, 99% cutoff on controller runtimes with confidence 99% — and as can be seen in Figure 6.14, ζ_N^* exceeds the true value. Finally, to corroborate Corollary 22, we evaluated the recursive feasibility cost function C as per (6.17) in each of the same 460 randomly sampled scenarios from prior. In each scenario, the percentile controller was successively feasible, indicating that with 99% probability the controller will be successively feasible. Evaluating the same cost for 5000 more uniformly chosen samples resulted in the controller being successively feasible each time, corroborating Corollary 22.

6.5 Conclusion

This chapter detailed the application of our risk-measure bounding and percentile optimization procedures to various aspects of the nominal controller synthesis

paradigm. Specifically, we used a combination of both procedures to develop stochastic system models offline, against which we can synthesize and verify controllers that exhibit similar performance on hardware as exhibited on the stochastic simulator. Next, we provided an online procedure to learn a disturbance Surface-at-Risk, a concept we introduced, with limited system data. By leveraging existing input-to-state stable Lyapunov results, we can query this disturbance Surface-at-Risk to robustify controllers during operation as we showed on a drone mid-flight. Finally, we detailed how percentile optimization methods could facilitate input selection and guarantee generation for finite-time non-convex optimal controllers, as the guarantees can be expressed as optimization problems themselves.

Chapter 7

SUMMARY AND FUTURE WORK

As mentioned in the first introductory chapter, the premise of this thesis was to detail our efforts in the development of a risk-aware verification pipeline that addressed a few key challenges hindering theoretical development in black-box safety-critical system verification. As such, in Chapter 3 we focused on a precursor to the verification pipeline, difficult test generation. Specifically, we mentioned how traditional test generation methods phrase difficult test synthesis as an optimization problem. As such, these methods yield "static" tests insofar as they do not react to system choices made during the test, and they are controller-specific insofar as the objective for said optimization problem depends on the controller for the system under test. However, we posited that what is difficult for a system to achieve should be independent of the controller used to steer it and that reactive tests should help uncover more problematic system behavior than static tests. To that end, we detailed the development of a reactive, controller-agnostic test-synthesis method for both continuous and discrete-time systems subject to reach-avoid signal temporal logic specifications. We showed that this method, phrased as a game-theoretic optimization problem, always has a solution, and this solution corresponds to the most difficult test of system behavior at that state. Finally, we showcased the ability of this method to generate realizable and difficult tests on hardware, by performing the tests provided by the procedure on a quadruped in satisfaction of a simple objective.

Chapter 4 focused on the verification pipeline as a whole and mentioned that the primary hindrance to theoretical development in this vein is our inability to calculate risk measures for random variables whose distributions are unknown. Additionally, both risk-aware verification and controller synthesis are traditionally expressed as (non-convex) optimization problems, and despite recent advances in learning-based optimization, *e.g.* Bayesian Optimization, Thompson Sampling, *etc*, identifying optimal solutions to these non-convex problems remains difficult. As a result, this chapter detailed our efforts in risk-measure estimation for random variables whose distributions are unknown. This was to facilitate risk-aware verification in the following chapter. Likewise, we mentioned how the same methods could generate percentile solutions to non-convex optimization problems, laying the groundwork for risk-aware controller synthesis in the following chapter. Finally, we mentioned how

multiple applications of a percentile approach could also bound the sub-optimality of a provided percentile solution, and showcased these results on a few benchmark optimization problems including the traveling salesman problem.

Chapter 5 leveraged the mathematical foundation offered by the prior chapter to formalize our risk-aware safety-critical controller verification and synthesis pipelines. Specifically, we mentioned how risk-aware verification can be phrased as a risk-measure estimation problem and utilized the prior bounding schemes to provide probabilistic verification statements for safety-critical systems subject to any signal temporal logic specification. As this method outputs a probabilistic lower bound on achievable system robustnesses, we noted that risk-aware controller synthesis can be intuitively expressed as an optimization problem to design controllers that maximize this probabilistic lower bound. Furthermore, we provided sample-complexity results for both risk-aware verification and synthesis and showed the efficacy of our approaches on a few simulated examples. Lastly, by verifying both the produced, risk-aware controller and the baseline controller for the simulator utilized, we show that our procedure provides a noticeable increase in safe performance, as expected via our optimization-based approach.

Finally, Chapter 6 started to utilize these mathematical results in different contexts related to safety-critical control. Specifically, we mentioned how we can determine model inaccuracies, *i.e.* the sim2real gap, by phrasing such a determination as a risk-measure estimation problem and can produce a stochastic model that provably approximates real-world system evolution by utilizing the determined bound. Then, we can synthesize and verify controllers in simulation against this stochastic model, to produce controllers that exhibit reliable performance on hardware. Extending this offline approach, we also offered an approach to learn disturbance models online with limited system data which can be leveraged using existing input-to-state stable results to robustify controllers mid-operation. We showcased the results of this online procedure by robustifying a drone's controller mid-flight. Finally, we detailed how percentile optimization schemes can facilitate optimal input selection and guarantee generation for non-convex optimal controllers and similarly showcased our results on multiple hardware systems as well.

7.1 Future Work

One vein of future work concerns extensions to our uncertainty quantification results mentioned in Chapter 4. First, the risk-measure estimation results are accurate but

conservative. Due to their utilization in all aspects of the mentioned risk-aware verification and synthesis pipelines, reducing this conservatism would directly impact our ability to generate safe, effective, and verified controllers that do not unnecessarily sacrifice performance. Second, decreasing the conservatism of the reported optimality gap upper bounds would increase our confidence that the reported percentile solutions are "close" to optimality. Another vein of work concerns the utilization of these methods more broadly. For example, risk-aware controller verification is just one type of verification problem. The notion of verification, however, exists broadly in multiple fields. As such, we believe there is a tremendous amount of interdisciplinary work involving the quantification of assumption or objective satisfaction and the successive application of these risk-aware verification results to make verification statements in a variety of other fields. In a similar vein, one could also utilize the risk-aware synthesis results to inform sub-optimal but useful decision selection in these areas as well. Game theoretic decision-making is a prime example here. Specifically, the identification of Nash equilibria for non-convex min-max games remains an exceedingly difficult problem. However, verifying whether a given decision pair outperforms a large fraction of possible decision pairs is a well-posed verification problem. Similarly, the identification of "optimal" decision pairs in a percentile sense is a well-posed percentile optimization problem as well. Genetic design is yet another example. Here, we can view the gene to be designed as a vector of elements taking values in a discrete set. Quantifying an objective to be optimized and saturating the corresponding function corresponds to an optimization problem easily addressed by a percentile approach. From a philosophical standpoint then, the aim would be to start viewing optimization problems as the lens through which we select "good" decisions. From that point, we can start to leverage existing control theoretic tools and our percentile method to rapidly synthesize useful decisions in arbitrary contexts across multiple fields.

BIBLIOGRAPHY

[1] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," *Foundations and Trends® in Robotics*, vol. 5, no. 4, pp. 261–349, 2017, ISSN: 1935-8253. DOI: 10.1561/2300000052. [Online]. Available: http://dx.doi.org/10.1561/2300000052.

[2] N. Akalin, A. Kristoffersson, and A. Loutfi, "Do you feel safe with your robot? factors influencing perceived safety in human-robot interaction based on subjective and objective measures," *International Journal of Human-Computer Studies*, vol. 158, p. 102 744, 2022, ISSN: 1071-5819. DOI: https://doi.org/10.1016/j.ijhcs.2021.102744. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1071581921001622.

[3] A. Zacharaki, I. Kostavelis, A. Gasteratos, and I. Dokas, "Safety bounds in human robot interaction: A survey," *Safety Science*, vol. 127, p. 104 667, 2020, ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2020.104667. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925753520300643.

[4] Z. H. Khan, A. Khalid, and J. Iqbal, "Towards realizing robotic potential in future intelligent food manufacturing systems," *Innovative Food Science & Emerging Technologies*, vol. 48, pp. 11–24, 2018, ISSN: 1466-8564. DOI: https://doi.org/10.1016/j.ifset.2018.05.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1466856417300401.

[5] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, "Safety-critical manipulation for collision-free food preparation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 954–10 961, 2022. DOI: 10.1109/LRA.2022.3192634.

[6] A. Gafer, D. Heymans, D. Prattichizzo, and G. Salvietti, "The quad-spatula gripper: A novel soft-rigid gripper for food handling," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, 2020, pp. 39–45. DOI: 10.1109/RoboSoft48309.2020.9115968.

[7] R. Murai, T. Sakai, H. Kawano, *et al.*, "A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital," in *2012 IEEE/SICE International Symposium on System Integration (SII)*, 2012, pp. 510–516. DOI: 10.1109/SII.2012.6427311.

[8] D. Stoianovici, C. Kim, D. Petrisor, *et al.*, "Mr safe robot, fda clearance, safety and feasibility of prostate biopsy clinical trial," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 115–126, 2017. DOI: 10.1109/TMECH.2016.2618362.

[9] H.-M. Gross, C. Schroeter, S. Mueller, *et al.*, "Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 2430–2437. DOI: 10.1109/IROS.2011.6094770.

[10] M. Manti, A. Pratesi, E. Falotico, M. Cianchetti, and C. Laschi, "Soft assistive robot for personal care of elderly people," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2016, pp. 833–838. DOI: 10.1109/BIOROB.2016.7523731.

[11] J. Leitner, "Multi-robot cooperation in space: A survey," in *2009 Advanced Technologies for Enhanced Quality of Life*, 2009, pp. 144–151. DOI: 10.1109/AT-EQUAL.2009.37.

[12] K. Yoshida, "Achievements in space robotics," *IEEE Robotics & Automation Magazine*, vol. 16, no. 4, pp. 20–28, 2009. DOI: 10.1109/MRA.2009.934818.

[13] M. Panzirsch, H. Singh, T. Krüger, C. Ott, and A. Albu-Schäffer, "Safe interactions and kinesthetic feedback in high performance earth-to-moon teleoperation," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–10. DOI: 10.1109/AERO47225.2020.9172665.

[14] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1243–1274, 2019. DOI: 10.1109/COMST.2018.2888904.

[15] B. R. Kiran, I. Sobh, V. Talpaert, *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022. DOI: 10.1109/TITS.2021.3054625.

[16] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017. DOI: 10.1109/TIV.2017.2749181.

[17] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020. DOI: 10.1109/ACCESS.2020.2983149.

[18] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017. DOI: 10.1109/TAC.2016.2638961.

[19] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, K. Chatterjee and T. A. Henzinger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–106.

[20] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32.

[21] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013, ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.02.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109813000678.

[22] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020. DOI: 10.1146/annurev-control-090419-075625. eprint: https://doi.org/10.1146/annurev-control-090419-075625. [Online]. Available: https://doi.org/10.1146/annurev-control-090419-075625.

[23] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066. DOI: 10.1109/CDC.2018.8619572.

[24] Y. Li, C. Lv, J. Zhang, Y. Zhang, and W. Ma, "High-precision modulation of a safety-critical cyber-physical system: Control synthesis and experimental validation," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2599–2608, 2018. DOI: 10.1109/TMECH.2018.2833542.

[25] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 6059–6066. DOI: 10.1109/CDC.2018.8619572.

[26] P. Nuzzo, J. B. Finn, A. Iannopollo, and A. L. Sangiovanni-Vincentelli, "Contract-based design of control protocols for safety-critical cyber-physical systems," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–4. DOI: 10.7873/DATE.2014.072.

[27] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Agha-mohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4093–4099. DOI: 10.1109/ICRA40945.2020.9196709.

[28] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman, "Cautious adaptation for reinforcement learning in safety-critical settings," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 2020, pp. 11 055–11 065. [Online]. Available: https://proceedings.mlr.press/v119/zhang20e.html.

[29] W. Luo, W. Sun, and A. Kapoor, "Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 372–383. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/03793ef7d06ffd63d34ade9d091f1ced-Paper.pdf.

[30] A. Dixit, M. Ahmadi, and J. W. Burdick, "Risk-sensitive motion planning using entropic value-at-risk," in *2021 European Control Conference (ECC)*, 2021, pp. 1726–1732. DOI: 10.23919/ECC54610.2021.9655104.

[31] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *2021 European Control Conference (ECC)*, 2021, pp. 1556–1563. DOI: 10.23919/ECC54610.2021.9654841.

[32] J. Yoo, E. Jee, and S. Cha, "Formal modeling and verification of safety-critical software," *IEEE Software*, vol. 26, no. 3, pp. 42–49, 2009. DOI: 10.1109/MS.2009.67.

[33] S. Mitra, T. Wongpiromsarn, and R. M. Murray, "Verifying cyber-physical interactions in safety-critical systems," *IEEE Security & Privacy*, vol. 11, no. 4, pp. 28–37, 2013. DOI: 10.1109/MSP.2013.77.

[34] D. Saxena and V. Raychoudhury, "Design and verification of an ndn-based safety-critical application: A case study with smart healthcare," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 991–1005, 2019. DOI: 10.1109/TSMC.2017.2723843.

[35] E. M. Clarke, "Model checking," in *Foundations of Software Technology and Theoretical Computer Science*, S. Ramesh and G. Sivakumar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 54–56, ISBN: 978-3-540-69659-9.

[36] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[37] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Comput. Surv.*, vol. 52, no. 5, 2019, ISSN: 0360-0300. DOI: 10.1145/3342355. [Online]. Available: https://doi.org/10.1145/3342355.

[38] A. Corso, R. Moss, M. Koren, R. Lee, and M. Kochenderfer, "A survey of algorithms for black-box safety validation of cyber-physical systems," *Journal of Artificial Intelligence Research*, vol. 72, pp. 377–428, 2021.

[39] T. Dreossi, D. J. Fremont, S. Ghosh, *et al.*, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Computer Aided Verification*, I. Dillig and S. Tasiran, Eds., Cham: Springer International Publishing, 2019, pp. 432–442.

[40] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems.," in *CAV*, Springer, vol. 10, 2010, pp. 167–170.

[41] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 254–257, ISBN: 978-3-642-19835-9.

[42] H. A. Taha, *Operations research: an introduction*. Pearson/Prentice Hall Upper Saddle River, NJ, USA, 2011, vol. 790.

[43] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," in *Robotics Research*, N. M. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds., Cham: Springer International Publishing, 2020, pp. 75–84.

[44] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical finance*, vol. 9, no. 3, pp. 203–228, 1999.

[45] BCBS, *Fundamental review of the trading book: A revised market risk framework*, 2013.

[46] S. Höfer, K. Bekris, A. Handa, *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021. DOI: 10.1109/TASE.2021.3064065.

[47] C. Doersch and A. Zisserman, "Sim2real transfer learning for 3d human pose estimation: Motion to the rescue," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[48] M. Kaspar, J. D. Muñoz Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4383–4388. DOI: 10.1109/IROS45743.2020.9341260.

[49] M. Maiworm, T. Bäthge, and R. Findeisen, "Scenario-based model predictive control: Recursive feasibility and stability," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 50–56, 2015.

[50] W. Esterhuizen, K. Worthmann, and S. Streif, "Recursive feasibility of continuous-time model predictive control without stabilising constraints," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 265–270, 2020.

[51] X. Fang and W.-H. Chen, "Model predictive control with preview: Recursive feasibility and stability," *IEEE Control Systems Letters*, vol. 6, pp. 2647–2652, 2022.

[52] S. Yu, X. Li, H. Chen, and F. Allgöwer, "Nonlinear model predictive control for path following problems," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 8, pp. 1168–1182, 2015.

[53] S. Lucia, S. Subramanian, D. Limon, and S. Engell, "Stability properties of multi-stage nonlinear model predictive control," *Systems & Control Letters*, vol. 143, p. 104 743, 2020.

[54] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[55] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *Machine Learning*, pp. 1–35, 2021.

[56] N. Jaquier, V. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo, "Geometry-aware bayesian optimization in robotics using riemannian matérn kernels," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, 2022, pp. 794–805. [Online]. Available: https://proceedings.mlr.press/v164/jaquier22a.html.

[57] M. Tucker, E. Novoseller, C. Kann, *et al.*, "Preference-based learning for exoskeleton gait optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 2351–2357. DOI: 10.1109/ICRA40945.2020.9196661.

[58] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM review*, vol. 44, no. 4, pp. 525–597, 2002.

[59] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation.," in *Robotics: Science and Systems*, 2017.

[60] M. Ahmadi, A. Singletary, J. W. Burdick, and A. D. Ames, "Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 4797–4803.

[61] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009, ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2009.06.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397509004149.

[62] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[63] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2018.

[64] L. Lindemann and D. V. Dimarogonas, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1916–1928, 2020.

[65] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of model checking*. Springer, 2018, vol. 10.

[66] J.-P. Katoen, "The probabilistic model checking landscape," in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 2016, pp. 31–45.

[67] A. Platzer and J.-D. Quesel, "Keymaera: A hybrid theorem prover for hybrid systems (system description)," in *International Joint Conference on Automated Reasoning*, Springer, 2008, pp. 171–178.

[68] M. Fitting, *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[69] J. M. Schumann, *Automated theorem proving in software engineering*. Springer Science & Business Media, 2001.

[70] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, 2016.

[71] A. Pnueli, "The temporal semantics of concurrent programs," *Theoretical computer science*, vol. 13, no. 1, pp. 45–60, 1981.

[72] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Springer, 2004, pp. 152–166.

[73] L. Bortolussi and L. Nenzi, "Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic," in *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, 2014, pp. 66–73.

[74] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

[75] O. Maler and D. Ničković, "Monitoring properties of analog and mixed-signal circuits," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 3, pp. 247–268, 2013.

[76] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *International Conference on Computer Safety, Reliability, and Security*, Springer, 2019, pp. 69–84.

[77] D. Ničković, O. Lebeltel, O. Maler, T. Ferrère, and D. Ulus, "Amt 2.0: Qualitative and quantitative trace analysis with extended signal temporal logic," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2018, pp. 303–319.

[78] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing s-taliro as an automatic test generation framework for autonomous vehicles," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2016, pp. 1470–1475.

[79] G. E. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel, "Verification of automotive control applications using s-taliro," in *2012 American Control Conference (ACC)*, IEEE, 2012, pp. 3567–3572.

[80] B. Hoxha, H. Abbas, and G. Fainekos, "Using s-taliro on industrial size automotive models," *Proc. of Applied Verification for Continuous and Hybrid Systems*, 2014.

[81] D. J. Fremont, J. Chiu, D. D. Margineantu, D. Osipychev, and S. A. Seshia, "Formal analysis and redesign of a neural network-based aircraft taxiing system with verifai," *arXiv preprint arXiv:2005.07173*, 2020.

[82] D. J. Fremont, E. Kim, T. Dreossi, *et al.*, "Scenic: A language for scenario specification and data generation," *arXiv preprint arXiv:2010.06580*, 2020.

[83] H.-D. Tran, X. Yang, D. M. Lopez, *et al.*, "Nnv: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," *arXiv preprint arXiv:2004.05519*, 2020.

[84] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, "Probabilistic temporal logic falsification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2s, pp. 1–30, 2013.

[85] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods Symposium*, Springer, 2015, pp. 127–142.

[86] M. Althoff and S. Lutz, "Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1326–1333.

[87] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2352–2358.

[88] D. J. Fremont, E. Kim, Y. V. Pant, *et al.*, "Formal scenario-based testing of autonomous vehicles: From simulation to the real world," *arXiv preprint arXiv:2003.07739*, 2020.

[89] S. Ghosh, F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor, "Verifying controllers against adversarial examples with bayesian optimization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7306–7313.

[90] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 318–328.

[91] T. A. Wheeler and M. J. Kochenderfer, "Critical factor graph situation clusters for accelerated automotive safety validation," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2133–2139.

[92] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of test cases for automated driving systems using bayesian optimization," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1961–1967.

[93] L. Lindemann and D. V. Dimarogonas, "Robust motion planning employing signal temporal logic," in *2017 American Control Conference (ACC)*, IEEE, 2017, pp. 2950–2955.

[94] L. Lindemann and D. V. Dimarogonas, "Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks," in *2019 18th European Control Conference (ECC)*, IEEE, 2019, pp. 89–94.

[95] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," 2016.

[96] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 81–87.

[97] I. Haghighi, N. Mehdipour, E. Bartocci, and C. Belta, "Control from signal temporal logic specifications with smooth cumulative quantitative semantics," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, IEEE, 2019, pp. 4361–4366.

[98] K. Bae and J. Lee, "Bounded model checking of signal temporal logic properties using syntactic separation," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[99] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[100] L. Lindemann and D. V. Dimarogonas, "Efficient automata-based planning and control under spatio-temporal logic specifications," in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 4707–4714.

[101] P. Akella, A. Badithela, R. M. Murray, and A. D. Ames, "Lipschitz Continuity of Signal Temporal Logic Robustness Measures: Synthesizing Control Barrier Functions from One Expert Demonstration," *62nd Conference on Decisions and Control (Submitted)*, Apr. 2023. DOI: 10.48550/arXiv.2304.03849. arXiv: 2304.03849 [eess.SY],

[102] C. Madsen, P. Vaidyanathan, S. Sadraddini, *et al.*, "Metrics for signal temporal logic formulae," in *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 1542–1547.

[103] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th international conference on hybrid systems: Computation and control*, 2015, pp. 239–248.

[104] K. Fan, "Minimax theorems," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 39, no. 1, p. 42, 1953.

[105] P. Akella, U. Rosolia, and A. D. Ames, "Learning Performance Bounds for Safety-Critical Systems," *arXiv e-prints*, arXiv:2109.04026, arXiv:2109.04026, Sep. 2021. DOI: 10.48550/arXiv.2109.04026. arXiv: 2109.04026 [eess.SY],

[106] J. R. Norris and J. R. Norris, *Markov chains*. Cambridge university press, 1998.

[107] A. Badithela and R. M. Murray, "Synthesis of static test environments for observing sequence-like behaviors in autonomous systems," *arXiv preprint arXiv:2108.05911*, 2021.

[108] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[109] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[110] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[111] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[112] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: An overview," in *Proceedings of 1995 34th IEEE conference on decision and control*, IEEE, vol. 1, 1995, pp. 560–564.

[113] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st. USA: John Wiley & Sons, Inc., 1994, ISBN: 0471619779.

[114] G. E. Monahan, "State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms," *Management science*, vol. 28, no. 1, pp. 1–16, 1982.

[115] S. Bhattacharya, S. Badyal, T. Wheeler, S. Gil, and D. Bertsekas, "Reinforcement learning for pomdp: Partitioned rollout and policy iteration with application to autonomous sequential repair problems," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3967–3974, 2020.

[116] L. T. Dung, T. Komeda, and M. Takagi, "Reinforcement learning for pomdp using state classification," *Applied Artificial Intelligence*, vol. 22, no. 7-8, pp. 761–779, 2008.

[117] S. Png and J. Pineau, "Bayesian reinforcement learning for pomdp-based dialogue systems," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 2156–2159.

[118] S. Singh, Y. Chow, A. Majumdar, and M. Pavone, "A framework for time-consistent, risk-sensitive model predictive control: Theory and algorithms," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2905–2912, 2018.

[119] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[120] M. Ahmadi, A. Dixit, J. W. Burdick, and A. D. Ames, "Risk-averse stochastic shortest path planning," *arXiv preprint arXiv:2103.14727*, 2021.

[121] M. Heger, "Consideration of risk in reinforcement learning," in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 105–111.

[122] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[123] O. Mihatsch and R. Neuneier, "Risk-sensitive reinforcement learning," *Machine learning*, vol. 49, no. 2, pp. 267–290, 2002.

[124] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[125] W. Korb, D. Engel, R. Boesecke, *et al.*, "Risk analysis for a reliable and safe surgical robot system," in *International Congress Series*, Elsevier, vol. 1256, 2003, pp. 766–770.

[126] F. Vicentini, M. Askarpour, M. G. Rossi, and D. Mandrioli, "Safety assessment of collaborative robotics through automated formal verification," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 42–61, 2019.

[127] R. Inam, K. Raizer, A. Hata, *et al.*, "Risk assessment for human-robot collaboration in an automated warehouse scenario," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, vol. 1, 2018, pp. 743–751.

[128] E. ISO, "10218: Robots and robotic devices-safety requirements for industrial robots-part 1: Robots," *ISO: Geneve, Switzerland*, 2011.

[129] I. ISO, "10218-2: 2011: Robots and robotic devices–safety requirements for industrial robots–part 2: Robot systems and integration," *Geneva, Switzerland: International Organization for Standardization*, vol. 3, 2011.

[130] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer, "A survey of algorithms for black-box safety validation," *arXiv e-prints*, arXiv–2005, 2020.

[131] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabhu, "Testing cyber-physical systems through bayesian optimization," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–18, 2017.

[132] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018.

[133] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 163–168.

[134] M. Koren and M. J. Kochenderfer, "Adaptive stress testing without domain heuristics using go-explore," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 1–6.

[135] P. Thomas and E. Learned-Miller, "Concentration inequalities for conditional value at risk," in *International Conference on Machine Learning*, PMLR, 2019, pp. 6225–6233.

[136] D. B. Brown, "Large deviations bounds for estimating conditional value-at-risk," *Operations Research Letters*, vol. 35, no. 6, pp. 722–730, 2007.

[137] Z. Mhammedi, B. Guedj, and R. C. Williamson, "Pac-bayesian bound for the conditional value at risk," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 919–17 930, 2020.

[138] A. Kagrecha, J. Nair, and K. Jagannathan, "Distribution oblivious, risk-aware algorithms for multi-armed bandits with unbounded rewards," *arXiv preprint arXiv:1906.00569*, 2019.

[139] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, 2008.

[140] A. Ahmadi-Javid, "Entropic value-at-risk: A new coherent risk measure," *Journal of Optimization Theory and Applications*, vol. 155, no. 3, pp. 1105–1123, 2012.

[141] W. Fenchel, "On conjugate convex functions," in *Traces and Emergence of Nonlinear Programming*, Springer, 2014, pp. 125–129.

[142] M. M. Flood, "The traveling-salesman problem," *Operations research*, vol. 4, no. 1, pp. 61–75, 1956.

[143] L. A. Rastrigin, "Systems of extremal control," *Nauka*, 1974.

[144] S. Wilson, P. Glotfelter, L. Wang, *et al.*, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[145] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[146] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[147] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. V. Dimarogonas, "Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 858–865, 2019.

[148] A. Puranic, J. Deshmukh, and S. Nikolaidis, "Learning from demonstrations using signal temporal logic," in *Conference on Robot Learning*, PMLR, 2021, pp. 2228–2242.

[149] P. Akella, M. Ahmadi, and A. D. Ames, "A Scenario Approach to Risk-Aware Safety-Critical System Verification," *arXiv e-prints*, arXiv:2203.02595, Mar. 2022. DOI: 10.48550/arXiv.2203.02595. arXiv: 2203.02595 [eess.SY],

[150] S. Wilson, P. Glotfelter, L. Wang, *et al.*, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020. DOI: 10.1109/MCS.2019.2949973.

[151] A. K. Tangirala, *Principles of system identification: theory and practice*. Crc Press, 2018.

[152] E. A. Morelli and V. Klein, *Aircraft system identification: theory and practice*. Sunflyte Enterprises Williamsburg, VA, 2016, vol. 2.

[153] L. Ljung, "System identification," in *Signal analysis and prediction*, Springer, 1998, pp. 163–173.

[154] K. J. Keesman and K. J. Keesman, *System identification: an introduction*. Springer, 2011, vol. 2.

[155] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Pub., 2006, ISBN: 9781575242835. [Online]. Available: `https://books.google.com/books?id=hgFVAAAAYAAJ`.

[156] T. Koh and E. Powers, "Second-order volterra filtering and its application to nonlinear system identification," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 33, no. 6, pp. 1445–1455, 1985.

[157] S. A. Billings, *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[158] K. Rodriguez-Vazquez and P. J. Fleming, "A genetic programming/narmax approach to nonlinear system identification," in *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, IET, 1997, pp. 409–414.

[159] S. Chen, X. Wang, and C. J. Harris, "Narx-based nonlinear system identification using orthogonal least squares basis hunting," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 1, pp. 78–84, 2007.

[160] G. Garofalo, C. Ott, and A. Albu-Schäffer, "Walking control of fully actuated robots based on the bipedal slip model," in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 1456–1463.

[161] M. Thieffry, A. Kruszewski, C. Duriez, and T.-M. Guerra, "Control design for soft robots based on reduced-order model," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 25–32, 2018.

[162] X. Xiong and A. D. Ames, "Bipedal hopping: Reduced-order model embedding via optimization-based control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3821–3828.

[163] C.-Y. Su, Y. Stepanenko, and A. A. Goldenberg, "Reduced order model and robust control architecture for mechanical systems with nonholonomic pfaffian constraints," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 3, pp. 307–313, 1999.

[164] S. Kolathaya and A. D. Ames, "Input-to-state safety with control barrier functions," *IEEE control systems letters*, vol. 3, no. 1, pp. 108–113, 2018.

[165] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 708–717.

[166] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A survey on dual control," *Annual Reviews in Control*, vol. 45, pp. 107–117, 2018.

[167] R. Soloperto, M. A. Müller, S. Trimpe, and F. Allgöwer, "Learning-based robust model predictive control with state-dependent uncertainty," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 442–447, 2018.

[168] J. Schneider, "Exploiting model uncertainty estimates for safe dynamic control learning," *Advances in neural information processing systems*, vol. 9, 1996.

[169] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[170] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE control systems magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[171] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[172] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 8352–8358.

[173] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[174] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using rrt* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

[175] B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung, "Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," *IEEE robotics and automation letters*, vol. 5, no. 3, pp. 4249–4256, 2020.

[176] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[177] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 1478–1483.

[178] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[179] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[180] E. D. Sontag and Y. Wang, "On characterizations of the input-to-state stability property," *Systems & Control Letters*, vol. 24, no. 5, pp. 351–359, 1995.

[181] J. P. Hespanha, D. Liberzon, and A. R. Teel, "Lyapunov conditions for input-to-state stability of impulsive systems," *Automatica*, vol. 44, no. 11, pp. 2735–2744, 2008.

[182] W. Ubellacker and A. D. Ames, "Robust locomotion on legged robots through planning on motion primitive graphs," in *2023 IEEE International Conference on Robotics and Automation (ICRA), submitted*, 2023.

[183] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[184] S. R. Chowdhury and A. Gopalan, "On kernelized multi-armed bandits," in *International Conference on Machine Learning*, PMLR, 2017, pp. 844–853.

[185] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[186] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4709–4715.

[187] T. Johannink, S. Bahl, A. Nair, *et al.*, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 6023–6029.

[188] A. Schperberg, Y. Tanaka, F. Xu, M. Menner, and D. Hong, "Real-to-sim: Deep learning with auto-tuning to predict residual errors using sparse data," *arXiv preprint arXiv:2209.03210*, 2022.

[189] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[190] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[191] M. O'Connell, G. Shi, X. Shi, *et al.*, "Neural-fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, eabm6597, 2022. DOI: 10.1126/scirobotics.abm6597.

[192] C. Folkestad, S. X. Wei, and J. W. Burdick, "Koopnet: Joint learning of koopman bilinear models and function dictionaries with application to quadrotor trajectory tracking," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 1344–1350.

[193] *Video*. [Online]. Available: `https://youtu.be/4i2GNU8ahSU`.

[194] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003.