Online Adaptation for Safe Control of Constrained Dynamical Systems

by

Hardik Parwana

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in the University of Michigan
2025

Doctoral Committee:

Professor Dimitra Panagou, Chair
Professor Necmiye Ozay
Professor Vasileios Tzoumas
Professor Ram Vasudevan

Hardik Parwana
hardiksp@umich.edu
ORCID iD: 0000-0001-5981-2108

© Hardik Parwana 2025

DEDICATION

Dedicated to my friends and family, in the broadest sense of the words

ACKNOWLEDGEMENTS

This dissertation is the culmination of the single longest journey of my life. Both personally and professionally, countless experiences have shaped me along the way, and I am deeply grateful to those whose kindness and support made this achievement possible.

The person who has followed my journey most closely is my advisor, Prof. Dimitra Panagou. She was among the first to respond to my PhD applications and has been welcoming ever since. From the very beginning, she set high expectations and never hesitated to point out my shortcomings, which, in turn, helped me grow professionally and evolve into an independent researcher. This growth would not have been possible without her supportive and understanding nature. She demonstrated remarkable flexibility in accommodating my research interests, even while managing three different funding sources, created new networking opportunities, and showed genuine concern not only for me as a researcher but also as a person. I would also like to express my gratitude to Prof. Tzoumas Vasileios, who I have come to know in various capacities—as an instructor, a friend, and, finally, a member of my PhD committee. His supportive and cheerful demeanor has provided me with valuable insights and fresh perspectives. Additionally, I extend my heartfelt thanks to Prof. Necmiye Ozay and Prof. Ram Vasudevan for serving on my PhD committee and for their invaluable feedback, which played a crucial role in shaping my research to its final form.

I also owe my PhD to several funding agencies and collaborations that significantly contributed to the research presented in this dissertation. I am deeply grateful to the Office of Naval Research (ONR) for funding the first three years of my PhD, shaping the main research theme of this work. I would also like to thank General Dynamics (GD) for supporting one semester of my studies. My collaboration with Prof. Aaron Ames and Ryan Cosner was incredibly insightful and played a pivotal role in improving my understanding of control and planning architectures. Additionally, I am immensely grateful to the Toyota Research Institute (TRI) team in Ann Arbor for funding two semesters of my research and offering me two invaluable internship opportunities during the third and fourth summers of my PhD. The mentorship of Dr. Georgios Fainekos, Dr. Bardh Hoxha, Dr. Danil Prokhorov, and Hideki Okamoto was instrumental in achieving several of the results presented in this dissertation. Their insights and long discussion sessions have been vital to advancing my work and exposing me to a wider research world.

I would like to thank my lab members for their invaluable roles as friends, collaborators, and pseudo-advisors throughout my PhD journey. The senior lab members—Dr. Kunal Garg, Dr. James Usevitch, and Dr. Vishnu Chipade—played a pivotal role in helping me navigate a new environment and always made me feel welcome, including inviting me to their memorable home gatherings. Dr. Joseph Breedon and Dr. Mitchell Black have been a huge source of inspiration, demonstrating how to excel in research while maintaining an incredible work-life balance. Devansh Agrawal and Alia Gilbert started their PhD journeys alongside me, and as we faced the challenge of being the first batch of PhD students to begin during the COVID-19 pandemic, I could not have asked for better lab mates and friends. Much of the work presented here would not have been possible without the long and late-night discussions with Devansh Agrawal, The junior students—Kaleb Ben Naveed, Taekyoung Kim (TK), Haejoon Lee (Joon), and Andre Lim—have brought a new energy to the lab, making it a livelier and more enjoyable place to work. They also provided opportunities for intellectual discussions and allowed me to step into a mentoring role, which I deeply appreciate. Finally, Panagiotis Rousseas, Ruiyang Wang, and Albus A. Fang, though involved for shorter durations, made lasting contributions through our collaborations and friendships.

I would also like to extend a very warm thanks to my friends and family — terms that are synonymous to me - for their constant encouragement and emotional support which helped me navigate through some of the hardest times. I would like to thank my mother Summi Parwana, aunt Mamta Chadha, and grandparents Khazan Chand Chadha and Vinod Bala Chadha for their love and support. Laura Kershaw, Brian Lai, Shashank Verma, and our lovely pet Jester have been the longest constants in my PhD and together with Devansh Agrawal, they constitute my most cherished family group - infusing my life with as much fun and wisdom as my academic endeavors. Youngsun Wi has been my greatest source of support and strength through her constant expression of faith in me, often acting as a real-time controller. Tanner Liu (Tanner), Mark Van der Merwe, and Nibarkavi (Niba) Naresh have played a similar role, always stepping in as my backup controllers with their reassuring presence. Michelle Ji and Samanta Rodriguez made up my most cherished small friend group and supported me greatly during the second year of PhD. My housemates over the years—Joseph Lee, Angelene Ku, Heming Huang, Xili Yi, and Zihan Wu—and our shared pets Luna, Jester, Shiru, Kong, and Tipi have all played instrumental roles in making Ann Arbor, or wherever I meet them again in the future, my home. I would also like to thank Kaleb Ben Naveed, Susan Zhao and Galen for their warm friendships in the last two years of PhD. I would like to extend my heartfelt gratitude to Ozgun Kilic for her caring friendship during the PhD defense phase, which helped me navigate that critical time with confidence.

I would also like to thank Melody Zhang for being a post-defense breather, offering positivity as I prepare to take on the challenges of new endeavors. While it may be impossible to fully acknowledge everyone who contributed to my journey, I would like to extend my heartfelt gratitude to Zirui Xu, Jingyu Song, Noah Curran, Naman Shah, Shuo Yang, Ozgur Ozmen, Kehan Long, Victoria Marie Tuck, Yan Miao, Farhad Shabani, Scott Erickson, Luis Angel, Yixiong Jing, Kai Shen, and Yifei Ma, for their friendships and the unforgettable experiences we shared. Finally, I would like to thank my old friends Shivendra Singh, Sandesh Dubey, Akash Sahu, Jay Patrikar, Ashutosh Kumar, Shubham Yadav, and Abhinav Jain for their unwavering support over the years.

Apart from research, I have invested more in personal growth than ever before. I would like to thank Title Boxing Gym for providing an energetic and motivating environment and the Art Corner Studio for offering a peaceful respite during chaotic times. I feel honored to have been part of the Robotics Department during its formative years. The Robotics Graduate Student Council (RGSC) and the entire Robotics staff, especially Denise Edmund and Damen Provost, have been instrumental in shaping the department into what it is today—a welcoming and supportive community. Finally, I extend my gratitude to the Graduate Employee Organization (GEO) for their dedicated efforts toward improving the campus community. My experiences with all these groups have been incredibly enriching and transformative.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES ix

LIST OF TABLES xiv

LIST OF APPENDICES xv

ABSTRACT xvi

CHAPTER

1 Introduction 1

1.1 Motivation 1
1.2 Statement of Contributions 2
1.3 Literature Review 6
1.3.1 Review of Safe Control and Planning 6
1.3.2 Multi-agent and Social Navigation 8
1.3.3 Auto-Controller Tuning 10
1.3.4 State Prediction under Uncertainties 12
1.4 Outline of the Thesis 13

2 Preliminaries 16

2.1 Notations 16
2.2 System Description 17
2.3 Optimization 18
2.4 Barrier Functions and Set Invariance 19
2.4.1 Set Invariance 20
2.5 Control Barrier Functions 20
2.5.1 Controller Description 24
2.6 Probability Distributions 25
2.7 Literature Review of CBFs 26
2.7.1 Current approaches and limitations 28

3 Feasible Space Perspective on CBFs 31

3.1 Literature Review and Statement of Contributions . . . 31
3.2 Problem Formulation . . . 34
3.3 Feasible Volume Control Barrier Function . . . 35
3.3.1 Polytope Volume Barrier Function . . . 35
3.4 An online method to monitor feasibility - Polytope Volume as Candidate CBF 39
3.4.1 Polytope Volume Estimation . . . 39
3.4.2 Smooth Approximation of Polytope Volume . . . 41
3.5 Simulation Results . . . 42
3.5.1 Case Study 1: Valid FS-CBF computation . . . 43
3.5.2 Case Study 2: Improving Feasibility and Sensitivity . . . 43
3.5.3 Case Study 3: AWS Hospital . . . 47
3.6 Conclusion . . . 47

4 Adaptation for CBFs . . . 49

4.1 Literature Review and Statement of Contributions . . . 49
4.1.1 Contributions . . . 51
4.2 Novel Formulation - Rate Tunable CBFs . . . 51
4.2.1 Problem Formulation . . . 52
4.2.2 Motivation for varying class-$\mathcal{K}$ functions . . . 53
4.2.3 Introducing RT-CBFs . . . 55
4.3 Continuous time Pointwise Optimal Adaptation for CBFs . . . 57
4.3.1 Strategy for designing point-wise sufficient condition . . . 58
4.3.2 An algorithm with Zero-Order-Hold implementation . . . 59
4.3.3 Simulation Results . . . 62
4.3.4 Trust-based adaptation for Decentralized Multi-agent Systems . . . 68
4.4 Discrete time Predictor-Corrector Approach to Parameter Optimization . . . 70
4.4.1 System Description . . . 72
4.4.2 Differentiable Convex Programs . . . 75
4.4.3 Problem Statement . . . 77
4.4.4 Dynamical System as a Computational Graph . . . 79
4.4.5 Gradient through Backpropagation . . . 80
4.4.6 SQP based Parameter Update - Back-propagation with Differential Optimization . . . 81
4.4.7 Simulation Results . . . 82
4.4.8 Conclusion and Future Work . . . 86

5 Enter the Uncertainties FORESEE - State Prediction under Uncertainties . . . 90

5.1 Literarture Review and Contributions . . . 90
5.1.1 State Prediction under Uncertainties . . . 90
5.1.2 Problem Statement and Approach . . . 95
5.2 State Distribution Prediction with Unscented Transform . . . 97
5.2.1 Generalized Unscented Transform . . . 97
5.3 Proposed Approach: FORESEE . . . 98
5.3.1 Expansion-Compression Layers . . . 99

5.4 Application to Online Stochastic Optimization - Constraint Aware Gradient Descent . . . 102
5.4.1 Simulation Results . . . 104
5.4.2 Benchmark: Accuracy of State Prediction . . . 104
5.4.3 Application to Stochastic Trajectory Optimization . . . 107
5.4.4 Application to Online Controller Tuning for Leader Follower . . . 111
5.4.5 Benchmark Comparison . . . 114
5.4.6 Experiment Results . . . 117
5.5 Application to MPPI for Hybrid Systems . . . 122
5.5.1 Motivation . . . 123
5.5.2 Related Work . . . 124
5.5.3 Notation and Problem Formulation . . . 125
5.5.4 Preliminaries . . . 127
5.5.5 Methodology: ECUT-MPPI . . . 129
5.5.6 Algorithm . . . 130
5.5.7 Simulation Results . . . 131
5.6 Conclusion . . . 137

6 Graceful Degradation of Constraints . . . 138

6.1 Related Work and Contributions . . . 139
6.2 Preliminaries . . . 141
6.3 Problem Formulation . . . 142
6.4 Methodology . . . 143
6.4.1 Chinneck's algorithm . . . 144
6.4.2 Application of Chinneck's Score to Problem P_0 . . . 144
6.4.3 Proposed Algorithms based on Lagrange Score . . . 145
6.5 Simulation Results . . . 148
6.6 Conclusion . . . 157

7 Conclusion . . . 158

7.1 Conclusions . . . 158
7.2 Future Work . . . 160
7.2.1 Adaptation of CBFs . . . 160
7.2.2 Predictions of state distributions . . . 161
7.2.3 Constraint Selection . . . 162

APPENDICES . . . 163

BIBLIOGRAPHY . . . 179

LIST OF FIGURES

FIGURE

1.1 Instances of autonomous systems exhibiting degraded performance in real-world settings (a) quadrotor exhibiting larger trajectory tracking error in presence of wind disturbances (b) Two mobile robots unable to pass safely through the corridor when operated together with their default safety-critical controllers [59] (c) newspaper clippings of autonomous cars getting into a deadlock in congested environments 3

3.1 Indoor navigation scenario (a) Robot (green circle) navigating in AWS Hospital gazebo environment using proposed CBF controller. (b) Rviz visualization of the global map, global planner's path (green), humans (blue) and the nearest static obstacles (red, found in the way described in Section 3.5.3) used by the robot for collision avoidance. 32

3.2 Feasible space (green) of a single integrator agent with X, Y velocity inputs as it navigates to its goal location with N=3 in (2.30) while avoiding obstacles (black). (a) The path in configuration space, (b), (c), (d) feasible control space visualization at different times. The dotted lines represent hyperplanes that constrain the control input. The 2×2 square is formed by the imposed control input bound. Other hyperplanes are formed as a result of CBF equations in (3.3). A video of animation can be found on thesis website: https://github.com/hardikparwana/PhD-Thesis 36

3.3 (a) Workspace of black robot with illustrative orange obstacles, (b) The state space. In the violet region, CBFs are compatible, i.e., the volume of the polytope $\mathcal{U}_c$ is > 0. The yellow region is the largest control invariant set contained in the violet region. (c) The feasible control space $\mathcal{U}_c$ defined by multiple CBFs at the current state and time. It's volume is given by the area of the green region in this 2D example. 38

3.4 Feasible space (green) defined by hyperplanes H1-H5. H3, H4 are overlapping. Suppose H4, H5 are free to move horizontally. It can be shown that the gradient of the area of the green region exists w.r.t position of H5 but not w.r.t H4. This is because if H4 is moved to the right, it decreases the green area substantially (non-zero right side gradient) whereas if moved to the left, the green area is unaffected. H5 on the other hand, although causes the volume to decrease if moved to the left, due to its intersection at a single point, has a left-side gradient zero. 38

3.5 Compatible space vs refined FS-CBF values at different X, Y and heading $\theta = 45^0$. The color bar represents the numerical values of FS volume and FS-CBF clipped to the interval [0,1]. (a),(b) Case 1 - initiating with three *valid* CBFs. (c)-(f) initiating with three *candidate* CBFs with aggressive (Case 2) and conservative (Case 3) class-$\mathcal{K}$ function parameters for HOCBF. The compatible space (purple) is a subset of the forward invariant set (yellow) as also illustrated in Fig. 3.3. 44

3.6 Case study 1 - humans move towards the right. (a),(b) Paths for the three cases of $k_v = 1, 1.5, 2.5$. Blue and red paths in (a) end when CBF-QP becomes infeasible. (c), (d), (e), (f) Feasible space visualization with and without volume barrier function for $k_v = 1.5$ in (c), (d) and $k_v = 2.5$ in (e), (f). Note when nominal input is outside the feasible space, the volume barrier function maps it to the interior of feasible space whereas CBF-QP maps it to a point on the boundary. 45

4.1 Trajectories with different initial parameter values without adaptation (dotted lines) and with adaptation (solid lines) for initial state $x = -0.5$. Dotted lines end when the underlying problem becomes infeasible. The solutions of the proposed method remain feasible until the end of the simulation horizon (5.5 seconds). 63

4.2 Time-to-failure as a function of the initial location $x_0 \in [-1.95, 3.0]$ of the double integrator agent. The initial velocity is zero. Algorithm 1 increases the time horizon over which the controller is feasible. 64

4.3 Velocity variation for the ACC problem. The dotted pink line ends when CBF-QP becomes infeasible. The leader decelerates for the first 10 seconds and then moves at constant velocity. The ego-agent approaches the boundary of the safe set (see Fig. 4.4(b)) and hence does not have enough bandwidth to steer back towards its desired velocity of $24m/s$. 65

4.4 The variation with time of (a) parameter ν_i^2: ν_i^2 increases as the control input approaches its bounds, and eventually reduces towards the nominal value of 0.7 due to the effect of the $\nu_{1,d}^1$ in (4.24), (b) barrier h_1: The ego agent approaches and maintains the minimum distance w.r.t. the other agent., and (c) normalized control input u: Normalization is done as v/M^2 66

4.5 (a) Environment setup for baseline comparison. Trajectories of non-ego agents are shown from their start position to the final position. The robot (red) navigates to the goal location while avoiding other agents. (b) Control input variation for DI ego-agent under different methods. (c) Time to infeasibility (normalized) as a function of initial X, Y position of the robot in meters. Fixed corresponds to fixed-parameter case. . . . 66

4.6 The timestamp is given by the colormap. Paths with bold colors result from the proposed method. Paths with increased transparency result from the application of CBFs with fixed class-$\mathcal{K}$ function parameter. 70

4.7 Scenario 1: Variation of class-$\mathcal{K}$ function parameters (a) and barrier functions (b) of Robot 1 with time. UC1, UC2 are the uncooperative agents starting from left and right respectively in Fig. 4.6a. Trust-based relaxation allows agents to go closer to the safe set boundary ($h_{ij} = 0$) compared to fixed α case, and hence leads to a less conservative response while still guaranteeing safety. 71

4.8 Plot of trajectories for different values of parameter. Trajectories end at point of infeasibility. The values of a and b shown are normalized by sampling time Δt. 75

4.9 Time to Infeasibility: X and Y axis represent parameter values(normalized by sampling time Δt) for which trajectory is simulated. The color gradation represents the time after which problem becomes infeasible. A value of 100 implies it remains feasible for maximum possible time and value of 0 implies it is infeasible from the start. (a)c=0.3, (b)c=0.7 in Example 4. 76

4.10 Nodes represent functions, specified here by dynamics or QP controller, and edges represent arguments to the function. Thick blue arrows show the flow of information when computing gradient of second reward $r_2 = R(x_2, u_2)$ with respect to θ through backpropagation. 81
4.11 GD iterates: All figures are for c=0.3 in Example 4. Initial state is 0.5 for (a),(b), and 0.1 for (c),(d). Each shows results for two different initialization of θ. (a), (c) show the change in θ. Points on same GD process are connected by line and direction of update marked with arrows. (b), (d) show plots of time to infeasibility with GD iteration number. Parameter values have been clipped to range (0,5) to better visualize infeasible parameters near top and left edges of the box. The values of a and b shown are normalized by Δt. 83
4.12 Comparison of MPC with the proposed method for two different initial parameter values given by blue and green trajectories. Receding horizon(RC) corresponds to gradient descent steps (4.53) applied as robots move in the environment. Fixed-tuned trajectories result from performing 20 iterations of (4.53) before actually moving the robot in the environment. Goal is at (2,2). Start location is at (-0.5,-0.5). 87
4.13 Leader-Follower Problem 88
4.14 Leader-follower simulation comparing different horizons for high-level optimization. (a) shows the scenario considered and (b) shows the rewards received with time. The adaptive parameter case manages to achieve higher rewards. (c) shows the control input variation and (d) shows variation of barrier functions with time. h_1 and h_2 represent minimum and maximum distance barriers respectively and converge to the value corresponding to desired relative location of leader. 89

5.1 The proposed Expansion-Contraction mechanism for predicting the distribution of state x_{t+1} given the distribution of state x_t. In this abstract 2D ($n = 2$) example, UT and GenUT choose 5 ($= 2n + 1$) points to represent a distribution (a) The red particle (sigma point) $\mathcal{S}_{t,i}$ and their weights $w_{t,i}$ represent the distribution of state x_t. (b) Since the uncertainty is state-dependent, each red point $\mathcal{S}_{t,i}$ gives rise to the states in the yellow region, each of which is represented by 5 blue points $\mathcal{S}^j_{t,i}$ and weights $w^j_{t,i}$ in the expansion layer. (c) The overall weight of $\mathcal{S}^j_{t,i}$ is obtained by multiplying the weight of the edge $w^j_{t,i}$ with the weight of the corresponding root node $w_{t,i}$. (d) The 25 points resulting from the expansion layer are then compressed back to five red-blue particles in the compression layer to represent the blue distribution of x_{t+1}. 92
5.2 The predicted state distribution under different methods for dynamics given by (5.16). Blue samples correspond to 50,000 MC particles. Yellow and salmon-colored samples are the sigma points generated using successive Expansion-Compression (EC) operations. Yellow samples employ UT (Alg. 2) and salmon samples employ Generalized UT [82] in its expansion and compression operation. Red ellipse represents the 95% confidence ellipse of the distribution obtained by successive Gaussian approximation. 105
5.3 First four moments with different methods: Monte Carlo (MC), successive Gaussian, Expansion-Compression (EC) with UT, and with Generalized UT (GenUT). The successive Gaussian approximation is only shown for the first two moments as it is identically zero and constant for skewness and kurtosis. The shaded areas show two standard deviation variations in MC runs. 106

5.4 The predicted state distribution under (5.19) for different dynamics. The colors of samples follow the notation in Fig. 5.2. E stands for expansion. 108
5.5 First four moments for different methods as a function of the horizon under (5.19). Colors follow the same notation as in Fig. 5.3. 109
5.6 Time taken by each of methods in a naive Python implementation (without any optimizations): Monte Carlo (MC), Expansion-Compression (EC), and Expansion-only (E). All operations are performed on a single core of CPU. H value denotes the horizon. The time per step for MC and EC is fixed. The time taken per step by E Layer increases with horizon as the number of particles increases. 110
5.7 The predicted trajectories of dynamic unicycle modeled robot. The start location is at (-0.5,-0.5) and the goal is at (2,2). The orange and blue ovals represent the 95% confidence interval (w.r.t mean and covariance) of the sigma point and Monte Carlo states respectively. On the left is the case where constraint satisfaction in enforced in expectation. On the right is the case where constraint satisfaction is enforced with 95% confidence interval. 112
5.8 Responses over time for the system reward, barrier functions and the control input for the cases of bounded and unbounded control input and with nominal (fixed) and adaptive controller parameters. 115
5.9 Quadrotor stabilization results for benchmarking in safe-control-gym. The ellipses in (a) and (c) show the 95% confidence interval Gaussian representing the states predicted by GP-MPC. The black dots are the (chosen) states from which predictions were made. The predicted ellipses originating from these dots are colored according to the colormap corresponding to the timestamp of the black dot. 118
5.10 Path and cost plots for circular and figure-8 reference trajectories. 120
5.11 Online policy optimization for trajectory tracking and collision avoidance of a quadrotor. Default parameter video: https://www.youtube.com/watch?v=D59R0nZvPWE, Proposed method video: https://www.youtube.com/watch?v=vsBRkm59rAA. In the video, the hoop represents the cylindrical obstacle and the reference is not visualized. 122
5.12 The robot's sampled trajectory (green) induces different responses from each of the human's (the non-ego agent in this example) sigma points. The ellipses represent the distribution of predicted human states and the dots inside them represent the sigma points. At $t = 6$, the human perceives the robot for the first time in its field-of-view for 2 out of 3 sigma points. The red and blue points get influenced by robot position at $t = 6$ while the pink point does not - resulting in three different future ellipses (of the same color respectively). The compression operation combines the three hollow ellipses into a single (solid) ellipse to represent human state distribution at $t = 7$. Note that distribution need not be an ellipse and depends on the variant of UT employed. 131
5.13 Simulation results for proposed method in Section 5.5.7.1. (a) The robot performs collision avoidance with humans (red) and obstacles (black). Robot trajectory is similarly shown in green. (b) Cost (5.39a) and stability cost (Q_c only) accumulated with time. (c), (d) Robot's minimum distance to any (c) human and (d) obstacle with time. The solid/dotted lines and shaded areas represent the mean and the 95% confidence interval in variation over 50 runs. 135

5.14 Simulation results for RA-MPPI inspired method in Section 5.5.7.1. (a),(b),(c) K =20 samples. Note from the increasing stability cost that the robot didn't converge to the goal location in the given time. (d), (e), (f) K =100 samples 136

6.1 A sample testing environment, in which each waypoint is shown in blue and the shaded blue regions show disturbances affecting the robot. The robot is required to travel through the series of waypoints in a specified order (as shown in white numbers) and returns to the red final position before T_{final}. 151
6.2 Performance of different algorithms for 9 waypoints. Note that the waypoints 4 in (c), 6 in (d), and 4 in (f), were dropped by the controller as they could not be reached within the prescribed time. The paths just happen to pass through them while the robot aims at the next waypoint. 152
6.3 Trajectories realized under different algorithms for 14 waypoints. 156

A.1 Illustration of the halfspaces induced by the constraint function. $Av \geq b$ represents the halfspace described by (A.11) with $A = \frac{\partial h_{ij}}{\partial x_j}$, $v = \dot{x}_j$, and $b = -\alpha h - \max\left\{\frac{\partial h_{ij}}{\partial x_i}\dot{x}_i\right\}$. The two x_j are possible instances of the actual motion of agent j. $\hat{s}$ is the normal to the hyperplane $Av = b$. $\hat{n}_j$ is the nominal direction of motion for agent j. 166
A.2 Agents j_1 and j_2 contribute the same constraint margin for satisfying the CBF derivative condition (A.9). 168

B.1 The original and filtered signal 175

LIST OF TABLES

TABLE

3.1 Sensitivity analysis of time to incompatibility for small (Scene 1,2) and large (Scene 3) perturbations of the initial state p_x, p_y, v and gains k_p, k_v. Variables are sampled with uniform distribution within the specified [lower bound, upper bound]. 47

6.1 Number of waypoints reached with different algorithms under the single integrator dynamics. 154

6.2 Reward achieved and computation time for different algorithms. rewards for each waypoint are as follows: Case 1: [1,1,1,1,1,2,2,2,2,2,4,4,4,4], Case 2: [4,4,4,4,4,2,2,2,2,2,1,1,1,1], Case 3: [1,2,3,4,1,2,3,4,1,2,3,2,1,2]. 155

B.1 Trajectory parameters for data collection 174

LIST OF APPENDICES

A RT-CBF proofs 163

Proof of analytical solution of QP for Theorem 3 163
A Synthesis Case Study: Designing RT-CBFs For Multi-Robot Systems 164

B FORESEE 171

Approximating chance constraints with robust constraints 171
Approximating the chance constraint 173
Data Collection and Wind Disturbance Learning 174

C Degradation of Constraints 177

Degradation of Constraints 177

ABSTRACT

Advances in sensing modalities and computational power have led to the prospect of a widespread deployment of robots in our society. Central to this objective is developing control and navigation stacks that avoid conservatism, presumed to be measured by a performance metric, while being provably and practically safe. A crucial element that must be accounted for is that controllers, which are typically designed for and tuned in laboratory or highly monitored industrial settings for a specific scenario, may experience a drop in performance and lose their safety guarantees when used elsewhere. It is of paramount importance therefore to import robots with the capability to adapt their controllers online to customize responses to a priori untested environments.

In brief, this dissertation presents (1) tools to adapt any parametric controller using a model-based approach to achieve simultaneous satisfaction of multiple state constraints and enhanced performance; (2) a numerical scheme for predicting future state distributions in systems governed by stochastic dynamics with state-dependent disturbances, which can be utilized in model-predictive approaches; and (3) a method to assist decision-making on dropping (disregarding) constraints when it is not feasible to satisfy all constraints simultaneously.

A significant part of the dissertation focuses on a specific safety-critical control method, called control barrier functions (CBF). The CBF-based controllers have garnered interest in recent years due to their ease of implementation. However, finding a theoretically valid CBF remains a challenge and in practice, they are prone to performance degradation and safety violations, especially when multiple CBFs are imposed together. This dissertation introduces a new notion of CBFs, called Rate-Tunable CBFs, that allows for time-varying parameters in theory and online tuning in practice.

The dissertation starts with an introductory chapter reviewing safety-critical controllers and planners under deterministic and stochastic settings. The second chapter provides a thorough technical review of CBFs as well as concepts from optimization, probability and set invariance relevant to this work. The third and fourth chapters focus almost exclusively on control barrier functions. The third chapter formally introduces the notion of compatibility of multiple CBF constraints and then follows up with a metric - the volume of the feasible solution space of the QP - to quantify distance to the infeasibility of CBF-QP controllers. A

new CBF is designed to prevent the volume from going to zero thereby ensuring the existence of a solution to the CBF-QP controller at all times. The fourth chapter then introduces our notion of Rate-Tunable CBFs that allow the parameters of class-$\mathcal{K}$ function to vary with time. This allows tuning of the response of CBF-based controllers. Two methods, one instantaneously locally optimal and another based on a model-predictive approach employing gradient-descent on parameters are introduced to design the parameter dynamics.

The fifth chapter examines the effects of uncertain dynamics with state-dependent disturbances. A numerical scheme called Expansion-Compression (EC) Layers, based on the Unscented Transform (UT), is proposed to predict future state distributions. The UT is a weighted, particle-based method; the expansion layer increases the number of particles to represent increased uncertainty due to state-dependent disturbances, while the compression layer uses moment-matching to consolidate these particles into fewer, representative points, resulting in a scalable scheme. Applications of the EC-UT are shown in the gradient-based model predictive auto-tuning framework and the model predictive path integral controller, which advances the state of the art in sample efficiency.

The sixth chapter takes a departure from parameter adaptation and imparts the controller the capability to permanently drop a constraint from its optimization problem. Such scenarios are of interest when some low-priority task specifications, imposed as state-input constraints, conflict with the safety or high-priority task constraints and need to be sacrificed. An algorithm is proposed to drop the minimum number of constraints under an additive priority scheme, a problem that is known to be NP-Hard but unexplored in the context of dynamical systems where the optimization plays the role of a controller. Towards this, a Lagrange multiplier-based heuristic that keeps track of active constraints in the past is introduced to form a more informed prior for solving the NP-Hard problem. It is shown empirically that the proposed heuristic outperforms the existing slack variable-based heuristics.

CHAPTER 1

Introduction

This dissertation presents a collection of tools I developed throughout my Ph.D. to address open problems in the control design for safety-critical operations of robots. This field of research is vast and this dissertation focuses only on an aspect of it - online adaptation of controllers to enable robots to self-adjust to previously untested environments. This chapter serves to introduce the motivation for studying such a class of problems, to provide a summary of contributions made herein, to survey the literature, and finally to present an outline of the remainder of the dissertation.

1.1 Motivation

Robot controllers are often evaluated using performance metrics, such as the error in reference tracking accumulated over time while adhering to all constraints. Controllers that achieve the highest performance are referred to as optimal controllers. In practice, we often find robots to use *non-optimal* controllers. This can be attributed to the following two reasons. First, the *optimal* controllers, even if amenable to theoretical formulation, are hampered by existing computational limitations when applied in practice. Second, outside of the laboratory or highly monitored industrial settings such as assembly lines, it is practically infeasible to consider all possible scenarios that the robot may encounter in the manual tuning or, in the context of modern controllers, the training phase. Therefore, supposing that robots are free to modify the parameters of a given controller, there is a dire need for robots to observe, interpret, and adapt their controllers for better performance in the real world.

In this dissertation, as in much of optimal control literature, *performance* is defined as the ability to perform a task that involves optimizing a user-given metric, such as minimizing power consumption or reaching the target in minimum time, while respecting state and input constraints. Robots deployed in the real world are subjected to several constraints that their controller and planner must account for. They can broadly be classified into three types - 1)

physical limitations imposed by their dynamics, sensor modalities, and actuator bounds, 2) constraints pertaining to the safety of themselves, the environment, and the external agents, and finally 3) constraints encoding task specifications such as reaching the goal location within a given time.

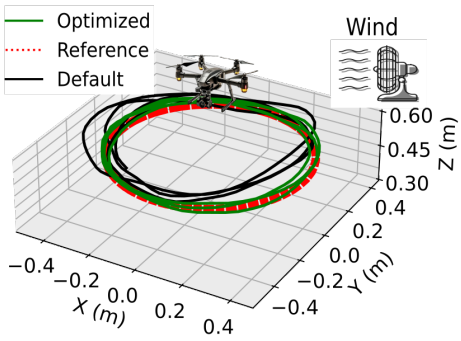
In the face of unforeseen conditions or untested environments, controllers exhibit degraded performance compared to their offline design phase along with violation of constraints as illustrated in Fig.1.1(a), (b). This dissertation aims to impart robots the capability to adapt to new scenarios by designing methods for the online optimization of controller parameters. Specifically, this dissertation utilizes and develops tools related to

- Constrained control: methods like model predictive control and safety filters such as control barrier functions for enforcing constraint satisfaction.
- Multi-agent systems: given that multi-agent systems inherently represent multiple time-varying constraints, this serves as a useful testbed to evaluate algorithms for constrained control.
- Stochastic dynamical systems: for modeling uncertainties to account for dynamics that are unknown during the offline design phase and are learned online from observations.
- Constraint prioritization: similar to task allocation but we bring a novel perspective that allows the robot to drop a constraint from its controller structure completely to ensure completion of higher-priority tasks.

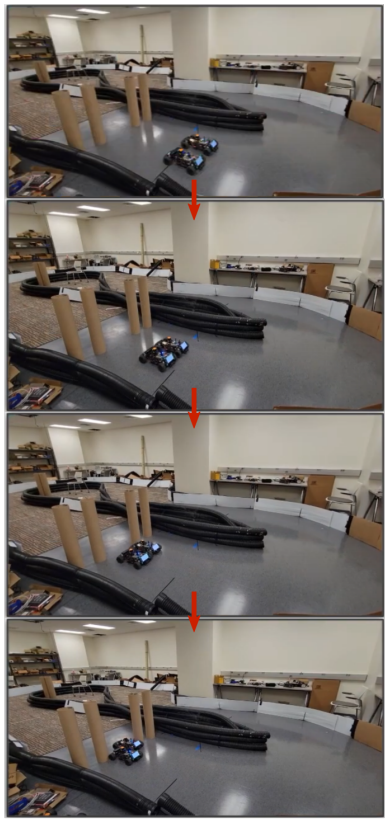
Since safety is the highest priority in practice for ethical reasons, a standard solution that developers often implement to circumvent all of the aforementioned issues is to make the system overly conservative. This conservatism however, although satisfying the original constraints, can lead to further possibly unsafe outcomes. For example, Fig. 1.1(c) shows snippets of newspaper clippings showing autonomous cars stopping at intersections thereby creating a safety hazard by disrupting the traffic. Therefore, a challenge is enabling robots to reduce conservatism and avoid deadlocks by adjusting their controller parameters without compromising safety.

1.2 Statement of Contributions

The progress of this thesis can be divided into three parts. In the first part that spans Chapters 2-4, we focus exclusively on control barrier function (CBF) based controllers. CBFs help convert a state constraint to a coupled state-control input by first monitoring the rate



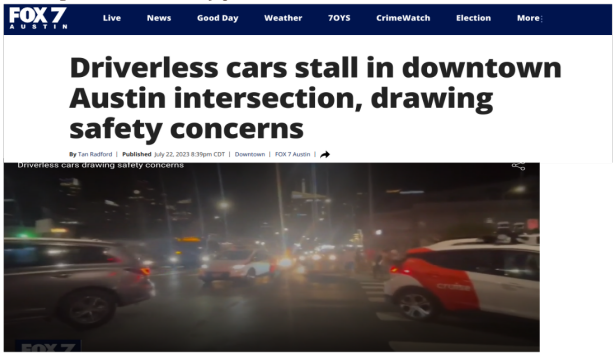
(a) The quadrotor is unable to track the trajectory well under wind disturbances when conrtoller gains tuned for non-windy environments are used. The algorithm proposed in Chapter 4 tunes the controller gains to improve tracking performance while incorporating input bounds.



(b) A controller learned offline for decentralized but cooperating multi-agent system gets stuck in real-world when static obstacles are also added showing that any change outside of training environment easily invalidates the controller.
source: https://www.youtube.com/watch?v=eh30X7NteJw&list=LL&index=6



source: https://www.thedrive.com/news/cruise-robotaxis-cause-austin-street-gridlock-due-to-heavy-pedestrian-traffic



source: https://www.fox7austin.com/news/cruise-driverless-cars-stall-downtown-austin-texas-safety



source: https://www.cnn.com/2023/08/14/business/driverless-cars-san-francisco-cruise/index.html

(c) Newspaper clippings of autonomous cars unable to find a feasible path and getting stuck in a congested environment.

Figure 1.1: Instances of autonomous systems exhibiting degraded performance in real-world settings (a) quadrotor exhibiting larger trajectory tracking error in presence of wind disturbances (b) Two mobile robots unable to pass safely through the corridor when operated together with their default safety-critical controllers [59] (c) newspaper clippings of autonomous cars getting into a deadlock in congested environments

of change of the constraint and then exploiting the dynamics of the system while using principles from set invariance theory. For example, to enforce a distance constraint, it would impose an instantaneous distance-dependent upper bound on the velocity of the robot, effectively slowing it down as it approaches the constraint boundary. One standard way to design a safety-critical controller using CBF constraints is to filter (project) a user-given reference controller to satisfy the constraint. This filtering (projection) operation can be performed by formulating an optimization problem to find the control input closest to the reference input subject to CBF constraints. Under control-affine dynamics, the CBF constraint is rendered linear in the control input and when used in conjunction with quadratic cost function, the optimization becomes a Quadratic Program (QP) leading to, what is popularly known as, the CBF-QP controller. CBFs face several challenges for use in the real world, as detailed in the next section, and we propose new methods to circumvent some of these challenges. CBF conditions are dependent on a parameter and we present two methods to tune these parameters online. The first algorithm proposes an instantaneous update law for these parameters based on the current state. The second proposed algorithm is based on a model predictive approach wherein we first predict future states (up to a time horizon) to compute quantities of interest (performance cost and constraint functions) and compute their gradients to update the parameters in an online gradient descent scheme.

In the second part of the thesis that covers Chapter 5, we delve into uncertain systems whose dynamics are represented as having state-dependent disturbances with arbitrary probability distribution. We aim to apply model-based methods to design controllers for such systems, including but not limited to the model-predictive framework introduced in Chapter 4. However, predicting states under generic stochastic dynamics is known to not be amenable to analytical expressions and several approximate or numerical approaches have been proposed over the years. This is because predicting future state distribution requires integrating functions composed of the dynamics and probability distribution functions (PDF) of the current state, which is intractable. A simple observation exemplifying this is to note that that the PDF of a Gaussian random variable X, one of the simplest distributions, is proportional to e^{-x^2} at $X = x$, which is known to be analytically non-integrable by itself and in composition with other functions in general. While Monte Carlo (MC) is one way to approximate such integrals, it is known to give accurate estimates only when the number of particles approaches a large value. Furthermore, when a finite number of particles are used to predict states over multiple time steps under state-dependent disturbances, the MC accuracy decreases drastically as each particle induces new uncertainties depending on where it is located in the state space. This necessitates an increase in the number of particles across each time step to keep track of the evolving uncertainties resulting in an unscalable

scheme. We propose a way to keep the number of particles fixed across each time step by compressing the increased number of particles based on a moment-matching operation. The idea used here is to choose a small number of particles that have the same moments (mean, covariance, skewness, and so on) as the given large number of particles. We also employ Unscented Transform (UT) instead of Monte Carlo owing to its much higher sample efficiency compared to the latter. We apply the proposed state-prediction scheme in tandem with a variation of the gradient-based model predictive framework proposed in Chapter 4 to tune the parameters of any arbitrary policy (not just CBF-QP). We also show the application of our state-prediction scheme to the Model Predictive Path Integral (MPPI) controller which is a popular gradient-free controller for finding solutions to optimal control problems. We evaluate our proposed MPPI algorithm on a case study for planning paths for a stochastic hybrid dynamical system showing better sample efficiency than state-of-the-art methods.

In Chapter 6, which represents the third and final part of this thesis, we attempt to answer a different but related problem associated with the satisfaction of constraints. We ask the question - what if it is impossible to satisfy all the constraints simultaneously? Which constraints should the controller drop and when? Such a scenario could result when the optimal controller cannot find a solution (that is, the task is impossible to achieve). It could also result when a given controller, even with all its adaptation laws, is unable to find a feasible solution in real time. In this work, we focus on the second scenario. We bring a novel optimization perspective to this problem and given a controller in the form of an optimization problem, we decide which constraints to remove to continue satisfying *higher priority* constraints. Priority can be defined in several ways - absolute, relative, or additive. We elaborate on our approach in Chapter 6 but notably, we observe that unless we are computing an optimal infinite horizon solution, removing constraints permanently is bound to have a different impact than relaxing the constraints (for example by adding a slack variable) in practice. Secondly, while removing constraints from an infeasible optimization problem to render it feasible is a well-known NP-Hard problem in optimization literature, it has not been explored in the context of dynamical systems where the parameters of the optimization problem change with the state of the system. We propose a heuristic to use the correlation between the current-time infeasible problem and past feasible optimization problems to generate a more informed prior for solving the NP-Hard problem. Empirical results show significantly better performance than existing methods to solve the aforementioned NP-Hard problem.

1.3 Literature Review

This section presents several existing methodologies for control and planning under deterministic and stochastic dynamics. Planning and control has a rich history as well as a fast-paced present and the review presented below is by no means close to complete however, is the most insightful to the work presented in this dissertation.

1.3.1 Review of Safe Control and Planning

When it comes to safe motion planning , Artificial Potential Fields [243, 244] (APF) were proposed in the 1980s and quickly became popular for their simplicity. APFs assign a potential value to the state under consideration by summing up the negative potentials assigned to obstacles and a positive potential function assigned to the target state. This method may be subject to local minima however, potential function design procedures to avoid them have also been proposed [271, 171]. The velocity of the agent is then designed to track the negative gradient of the PF. APFs are however unable to account for any arbitrary dynamics of the agent and, in practice, require a separate tracking controller.

Inspired by optimization literature, reciprocal barrier function or logarithmic barrier functions [77, 128, 170] emerged as an alternative way to APFs. The concept is extended to a control barrier function that imposes an instantaneous state-dependent bound on control input so that the closed-loop system satisfies the condition of the reciprocal barrier function [220]. The current prevalent form of CBF though is the zeroing control barrier function [15] that requires the barrier function be to zero [182] instead of infinity on the boundary of the safe set. In practice, zeroing CBFs leads to a well-behaved functions suitable for a numerical implementation. The original CBF was designed for an ideal deterministic system in continuous time. The recent variations include extensions to discrete-time [9], sampled-data [42], hybrid [54, 53, 182], input bounded [10], event-triggered [165, 218] and stochastic dynamical systems [67, 259, 31]. However, designing valid CBFs such that the state-feedback controller results in a closed-loop vector field satisfying the barrier condition is non-trivial. Towards this end, sum-of-squares programming, Hamilton-Jacobi reachability [226], offline learning [184, 203, 138], geometrical analysis [44], and iterative refining procedures [10] have been successfully applied to constructive valid CBFs. However, each of the above methods is dependent on some user-chosen parameters, and their coupling with the nominal controller, which is important, is not considered resulting in non-optimal controllers. Consequently, while they provide an excellent prior, they still require a module for online refinement of CBFs. Furthermore, since the safety filter defined by CBF constraint and the nominal controller may work against each other, CBF-based controllers, like APFs, are also prone to local

minima or deadlocks [55]. CBFs still carry several advantages over APFs and the reader is referred to [202] for additional comparison between them.

APFs and CBFs are also prone to, what is popularly termed as, being *myopic* (meaning short-sightedness). They design control input at time t based only on the state at time t, that is, independent of (future) states at time $\tau > t$. When the objective is to optimize cumulative performance over a time period, such a controller is known to be optimal over time only for unconstrained liner systems in general (LQR control for an informed reader). To solve optimal control problems in real-time, one of the most popular control and (short horizon) kinodynamic planning method is Model Predictive Control [155, 187] (MPC). MPC, when posed as an optimization problem, encodes arbitrary constraints while optimizing a user-defined performance objective. In contrast to most state-feedback control laws developed specifically for stability over the years, MPC, inits scope of implementation, does not require the system to be controllable thus leading to its immense popularity as a model-based control method upon its inception. The two main drawbacks of MPC are practically intractable nonlinear optimization for nonlinear dynamics and constraints and difficulty in proving recursive feasibility unless a controlled invariant set is known a priori. With improvements in computation speed though, MPC as a nonlinear program is getting more tractable, but linearizing the constraints and dynamics to result in a QP controller is a standard approach.

Iterative LQR [132] (iLQR) is a similar iterative procedure for trajectory optimization that linearizes the dynamics and uses the known optimal LQR solution at every time step. Differential dynamic programming [147] (DDP) instead uses a second-order approximation of the cost function. Each iteration of finite horizon iLQR and DDP is based on backward and forward passes to optimize the sequence of control input, as is typically done in LQR. More recently, variations of DDP accommodating hard state-input constraints have also been proposed [258, 201].

Sampling provides an alternative procedure to gradient-based techniques with the potential to work with a broader class of functions besides being differentiable and not being prone to local minima. Planners like rapidly-exploring random trees [124, 125] (RRT) and cross-entropy sampling [194, 118] have consequently received equal attention over the years however were limited to low-dimensional kinematic systems in their nascent years since sampling scales poorly with state-action dimension. Kinodynamic variants [245, 250] have also been proposed. In general, however, expensive planners such as RRT and long-horizon MPC are still implemented in multi-rate navigation architecture requiring a pre-defined tracking controller [143, 11, 193] operating at low-level.

Assessing the cost-benefit of the pros and cons of all the aforementioned controllers and

planners is application-dependent to a large extent. The prevalent landscape is also often reshaped by advances in computing capabilities, including both software and hardware. For example, learned neural network controllers and planners [72, 178] and parallelizable sampling-based methods like model predictive path integral [246] (MPPI) controllers utilize GPUs to provide real-time implementation suitable for a wide variety of tasks.

Despite the advancements, all the methods are limited by the computational requirements affecting the choice of prediction horizon, number of samples, dimension of state and inputs, etc. This necessitates online adaptation and real-time response to unexpected behavior to result in more optimal paths. Note that these shortcomings are only compounded by considerations of noise, partial state observation.

1.3.2 Multi-agent and Social Navigation

Teams of collaborating robots can enable tasks such as payload transportation, surveillance and exploration [189]. The ability to meet specifications such as safety and goal reaching even in the presence of non-cooperative agents is essential to successful operation and mission accomplishment. This dissertation often uses multi-agent systems as a testbed for evaluating algorithms. An account of the history of multi-agent systems is not necessary to understand the progression of this dissertation but it has inspired several research problems that we propose solutions to. Thus, the survey here is presented in the spirit of making this thesis an amalgam of unique research areas that render robots capable of observing, interpreting, and adapting their controllers and planners for better navigation. As such, although the field of multi-agent systems has a rich history, we only delve into topics of interest to us. We focus specifically on non-centralized control design for multi-agent systems incorporating uncooperative as well as adversarial agents.

Control synthesis for multi-agent systems in the presence of non-cooperative agents has also received vivid interest, with many remarkable results [176, 213, 230, 197, 270, 100, 156, 180]. In [197, 213, 176, 230, 270], the authors present resilient algorithms for flocking and active target-tracking applications, respectively. In [180] and [156], the authors design game-theoretic and adaptive control mechanisms to directly reject the effect of adversaries without the need of knowing or detecting their identity. Recently, CBF-based approaches have also been presented for achieving resilience under safety and goal-reaching objectives. In particular, [39] developed multi-agent CBFs for non-cooperative agents. [231] designed a mechanism to counter collision-seeking adversaries using CBF-based control that is robust to worst-case actions by the adversaries. All of the above studies either assume that the identities of adversarial agents is known *a priori*, or they design a robust response without

knowing or detecting their identities; nevertheless, as also evidenced by our simulations, both cases can lead to conservative responses. Moreover, most of these works assume further that each agent knows the exact dynamics of the other agents.

Some of the practical challenges in this dissertation were also motivated specifically by problems in social navigation [86] - the study of robot navigation amongst humans. Two mainstream research directions in social navigation are human motion modeling and planning and control architecture for robots. For the former, several potential field-based models were developed initially however the recent shift is towards data-driven learning. Several deep-learning based trajectory forecasting algorithms [196, 263] have been published for performance evaluation on specific open-source datasets - see [86] for a list of human motion datasets. The trajectory forecastign algorithms along with simulated models can be used to train robot policies for navigation [57]. Generalizability of the learned models to out-of-distribution scenarios remains a challenge nonetheless however in this work, we only focus on the latter.

Social navigation inside buildings adds another layer of the challenge of navigating in narrow spaces. Several existing approaches such as [200, 177, 48] for indoor navigation with humans rely only on fast replanning through a low-level planner based on simplified dynamics for collision avoidance, and do not employ a safety-critical controller. These approaches, although perform well in practice through fast replanning, do not guarantee safety and often make robots conservative. Designing guaranteed safe controllers and planners for this problem remains a challenge with no viable and generalizable solution proposed yet. Among CBF-based approaches, which is of interest in this dissertation, [237] uses CBF-MPC, an MPC controller with constraints defined by CBF derivative conditions, for navigation in crowd, and [223] designs a novel CBF for polytopic obstacles and uses it in the CBF-MPC framework for navigation in corridor environments. From an application perspective, [237] does not consider indoor environments and [223] does not consider dynamic moving agents. While MPC does help with maintaining the feasibility of multiple CBF constraints when imposed together, no guarantees of the recursive feasibility of MPC have been shown to exist in dynamic indoor environments, these controllers are still prone to infeasibility. While enforcing compatibility (simultaneous satisfaction) of multiple CBF constraints remains a challenge, we provide a more thorough literature review in Chapter 3 and also propose a novel way to solve this problem. We evaluate our algorithm on a ROS-Gazebo hospital environment and show better feasibility maintenance over vanilla CBF-QP.

Another notion of interest to the manifold of work presented in this dissertation is *trust*. We share inspiration with work that aims to develop beliefs on the behavior of other agents, and then use them for path planning and control. Deep Reinforcement Learning (RL)

has been used to learn the underlying interactions in multi-agent scenarios [45], specifically for autonomous car navigation, to output a suitable velocity reference to be followed for maneuvers such as lane changing or merging. A few papers explicitly model trust for a multi-agent system [232, 85, 107]. In particular, [232] develops an intent filter for soccer-playing robots by comparing the actual movements to predefined intent templates, and gradually updating beliefs over them. Among the studies focusing on human-robot collaboration, [85] learns a Bayesian Network to predict the trust level by training on records of each robot's performance level and human intervention. [107] defines trust by modeling a driver's perception of vehicle performance in terms of states such as the relative distance and velocity to other vehicles, and then uses a CBF to maintain the trust above the desired threshold. The aforementioned works show the importance of making inferences from observations. However, they were either developed specifically for semi-autonomous and collaborative human-robot systems using offline data sets or when they do consider trust, it is mostly computed with a *model-free* approach, such as RL. The above studies also use the computed trust as a monitoring mechanism rather than being related directly to the low-level controllers that decide the final control input of the agent and ensure safety-critical operation. In Chapter 4, we introduce a notion of CBF that allows changing parameters with time to manipulate the conservatism or aggressiveness of a CBF controller. We present a small case study that defines *trust* based on the observed contribution of another agent to the satisfaction of CBF conditions and then use it to modify the parameters in the CBF controller.

1.3.3 Auto-Controller Tuning

Controllers are traditionally tuned by hand by experienced professionals however manual tuning can still be inefficient especially when the parameter space is high-dimensional. For PID, which is still widely used for linear systems or nonlinear systems with feedback linearization, several techniques such as Zieglar-Nichols [162] and software tools exist [133, 272] for stabilization tuning. However, these methods have limited scope as most real-world systems are not linear, have input and state constraints to account for, and do not employ PID control.

Two classes of automated tuning techniques are model-based and model-free. A methodology commonly applied in both approaches is to define a cost function and perform a search for parameters, either by sampling or gradient descent, that optimize the cost function. The cost function could, for example, enhance stability objectives over a time horizon.

Data-driven model-free methods approximate the gradient by constructing a surrogate model of the cost function using collected data. One of the most popular techniques for

model-free tuning is Bayesian Optimization [112](BO). BO typically uses Gaussian Process regression to build a surrogate model of the cost function from observed data and uses the uncertainty estimates of GP and possibly its gradients to explore and optimize the parameter values. BO has already been applied to tune the parameters of a wide range of controllers [144, 50, 139] such as MPC [20, 208, 94], LQR [144], PID [58, 84], Kalman Filter [209, 60](assuming Kalman gain is a controller parameter to help stabilize state estimate error!). Safe BO additionally attempts to sample parameters that prevent unsafe behavior. For example, SafeOpt [211] was used in [29] to tune quadrotor controller gains. It has additionally shown versatility in an accommodating variety of scenarios such as finding Pareto in front of conflicting objectives, time-varying objective functions, and preference-based tuning. BO is however known to be inefficient in high-dimensional state and parameter spaces [87]. Besides BO, metaheuristics such as Particle swarm optimization (PSO) and ant colony optimization (ACO) have also been used in several works to explore parameter space. Readers are referred to [207] for a comparative study of existing BO methods and metaheuristics for controller tuning. Another popular parameter tuning algorithm is Markov-chain Monte Carlo (MCMC). More recently, [140] uses Metropolis–Hastings algorithm, a MCMC algorithm, to tune quadrotor controller gains for high-speed flights and deep-learning. Finally, all of deep-learning (model-free or model-based) is also arguably tuning parameters of neural networks to output a well performing policy.

In this dissertation, we take a model-based approach to develop a real-time controller tuning algorithm for their sample and computational efficiency. Model-based methods [227, 123, 175] leverage the knowledge of known dynamics to improve performance, often by computing the gradient of the cost function w.r.t controller parameters and applying gradient descent. The cost function is evaluated by rolling out dynamics to predict future states in the simulation Model-based RL algorithms such as PILCO [75, 190] also lie in this domain. Following seminar work in OptNet [18] to differentiate through a QP's solution, a lot of work has been done for differentiating through various types of convex programs [7] (also known as performing sensitivity analysis), thus popularizing the field of differentiable optimization (DO). DO has been used to tune various controllers such as MPC [17, 81], Pontryagin differential programming for optimal control [111, 110], control barrier functions [175, 142, 257](includes this dissertation's work) and LQR [235]. Chapter 4 and 5 also employ DO to tune CBF-QP controllers for deterministic as well as stochastic dynamics in the presented case study. More recently, DiffTune [62, 63, 216] builds up on the aforementioned works and proposes a generic controller auto-tuning framework utilizing auto-differentiation while addressing issues like gradient-descent learning rate tuning.

1.3.4 State Prediction under Uncertainties

In the absence of an accurate system model, the field of stochastic control and planning aims to offer protection against disturbances. The notion of safety therein is also defined to enforce constraint satisfaction in probability, also known as chance constraints. These problems are also researched by names of risk-aware planning and control [103] and scenario-based optimization [145, 51]. Note that while the field of robust control also offers protection against bounded disturbances, in several real-world systems uncertainties are considered to be of a stochastic nature and can be adequately characterized [149]. Some of these sources may be aleatoric, while others are epistemic. The uncertainties, whose sources can be aleatoric or epistemic, thus naturally lead to noisy state estimates from sensors and well-defined probabilistic dynamics models that govern the motion. Therefore, it is more natural and less conservative to consider stochasticity in states than just assuming hard bound without any structure within.

Extensions of MPC [149, 150, 41], DDP [222], CBFs [67, 259, 31], and MPPI [262] to stochastic dynamics settings with the new notion of risk have appeared in the literature. Two of the challenges to these are 1) the notion of risk to be used, for example, based on the central confidence interval of Gaussian or conditional value at risk [192], and 2) tractable representation and time propagation of state distribution. In this dissertation, we only consider the second, that is, the representation of and propagation of state distribution across multiple time steps.

One of the standard approaches is to approximate the state distribution as Gaussians and propagate it through a linearized dynamics function with additive white noise [114]. This problem accepts analytical expressions and is also popularly used in Extended Kalman Filter [188] (EKF). Another standard approach is to employ Monte Carlo sampling [262] or scenario-based optimization [51] and optimize control input to satisfy constraints across all samples or scenarios with user-imposed probability [149]. The former is efficient but induces large approximation errors whereas the latter has the potential to be precise in theory but computationally expensive in practice.

Several studies have either proposed intermediate solutions in terms of accuracy and computational efficiency. In general, a Gaussian passed through a nonlinear dynamics function does not accept analytical expression however in the special case of dynamics given by Gaussian Process (GP) utilizing a Gaussian kernel, Practical Inference for Learning and Control (PILCO) [75] derived explicit analytical expressions. PILCO proved to be a seminal model-based reinforcement learning (RL) algorithm achieving better performance than model-free methods at the time while being more sample-efficient, learning to stabilize a cart-pole with only 18 seconds of interaction with reals-system and having no prior dynamics knowledge.

PILCO's application nonetheless only holds for Gaussian kernel in GP dynamics. The analytical expressions proposed by PILCO have been adapted for stochastic MPC and trajectory optimization [115], and stochastic differential dynamic programming [222, 168]. While still widely employed, PILCO's update, first only applies to a specific kernel in GP, and second does not apply to any other output distribution besides Gaussian. Some works have tried to alleviate this problem by maintaining higher-order moments, or by approximating distributions with more generic parametric representations. [205] and [13] employ Monte Carlo (MC) to estimate future rewards and their gradients. The authors in [166] used Unscented Transform (UT) to propagate uncertainty in the presence of Gaussian noise and improve an initially feasible trajectory with DDP. The approach in [236] represented distributions as a mixture of Gaussians, maintaining coefficients for basis functions, and used the Gaussian quadrature rule to propagate the coefficients over multiple time steps. Chapter 5 contributed to this growing literature and proposed a new scheme based on Unscented Transform for models with state-dependent uncertainty models. UT is more sample efficient than MC and while previous UT approaches were designed for state-independent noise or only considered the mean of GP's output [240] (ignoring the covariance outputted by GP), Chapter 4 explicitly considers the evolution of uncertainty of each involved particle used to approximate the state's probability distribution with UT.

1.4 Outline of the Thesis

In what follows, an outline of the remaining chapters is provided along with a brief overview of the contributions made therein. Publications that resulted from these chapters are also listed.

- Chapter 2 presents an overview of optimization, control barrier functions, and probability theory that will be prerequisites for understanding the concepts that unfold in the subsequent chapters. We also present a thorough and technical literature review of CBFs at the end of Chapter 2 hoping that the preliminaries introduced would better enable readers to appreciate the advances in this field.

- Chapter 3 focuses on the problem of enforcing multiple CBFs together and proposes a solution based on quantifying the feasible solution space of multiple CBF constraints using polytope volume. We then define a new CBF to prevent this volume from going to zero, thereby maintaining compatibility of multiple CBFs.

  – Hardik Parwana, Mitchell Black, Bardh Hoxha, Hideki Okamoto, Georgios Fainekos, Danil Prokhorov, and Dimitra Panagou (2024) (2023). Feasible Space

Monitoring for Multiple Control Barrier Functions with application to Large Scale Indoor Navigation. arXiv preprint arXiv:2312.07803 (under review)

- Next, to allow real-time adaptation of CBF based controllers, Chapter 4 introduces a new notion of Rate Tunable CBFs that allows for time-varyig class-$\mathcal{K}$ function. Two algorithms, one myopic and another based on model-predictive framework are introduced to improve performance of CBF-QP controllers while maintaining feasibility.
  - Hardik Parwana, and Dimitra Panagou. "Recursive feasibility guided optimal parameter adaptation of differential convex optimization policies for safety-critical systems." 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022.
  - Hardik Parwana, Aquib Mustafa, and Dimitra Panagou. "Trust-based rate-tunable control barrier functions for non-cooperative multi-agent systems." 2022 IEEE 61st Conference on Decision and Control (CDC). IEEE, 2022.
  - Hardik Parwana and Dimitra Panagou. "Rate-tunable control barrier functions: Methods and algorithms for online adaptation." To appear in 2025 American Control Conference (ACC). IEEE, 2025. Available on arXiv:2303.12966 (2023).
- Chapter 5 delves into uncertain systems and introduces a numerical scheme based on Unscented Transform to propagate state distributions in the presence of state-dependent uncertainties. This algorithm is called Expansion-Compression Unscented Transform (ECUT). Applications of ECUT are shown to - 1) the model-predictive tuning framework from Chapter 4 and 2) to model predictive path integral (MPPI) controller for designing control inputs for stochastic hybrid dynamical systems.
  - Hardik Parwana, Mitchell Black, Bardh Hoxha, Hideki Okamoto, Georgios Fainekos, Danil Prokhorov, and Dimitra Panagou (2024). Risk-aware MPPI for Stochastic Hybrid Systems. To appear in 2025 American Control Conference (ACC). IEEE, 2025. Available on arXiv:2411.09198.
  - Hardik Parwana, Albus Fang, and Dimitra Panagou. FORESEE: Prediction with Expansion-Compression Unscented Transform for Online Policy Optimization. arXiv preprint arXiv:2209.12644 (under review, conditionally accepted in IEEE TCST.)
- Chapter 6 introduces the problem of dropping constraints from an optimization problem. This problem is known to be NP-Hard however has not been explored in the context of dynamical systems where the optimization problem plays the role of the

controller and whose parameters change with state of the system. A heuristic is proposed to find the correlation between current infeasible optimization and past feasible optimizations to make a more informed prior for solving the NP-Hard problem.

– Hardik Parwana, Ruiyang Wang, and Dimitra Panagou. "Algorithms for Finding Compatible Constraints in Receding-Horizon Control of Dynamical Systems." 2024 American Control Conference (ACC). IEEE, 2024.

CHAPTER 2

Preliminaries

In this chapter, we present a general form of the system that is considered in the first few chapters. We also introduce our notations and provide an overview of definitions and concepts from control, probability, and optimization theory that are used or will aid understanding throughout this thesis. The later chapters however will introduce additional theories as needed. We also provide a thorough literature survey of the control barrier function (CBF) based methods at the end of this chapter. This choice was made to first provide the reader with sufficient background to understand the limitations of existing methods, and second, to allow the reader to skip directly to either of the chapters 3, 4 or 6 and still appreciate the contribution made there.

2.1 Notations

The set of real numbers is denoted as $\mathbb{R}$ which is the field this thesis will work unless stated otherwise. The set of non-negative numbers is denoted as $\mathbb{R}^+$. Let $\mathbb{R}^n$ denote n-dimensional vectors of real numbers and $\mathbb{R}^{m\times n}$ the set of matrices with m rows and n columns. In this work, a vector in $\mathbb{R}^n$ is understood to be a matrix with one column, that is, $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. M^T denotes the transpose of a matrix $M \in \mathbb{R}^{m\times n}$. I denotes the identity vector or matrix depending on the context. Let $\mathbb{Z}$ denote the set of integers.

For $x \in \mathbb{R}, y \in \mathbb{R}^n$, $|x|$ denotes the absolute value of x and $||y||$ denotes the L_2 norm of y. Unless stated otherwise, the inner product $\langle x, y\rangle$ is defined as the standard inner product induced by L_2 norm, that is, $\langle x, y\rangle = x^T y$. The character $\exists$ is shorthand for "there exists" and the character $\forall$ is shorthand for "for all" or "for any".

A set is a collection of elements. We use upper-case letters to denote sets. Given a set X, $2^{\mathcal{X}}$ denotes the power set of X, Given sets A and B, $A \subset B$ denote that A is a subset of B. We do not distinguish between $\subset$ and $\subseteq$ and $A \subset B$ allows $A = B$. Let $|A|$ denote the cardinality (number of elements) of the set A. The interior and boundary of a set $\mathcal{C}$

is denoted by $\text{Int}(\mathcal{C})$ and $\partial\mathcal{C}$. The empty set is denoted by $\emptyset$. When sets are functions of another variable, for example when the elements change with time t, we use the notation $A(t) : \mathcal{T} \to 2^X$. $A \cap B$ denotes the intersection and $A \cup B$ denotes the union of sets A, B.

Let $\lim_{x\to a} f(x)$ denote the limit of function f as its argument approaches a. The character x is used in this book mostly to refer to the system's state under consideration. The dependence on time is implicit, that is, $x \equiv x(t)$. The time derivative of x is denoted by $\dot{x}$ and is defined as $\dot{x}(t) = \lim_{\delta\to 0} \frac{x(t+\delta)-x(t)}{\delta}$. The domain D and co-domain R of a function f will be introduced as $f : D \to R$, a notation we have already used in the preceding paragraph. For a function $f(t, x)$, $\frac{\partial f}{\partial t}$ denotes the derivative (also called partial derivative) of the function w.r.t function's first argument t. Similarly $\frac{\partial f}{\partial x}$ denotes partial derivative w.r.t x. The total derivative w.r.t time t is denoted as $\frac{df(t,x)}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\dot{x}$. Finally, $\frac{\partial f}{\partial x}|_{x=y}$ is the partial (total derivative when only one argument is present) derivative evaluated at y. When the domain or co-domain is a vector space, $\frac{\partial f}{\partial x}$ is also called the gradient or Jacobian of the function f. If $f : \mathbb{R}^n \to \mathbb{R}^m$, $\frac{\partial f}{\partial x} \in \mathbb{R}^{m\times n}$.

Additional notations will be introduced in later chapters as needed.

2.2 System Description

A dynamical system is described by the following nonlinear dynamics

$$\dot{x} = F(x, u), \tag{2.1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$ and $u \in \mathcal{U} \subset \mathbb{R}^m$ represent the state and control input, and $F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is sufficiently regular so as to admit unique solutions to (2.1) for all times. In the first two chapters, we consider a special case of (2.1), the nonlinear control-affine dynamical system given by the following:

$$\dot{x} = f(x) + g(x)u, \tag{2.2}$$

where $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^m$ are locally Lipschitz continuous functions. The set $\mathcal{S}$ of allowable states is specified as an intersection of N sets $\mathcal{S}_i, i \in \{1, 2, .., N\}$, each of which is defined as the zero-superlevel set of a (sufficiently smooth) function $h_i : \mathcal{X} \to \mathbb{R}$ as:

$$\mathcal{S}_i \triangleq \{x \in \mathcal{X} : h_i(x) \geq 0\}, \tag{2.3a}$$

$$\partial\mathcal{S}_i \triangleq \{x \in \mathcal{X} : h_i(x) = 0\}, \tag{2.3b}$$

$$\text{Int}(\mathcal{S}_i) \triangleq \{x \in \mathcal{X} : h_i(x) > 0\}. \tag{2.3c}$$

In several parts of this thesis, the safe sets are also allowed to vary with time. In this case, the above definitions can be re-written to define barrier function as $h_i : \mathbb{R}^+ \times \mathcal{X} \to \mathbb{R}$ and the safe set as

$$\mathcal{S}_i(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) \geq 0\}, \tag{2.4a}$$

$$\partial\mathcal{S}_i(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) = 0\}, \tag{2.4b}$$

$$\text{Int}(\mathcal{S}_i)(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) > 0\}. \tag{2.4c}$$

2.3 Optimization

Given a cost function $f : \mathbb{R}^n \to \mathbb{R}$ and constraint function $g : \mathbb{R}^n \to \mathbb{R}^k$, an optimization problem in $u \in \mathbb{R}^m$ is given by the following

$$\min \quad f(u) \tag{2.5a}$$

$$\text{s.t.} \quad g(u) \geq 0. \tag{2.5b}$$

A trajectory optimization or model predictive control generally results in nonlinear f and g functions and hence in nonlinear optimization problems (2.5) that are notoriously difficult to solve for their globally optimal value. Several solvers exist however to find a locally optimal solution. In this work, f and g are differentiable functions and we use IPOPT [238] solver which is an interior-point method.

Some forms, especially when f and g are convex functions, are more tractable and do not possess locally optimal points thus easily giving the (unique) globally optimal solution. The following are standard forms of some convex programs frequently used in this thesis - a Linear program (LP) and a Quadratic Program (QP)

$$\text{LP} : \min \quad A^T u \tag{2.6a}$$

$$\text{s.t.} \quad C^T u \geq 0 \tag{2.6b}$$

$$\text{QP} : \min \quad u^T Q u \tag{2.7a}$$

$$\text{s.t.} \quad C^T u \geq 0, \tag{2.7b}$$

where $A, Q \succeq 0$ and $C \in \mathbb{R}^m$. Most of the implementations in this thesis use the GUROBI [102] solver to solve LPs and QPs.

Finally, we would like to introduce Lagrange multipliers for they play a brief but important

role in the last chapter of this thesis. Given a constrained optimization problem (2.5), Lagrange multipliers are used to convert it into the following min max problem subject to a simpler constraint

$$\max_{\lambda} \min_{u} \quad f(u) + \lambda^T g(u) \tag{2.8a}$$

$$\text{s.t.} \quad \lambda \geq 0. \tag{2.8b}$$

Notably, the optimal solution x^*, λ^* of (2.8) satisfies the complementary slackness condition (one of the KKT conditions of optimality)

$$\lambda^{*T} g(x^*) = 0, \tag{2.9}$$

implying that, for all $i \in \{1, .., k\}$, λ_i is zero when the constraint $g(x)_i$ is inactive and non-zero when constraint $g_i(x)$ is inactive. Therefore, the values of the Lagrange multiplier (usually returned by the solver) can be used to track the constraints have been active over time when a controller is formulated as an optimization problem whose parameters (such as matrices A, C, Q in (2.6) and (2.7)) vary with state x of the system. We exploit this insight in Chapter 6.

2.4 Barrier Functions and Set Invariance

In this section, we introduce some important definitions and results from the theory of set invariance that will aid understanding of the next section.

Definition 1. (class-$\mathcal{K}$) A continuous function $\alpha : [0, a) \to \mathbb{R}^+$ is said to belong to class-$\mathcal{K}$, denoted $\alpha \in \mathcal{K}$, if $\alpha(0) = 0$ and α is strictly increasing.

Definition 2. (class-$\mathcal{K}_\infty$) A continuous function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ is said to belong to class-$\mathcal{K}_\infty$, denoted $\alpha \in \mathcal{K}_\infty$, if $\alpha \in$ class-$\mathcal{K}$ with $a = \infty$ and $\lim_{\lambda \to \infty} \alpha(\lambda) = \infty$.

Furthermore, if the domain of the function is $[-a, a]$ ($(-\infty, \infty)$), then the function is called extended class-$\mathcal{K}$ (extended class-$\mathcal{K}_\infty$).

2.4.1 Set Invariance

Definition 3. (Distance from a set) Given a set $\mathcal{S} \subset \mathbb{R}^n$ and a point $y \in \mathbb{R}^n$, the distance of y from $\mathcal{S}$ is defined as

$$\text{dist}(y, S) = \inf_{w \in \mathcal{S}} ||y - w|| \tag{2.10}$$

Definition 4. (Tangent Cone) Given a closed set $\mathcal{S}$, the tangent cone of $\mathcal{S} \subset \mathbb{R}^n$ at $x \in \mathbb{R}^n$ is defined as follows

$$\mathcal{T}_{\mathcal{S}}(x) \triangleq \left\{ z \in \mathbb{R}^n \middle| \liminf_{\tau \to 0} \frac{\text{dist}(x + \tau z, \mathcal{S})}{\tau} = 0 \right\} \tag{2.11}$$

Below we present a simplified version of Nagumo's theorem based on [37][Theorem 4.7].

Lemma 1. (Nagumo's Theorem) Let $\dot{x} = F(x)$ and assume that for each initial condition $x(0)$ in an open set $O \subset \mathbb{R}^n$, it admits a (not necessarily unique) solution defined for all $t \geq 0$. Let $\mathcal{S} \subset O$ be a closed set. Then $\mathcal{S}$ is weakly positively invariant for the system if and only if velocity vector satisfies Nagumo's condition

$$F(x) \in \mathcal{T}_{\mathcal{S}}(x), \tag{2.12}$$

where weakly positive invariant implies that at least one of the solutions (since solutions may not be unique for $\dot{x} = F(x)$) lies inside $\mathcal{S}$. When the field $F(x)$ is such that a unique solution exists for all initial conditions $x(0) \in O$, then the unique solution is guaranteed to lie inside $\mathcal{S}$, that is, we get positive invariance instead of just weak positive invariance.

Note that in the interior of the set $\mathcal{S}$, $\mathcal{T}_{\mathcal{S}}(x) = \mathbb{R}^n$. Therefore Nagumo's condition is meaningful only on the boundary ∂S of the safe set $\mathcal{S}$.

Remark 1. In simple words, Nagumo's theorem implies that the state remains inside the safe set if and only if the velocity field ($\dot{x}$) points inside or is tangent to $\mathcal{S}$.

2.5 Control Barrier Functions

The control barrier function (CBF) is a popular method to convert state constraints to coupled state-control input constraints by exploiting the knowledge of dynamics (2.2) to control the flow of the time derivative of the state constraint. The idea is to design a control input such that the closed-loop dynamics satisfies the Nagumo's condition (2.12).

Let r_i denote the the relative degree of barrier functions h_i. When $r_i = 1$, we have the following classical CBF definition

Definition 5. (CBF) [14] The function $h_i : \mathcal{X} \to \mathbb{R}$ is a CBF for the system (2.2) on the set $\mathcal{S}_i$ defined in (2.3) if there exists an extended class-$\mathcal{K}_\infty$ function $\alpha_i : \mathbb{R} \to \mathbb{R}$ and a set $\mathcal{D}_i$ with $\mathcal{S}_i \subset D_i$ such that

$$\sup_{u \in \mathcal{U}} \left[\frac{\partial h_i}{\partial x}(f(x) + g(x)u) \right] \geq -\alpha_i(h_i(x)),\ \forall x \in \mathcal{D}_i. \tag{2.13}$$

Since α_i, being a class-$\mathcal{K}$ function, satisfies $\alpha_i(0) = 0$, at $h_i = 0$ Eq. (2.13) results in

$$\dot{h}(x, u) = \frac{\partial h_i}{\partial x}(f(x) + g(x)u) \geq 0. \tag{2.14}$$

Since (2.14) satisfied Nagumo's Theorem, the existence of a CBF h_i guarantees forward invariance of the set S_i.

Remark 2. (Candidate CBF) Note that finding a valid CBF entails simultaneously finding a function h and a class-$\mathcal{K}$ function α such that the condition (2.13) is satisfied for all $x \in \mathcal{D}$. In practice, this is not a trivial task and and often everyone uses an invalid barrier function h in their controllers to impose condition (2.13). We refer to invalid CBFs as Candidate CBFs.

Definition 5 can be generalized to barrier functions with higher relative degrees. Suppose the relative degree of the function h_i w.r.t. the control input u under the system (2.2) is $r_i \geq 2$. For r_i class-$\mathcal{K}$ functions $\alpha_i^k, k \in \{1, 2, \ldots, r_i\}$, we define the following functions:

$$\begin{aligned} &\psi_i^0(x) = h_i(x), \\ &\psi_i^k(x) = \dot{\psi}_i^{k-1}(x) + \alpha_i^k(\psi_i^{k-1}(x)),\ k \in \{1, 2, \ldots, r_i\}, \end{aligned} \tag{2.15}$$

and the set:

$$\mathscr{C}_i = \bigcap_{k=1}^{r_i} \{x \mid \psi_i^{k-1}(x) \geq 0\}. \tag{2.16}$$

Definition 6. (HOCBF) [214] The function $h_i(x) : \mathbb{R}^n \to \mathbb{R}$ is a Higher-Order CBF (HOCBF) of r_i-th order on the set $\mathscr{C}_i$ for system (2.1) if there exist r_i extended class-$\mathcal{K}$ functions $\alpha_i^k : \mathbb{R} \to \mathbb{R},\ k \in \{1, 2, .., r_i\}$, and an open set $\mathscr{D}_i \subset \mathbb{R}^+ \times \mathbb{R}^n$ with $\mathscr{C}_i \subset \mathscr{D}_i \subset \mathcal{X}$

such that

$$\sup_{u \in \mathcal{U}} \left[\dot{\psi}_i^{r_i-1}(x,u) + \alpha_i^{r_i}(\psi_i^{r_i-1}(x)) \right] \geq 0, \ \forall x \in \mathscr{D}_i. \tag{2.17}$$

In several applications in this thesis, we only consider linear class-$\mathcal{K}$ functions defined as follows

$$\alpha_i^k(h_i) = \nu_i^k h_i, \tag{2.18}$$

where $\nu_i^k \in \mathbb{R}^+$ is a constant.

Note that for $r_i = 1$, (2.17) reduces to $\dot{h}_i(x,u) \geq -\nu_i^1 h_i(x)$. And for $r_i > 1$, $\dot{\psi}_i^k$ does not depend on u for $k < r_i - 1$. If h_i is a HOCBF for the set $\mathcal{S}_i$ and there exists a Lipschitz continuous controller $u : \mathbb{R}^+ \times \mathcal{S}_i \to \mathcal{U}$ that satisfies (2.17), then the forward invariance of the set $\mathcal{S}_i$ is guaranteed under the closed-loop dynamics (2.2) [15, 134].

Remark 3. (Time-varying safe sets) Although the definitions 5 and 6 consider time-invariant safe sets, variations for time-varying safe sets have also been proposed [134, 254]. Notably, the conditions remain the same except that the derivative of CBF and HOCBF should also consider variation with time. The result can alternatively be derived using Nagumo's theorem applied to non-autonomous systems [52, Theorem 3.5.2]. The CBF condition becomes

$$\sup_{u \in \mathcal{U}} \left[\frac{\partial h_i(t,x)}{\partial t} + \frac{\partial h_i(t,x)}{\partial x}(f(x) + g(x)u) \right] \geq -\alpha_i(h_i(t,x)), \ \forall x \in \mathcal{X}, \forall t \geq 0, \tag{2.19}$$

and the HOCBF condition becomes

$$\sup_{u \in \mathcal{U}} \left[\dot{\psi}_i^{r_i-1}(t,x,u) + \alpha_i^{r_i}(\psi_i^{r_i-1}(t,x)) \right] \geq 0, \ \forall x \in \mathscr{D}_i, \forall t \geq 0. \tag{2.20}$$

Example 1. Consider a double integrator robot with state $x = [p_x, p_y, \dot{p}_x, \dot{p}_y]$ and dynamics

$$\dot{x} = \begin{bmatrix} x(3) \\ x(4) \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u, \tag{2.21}$$

where $p_x, p_y \in \mathbb{R}$ are X, Y positions of the robot and $u \in \mathbb{R}^2$ is the acceleration input to the system. Now let $h = p_x - 5$ be a 2^{nd} order barrier function and the derived barrier functions

be

$$\psi^0 = p_x - 5, \tag{2.22}$$

$$\psi^1 = \dot{\psi}^0 + \nu^1 \psi^0 = \dot{p}_x + \nu^1 (p_x - 5), \tag{2.23}$$

and the HOCBF condition is given by

$$\sup_{u \in \mathbb{R}} [\dot{\psi}^1 + \nu^2 \psi^1] = \sup_{u \in \mathbb{R}} [u_1 + \nu^1 \dot{p}_x + \nu^2 (\dot{p}_x + \nu^1 (p_x - 5))] \geq 0. \tag{2.24}$$

Upon simple glance, it can be seen that Eq. (2.24) can always be satisfied since u_1 is unbounded and, for this specific example, the coefficient of u_1 is equal to 1 (independent of state) and never vanishes. Therefore $h = p_x - 5$ is a valid HOCBF in the sense of Definition 6.

Usually, finding a valid CBF is not an easy problem. Consider that input u is bounded, $|u_1| \leq 5$. In this case, (2.24) cannot be satisfied for all safe x ($h(x) \geq 0$), and therefore $h(x)$ is not a valid HOCBF. Alternatively, consider unbounded control input but the following extended unicycle dynamics

$$\dot{p}_x = v \cos\theta, \tag{2.25}$$

$$\dot{p}_y = v \sin\theta, \tag{2.26}$$

$$\dot{\theta} = \omega, \tag{2.27}$$

$$\dot{v} = a, \tag{2.28}$$

where θ is the heading angle, v is the forward velocity, a is the forward acceleration, ω is the angular velocity, and $u = [a\ \omega]$ is the control input. The HOCBF condition then becomes

$$\sup_{u \in \mathbb{R}^2} [a \cos\theta - v \sin\theta \omega + (\nu^1 + \nu^2) v \cos\theta + \nu^1 \nu^2 (p_x - 5)] \geq 0. \tag{2.29}$$

In (2.29), coefficients of a and ω can vanish simultaneously. When they do, it may not be possible to satisfy (2.29) even if $h(x) \geq 0$ irrespective of the choice of ν^1, ν^2 (or even any nonlinear class-$\mathcal{K}$ functions).

The above example serves to illustrate that finding a valid CBF/HOCBF for dynamical systems is a non-trivial problem and requires simultaneous consideration of dynamics, control input bounds, and choice of class-$\mathcal{K}$ function.

Remark 4. Strong forward invariance using Nagumo's theorem requires the closed-loop system to admit a unique solution. Most CBF papers including seminal the work [15] (for zeroing barrier function) make the assumption that the solution is unique and then

show sufficiency of CBF condition in guaranteeing safety. It can be argued that this is a very strong assumption and infact not necessary. Some works such as [119, 254] do not require Nagumo's theorem to prove sufficiency of a CBF and CBF-like conditions in enforcing forward invaraince of safe set. They instead construct a tractable comparison system and use comparison lemma to show forward invariance. Note however that a safe system by definition lies inside a compact set (with the exception of states approaching infinity which we ignore) and therefore Nagumo's theorem is a necessary condition nonetheless and the closed-loop system must satisfy it. In this dissertation, we continue referring to Nagumo theorem since it the behavior of controllers near safe set boundary with an intuitive explanation.

Remark 5. The seminal work in [15] does not state that the class-$\mathcal{K}$ function α should also be Lipschitz continuous. However, the results it uses from [116] to prove its claims do require the class-$\mathcal{K}$ function to be Lipschtiz continuous. Some of the follow-up papers [254] explicitly state class-$\mathcal{K}$ to be Lipschtiz continuous. In this dissertation, we also assume that class-$\mathcal{K}$ functions are Lipschtiz continuous.

2.5.1 Controller Description

A standard way to design a controller satisfying CBF conditions is to impose conditions (2.17) in a Quadratic Program as follows

$$u = \arg\min_{u \in \mathcal{U}} \quad \|u - u_r(x)\|^2 \tag{2.30a}$$

$$\text{s.t.} \quad \dot{\psi}_i^{r_i-1}(x,u) \geq -\alpha_i^{r_i}(\psi_i^{r_i-1}(x)),\ i \in \{1,..,N\}, \tag{2.30b}$$

where $u_r : \mathbb{R}^n \to \mathcal{U}$ is the reference control input provided by the user. In practice, u_r is typically designed without any regard to safety constraints to fulfill stability objectives like reaching a target location. While several works also impose relaxed Lyapunov equations (relaxed with slack variable to give higher priority to safety constraints), we assume all such relaxed constraints are part of nominal controller u_r.

Controller (2.30) reduces to the following popular form when all relative degrees are 1 and linear class-$\mathcal{K}$ functions are used

$$u = \arg\min_{u \in \mathcal{U}} \quad \|u - u_r(x)\|^2 \tag{2.31a}$$

$$\text{s.t.} \quad \dot{h}(x,u) \geq -\nu_i^1 h(x),\ i \in \{1,..,N\} \tag{2.31b}$$

2.6 Probability Distributions

For $y \in \mathbb{R}^n$, $y \sim D$ denotes that the random variable y follows the distribution D. The distribution may be multivariate, that is, $y \in \mathbb{R}^n$. In this book, the distribution may also change with state x of the system and will be used with appropriate arguments as needed, for example, $y \sim D(x)$. The probability of an event $A \subset \Omega$ where Ω, where Ω is the sample space, is denoted by $P(A)$. We only use continuous random variables in this work. The cumulative distribution function (CDF) is defined $F_X(x) := P\big(X \leq x\big)$ and the probability distribution function (PDF) is defined as $f_X(x) := \dfrac{\mathrm{d}}{\mathrm{d}x} F_X(x)$.

The n-th moment of a real-valued continuous random variable with PDF $f(x)$ about a value c is the integral

$$\mu_n = \int_{-\infty}^{\infty} (x - c)^n f(x) dx. \tag{2.32}$$

In this work, we use the standard values of c, $c = 0$ for $n = 1$, and $c = \mu_1$ for $c > 1$. The first moment $n = 1$ is referred to as the mean or expectation of X. We use the notation $\mathbb{E}[X]$ and $\mu(X)$ to refer to expectation as both are commonly used. The variance, the second moment, is denoted as $\Sigma[X]$, skewness, the third moment, is denoted by $S(X)$, and kurtosis, the fourth moment, is denoted by $K(X)$. Based on (2.32), the aforementioned moments are thus defined as follows

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) dx, \tag{2.33}$$

$$\Sigma[X] = \int_{-\infty}^{\infty} (x - \mathbb{E}[X])(x - \mathbb{E}[X])^T f(x) dx, \tag{2.34}$$

$$S(X)_{ijk} = \mathbb{E}[(x - \mathbb{E}[X])_i (x - \mathbb{E}[X])_j (x - \mathbb{E}[X])_k], \quad i, j, k \in \{1, .., n\}, \tag{2.35}$$

$$K(X)_{ijkl} = \mathbb{E}[(x - \mathbb{E}[X])_i (x - \mathbb{E}[X])_j (x - \mathbb{E}[X])_k (x - \mathbb{E}[X])_l] \quad i, j, k, l \in \{1, .., n\}, \tag{2.36}$$

where note that $\mathbb{E}[X]$ is $n \times 1$ vector, Covariance is a $n \times n$ symmetric matrix (second-order tensor), S is a 3-dimensional matrix (3^{rd} order tensor) whose (t, j, k) element is denoted as S_{ijk}, and kurtosis is a 4-dimensional matrix (4^{th} order tensor) whose i, j, k, l element is denoted as K_{ijkl}. Covariance is a measure of the spread of the random variable but it cannot encode asymmetry. Skewness on the other hand is a measure of asymmetry. For a Gaussian random variable X, skewness is zero.

In Chapter 5, we work on the premise that retaining information about a distribution is equivalent to retaining information about its moments. Although this thesis does not require the reader to understand the origin of this premise, we mention the two contributing results

here for the sake of completeness.

The following theorem states that a probability distribution is completely determined by its moments under the specified conditions

Theorem 1. (Hamburger moment problem) [79] Let μ be a measure on $\mathbb{R}$ such that all the moments

$$m_n = \int_{-\infty}^{\infty} x^n d\mu(x) \tag{2.37}$$

are finite. If $\sum_{n=1}^{\infty} m_{2n}^{-\frac{1}{2n}} = +\infty$ then the moment problem for (m_n) is determinate, that is, μ is the only measure on $\mathbb{R}$ with (m_n) as its sequence of moments.

Another result that contributed to our choice of moments is the following Taylor series expansion of the function of random variable [161] [82]. Given a random variable X with mean μ and a infinitely differentiable function f, we have

$$f(X) = f(\mu) + \sum_{i=1}^{n} \frac{\partial f(X)}{\partial X_i}(X_i - \mu_i) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f(X)}{\partial X_i \partial X_j}(X_i - \mu_i)(X_i - \mu_i) + \cdots . \tag{2.38}$$

We can now take moments from both sides to find expressions of moments of $f(X)$ in terms of moments of X and derivatives of f. Readers are referred to [161] [82] for more details.

2.7 Literature Review of CBFs

As mentioned in Chapter 1, we take an unconventional approach and include a thorough survey of existing CBF methods in this last section. The preliminaries introduced in this chapter would aid in a better understanding of the limitations of the existing approaches and the contributions made herewith in the next two chapters.

Despite significant extensions of the CBF concept to address various challenges (e.g., modeling/parameter uncertainty, input constraints) over the last few years, we highlight three unsolved issues.

2.7.0.1 Finding valid CBFs

Given a single constraint function, finding a suitable (valid) CBF reduces in effect to finding both a barrier function and a class-$\mathcal{K}$ function so that the CBF condition can be realized for given dynamics and control input bounds. This task is non-trivial and several approaches, mostly offline, have been proposed recently [72, 109]. When the safe set is represented as an

intersection of the sets defined by several constraints, for example, when there are multiple unsafe regions such as obstacles, the objective becomes finding a controller that simultaneously ensures the invariance of the intersection of the sets. Existing CBF implementations either combine all barriers into a single barrier function [96, 210, 169] resulting in a single CBF derivative condition, or consider each barrier separately [229, 256, 241] thereby imposing multiple CBF derivative conditions simultaneously. Either case only adds to the complexity of finding valid CBFs.

2.7.0.2 Feasibility and optimality

CBF controllers are in principle obtained as the result of a Quadratic Program (QP) searching for the point-wise minimum-norm control input, and therefore are myopic, i.e., only point-wise optimal. This can lead to sub-optimal performance in the long term. In other words, the point-wise solutions can be inferior in terms of performance compared to trajectories obtained by the exact solution of the Infinite or Finite Horizon Optimal Control [69, 175] problem. Furthermore, imposing multiple CBFs and control input bounds in the optimization problem can lead to infeasibility of the solution (controller) at the current or at some future time. We call this problem "lack of persistent feasibility" of the low-level CBF controller. Note that a good choice of CBF parameters may lead to persistent feasibility, whereas a poor choice may not; however, finding a good choice of parameters is not a trivial problem.

2.7.0.3 High relative degree

The notion of CBFs as first introduced in [15] applies to constraints of relative degree $r = 1$ with respect to the system dynamics, i.e., to constraints whose first-order derivative is a function of the control input. The concept is extended to constraint functions of higher relative degree $r > 1$, i.e., to the case where the control input appears in the r-th -order derivative of the constraint function. [159] proposed the class of Exponential CBFs when the class-$\mathcal{K}$ functions used in CBF derivative condition are linear in their argument. They employ input-output linearization and design controllers via pole placement so that the barrier state is stabilized to zero. [253] generalizes Exponential CBFs to generic nonlinear class-$\mathcal{K}$ functions in the form of Higher-Order CBFs (HOCBF). [219] proposes an alternate Lyapunov backstepping-based design for higher-order systems that always find a first-order barrier function.

2.7.1 Current approaches and limitations

2.7.1.1 Finding valid CBFs

In recognition of the importance of finding a valid control barrier function, several works have also attempted to learn a barrier function, or to find the domain over which a given barrier function satisfies the CBF condition. [10] does offline analysis to find the domain over which CBF condition can be satisfied in the presence of input bounds. [191, 137] perform offline analysis to find a barrier function that satisfies the CBF derivative condition. A more thorough survey on learning-based methods can be found in [73].

When multiple constraints are present, [210, 169] combine barrier functions into a single barrier constraint through smoothed minimum (or maximum) operator. [34] proposes a novel weighted-summation technique, where the weights are tuned for performance and to avoid singular states where the control input has no contribution to barrier function derivatives (usually the latter is only assumed, but not guaranteed, to not happen in the literature). The authors in [96] perform non-smooth analysis to construct a barrier function with the non-smooth minimum (or maximum) operation. The aforementioned approaches thus either involve smoothing approximations or require intricate non-smooth analysis. When multiple CBF derivative conditions are considered simultaneously instead as in [229, 256, 241], [215] proposes an offline sampling-and-grid refinement method to search the domain of state space over which multiple control barrier functions are compatible. Multiple CBF constraints also arise naturally in multi-agent scenarios. [135, 136] encode Signal Temporal Logic (STL) tasks and safety specifications as CBF constraints. [39, 231] propose CBFs for safe swarm behavior in the presence of uncooperative agents when their identities are known in advance. These uncooperative agents serve as an excellent example of uncontrollable systems, where different CBF class-$\mathcal{K}$ functions are suited to controllable and uncontrollable dynamics. The safe sets are also time-varying for the controllable dynamics, which most of the aforementioned papers have not considered in their theoretical analysis. [231] designs a robust CBF controller for systems with adversarial agents without detecting malicious agents. [157] detects malicious agents by checking for collisions in the predicted future wherein the ego and target agents, forgoing their other objectives, try to only maintain and violate respectively, the CBF derivative constraint. Both the above-mentioned approaches are conservative due to the consideration of worst-case bounds.

2.7.1.2 Feasibility

All of the aforementioned works use a constant class-$\mathcal{K}$ function over the whole domain, and its parameters are often pre-tuned manually before deploying for an actual application.

However, since it is not possible to test each and every scenario in the offline phase, there is a dire need for online adaptation, especially to maintain the feasibility of the controller in the presence of multiple constraints and input bounds. Several works relax the CBF constraint in the QP to ensure that the QP controller has a solution. [266] proposes an optimal-decay CBF-QP that multiplies the class-$\mathcal{K}$ function by a scalar variable that is optimized for (along with the control input) in the QP. [241], [91] introduced similar relaxation strategies for CBF constraints with the subtle difference being that instead of linear multiplicative factor to generic class-$\mathcal{K}$ function, they only consider linear class-$\mathcal{K}$ functions and treat the class-$\mathcal{K}$ function parameter as an optimization variable. [199] considers control non-affine dynamics and in light of the resulting loss of linearity of the CBF constraint, chooses to impose it through a penalty term in the cost function, thereby converting it into an unconstrained optimization problem.

The aforementioned adaptive approaches can be thought of as adapting *on a need basis*; i.e., they relax the CBF constraint whenever the original class-$\mathcal{K}$ function leads to infeasibility. However, persistent feasibility is not guaranteed for multiple CBF constraints in the presence of input bounds. Since they also solve for the control input point-wise (via a QP), they are still sub-optimal in the long term. Among the non-myopic approaches, [256] multiplies a scalar penalty to the class-$\mathcal{K}$ functions, and creates a dataset of persistently feasible and eventually infeasible parameters offline by randomly sampling parameters and simulating the system forward. Then they fit a differentiable classifier that is used to impose the resulting feasibility margin as another HOBCF constraint in the system. [142] performs offline learning to learn the best-performing parameters of the class-$\mathcal{K}$ function as a function of the state, although without consideration of feasibility.

2.7.1.3 Optimality

In an effort to produce optimal trajectories over a time horizon with the hitherto mentioned approaches, predictive frameworks similar to Model Predictive Control have inspired several developments. [252], [267] impose control barrier function constraints for every state and input pair in an MPC formulation. [265] employs CBFs with penalty class-$\mathcal{K}$ functions from [266] to specify safety constraints at each time step of the MPC formulation to resolve the trade-off between feasibility and safety, and enhance them both at the same time. [98] shows that, for legged robots, using CBF constraints in both the MPC-based high-level planner and the low-level controller leads to the best results in terms of number of missteps. While the aforementioned approaches report improvement in performance, incorporating CBF constraints in MPC usually results in nonlinear optimization that typically cannot be solved fast enough to replace low-level controllers running at, for example, 40Hz or more.

The low-level CBF controllers can also be made less myopic by taking incorporating future predictions in their design. [32] propose a future-focused CBF, wherein they construct a CBF based on the minimum distance to the boundary of unsafe set in the future predicted based on zero acceleration policy. [43] also proposes a predictive control barrier function (PCBF) that, given a nominal trajectory in terms of sequence of control inputs, predicts the future and upon encountering a safety violation, uses the sensitivity of state at the future instant of safety violation w.r.t the first input in the sequence to reduce the violation.

All of the aforementioned works except for [32] and [43] perform offline analysis (training or verification) to find a control barrier function. Since it is computationally infeasible to simulate all possible scenarios offline that the robotic agent may be subjected while operating in real-world, there is a need for online learning and adaptation in face of unforeseen environments. This need can result from, e.g., a varying number of agents in a multi-agent scenario, or time-varying safe sets that the robot is subjected to based on the user's whim, such as changing collision radius. Most of the learning and verification-based frameworks [72, 109] also focus on learning a barrier function that can satisfy the CBF derivative condition for a fixed class-$\mathcal{K}$ function. However, class-$\mathcal{K}$ can not only drastically change the realizable safe set, but also affects the performance in the long term. The methods that do adapt class-$\mathcal{K}$ function online, do so through a relaxation that does not help improve the optimality of the resulting trajectory or guarantee persistent feasibility in presence of time-varying multiple barrier constraints and input bounds. Therefore, a generic framework to handle multiple CBF constraints and ensure feasibility in an online phase does not exist.

CHAPTER 3

Feasible Space Perspective on CBFs

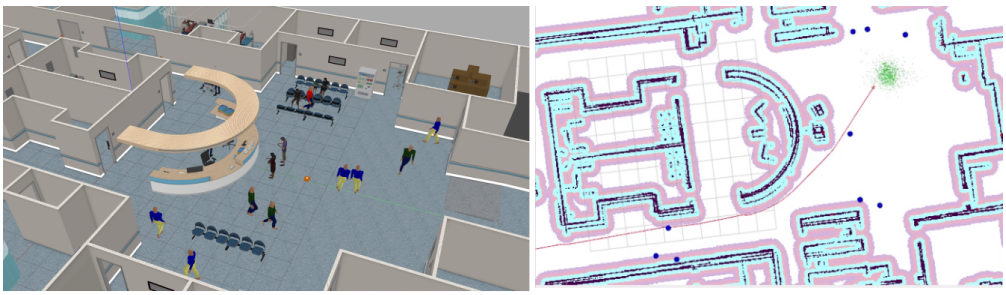
In Chapter 2, we formulated the QP controller (2.30) to enforce multiple CBF (or HOCBF). While each $h_i, i \in \{1, .., N\}$ may individually be a HOCBF, they need not admit a solution simultaneously. Consequently, a solution to (2.30) may not exist even if each h_i is HOCBF.

We call the property of having a common solution to all HOCBF conditions as ***compatibility***. In this chapter, we construct a new CBF to keep the user-provided N CBFs compatible. That is, it prevents the state from entering parts of state space where the N CBFs do not accept a common solution.

Besides having a theoretical contribution of introducing a *measure of compatibility* in Section 3.3, our method also has a practical appeal, arguably even more so than the theoretical contribution itself. Our proposed method relies on the use of an existing method of constructing valid CBFs, as will be seen later. Therefore, the use of our newly introduced notion of measure of compatibility is limited by the capabilities of existing methods. In practice, however, we use our measure of compatibility as a candidate CBF and show that it helps improve feasibility over the baseline HOCBF-QP controller given by (2.30).

3.1 Literature Review and Statement of Contributions

Designing a safety-critical robot motion stack for ensuring the satisfaction of multiple, possibly time-dependent, state and input constraints is an active area of research. This work deals with the formulation of low-level controllers for control-affine dynamical systems based on control barrier functions (CBFs). Given a constraint set, a CBF for given robot dynamics can be designed such that imposing a restriction on its rate-of-change in the control design guarantees constraint satisfaction for all times. This condition is typically imposed on the control input in an optimization problem such as a quadratic program (QP). In the presence of multiple constraint sets, it is common to simultaneously impose CBFs designed independently for each constraint in the controller. However, in such implementations, no



(a) AWS Hospital Gazebo Environment (b) Rviz visualization of map and planned path

Figure 3.1: Indoor navigation scenario (a) Robot (green circle) navigating in AWS Hospital gazebo environment using proposed CBF controller. (b) Rviz visualization of the global map, global planner's path (green), humans (blue) and the nearest static obstacles (red, found in the way described in Section 3.5.3) used by the robot for collision avoidance.

guarantees for existence of a control satisfying all the CBF constraints exist, and violation of safety may result.

In this work, we introduce a CBF that enforces the existence of a common solution to multiple CBF constraints at all times, a property we call persistent compatibility. Towards this, we introduce the following new metric. At a given state and time, we quantify the feasible control input space defined by multiple CBF constraints and input bounds in terms of its volume. For example, for control affine dynamics and polytopic input constraints, the feasible control space is also given by a convex polytope. Next, we introduce a novel feasible-space control barrier function (FS-CBF) that prevents this volume from going to zero. Our controller is thus a QP that augments the user-provided CBF constraints with an additional FS-CBF constraint. The existence of FS-CBF is shown to be a sufficient condition for enforcing the persistent compatibility of multiple CBFs.

Two commonly used approaches employed in the literature for handling multiple CBF constraints are: combining all barrier functions into a single one [169, 96, 210] or imposing the barriers simultaneously [229, 6, 241]. In [169, 210], barrier functions are combined into a single barrier constraint through a smoothed minimum (or maximum) operator. Employing non-smooth analysis, [96] constructs a barrier function with the non-smooth minimum (or maximum) operation. All of the aforementioned approaches assume compatibility of CBFs when imposed together [229, 6] or assume that the newly constructed CBF is a valid CBF for dynamics of the system and input bounds.

A few studies explicitly try to circumvent the persistent compatibility issue. A sampling-and-grid refinement method is proposed in [215] to search the domain of state space over which multiple control barrier functions are compatible. However, it does not propose a controller to confine the robot to this domain. A sum-of-squares (SoS)-based CBF synthesis

method when constraint sets are described by polynomial functions is given in [68]. In [44], sufficient conditions are provided for designing multiple CBFs concurrently so that a QP-based controller using all the CBFs will always be feasible. The work by [35] consolidates multiple CBFs into one CBF and proposes a predictor-corrector adaptation law for weights of constituent constraint functions. Additionally, a CBF learning framework is proposed in [184] for decentralized but cooperative multi-agent systems. Some works allow class-$\mathcal{K}$ functions to adapt by having time-varying parameters. The work by [256] creates a dataset of persistently compatible and eventually incompatible parameters by randomly sampling parameters and simulating the system forward. Adaptation of class-$\mathcal{K}$ parameters is also performed by posing CBF-QP as a differentiable layer in a predictor-corrector framework [175] and in learning paradigm [257] for learning of safe policies.

We introduce a procedure that exploits existing approaches such as reachability analysis [226] to synthesize a valid FS-CBF. Owing to the limitations of these approaches though, it might not always be computationally feasible to find FS-CBF. In such cases, we show empirically that that imposing feasible space volume as a candidate FS-CBF enhances the persistent compatibility of standard CBF formulations. It is also observed to reduce the sensitivity of CBF-QP controllers to user-chosen parameters such as gains of the nominal controller. Furthermore, finding the volume of feasible space and its gradient with respect to constituent hyperplanes are non-trivial problems as they do not accept analytical expressions in general. As a contribution, we also introduce some of the existing approximations suitable for control-oriented applications. To the best of our knowledge, we are the first to present this paradigm of quantifying the feasible control input space of CBFs as well as consider an optimization problem's feasible solution space volume and its gradient in control design. Previous approaches simply check if the CBF-QP is feasible, which amounts to checking whether the volume is non-zero. Note that instead of synthesizing FS-CBF, it may be possible to consolidate multiple CBFs into a single valid CBF offline. However, such an ideal scenario is difficult, if not impossible, to achieve due to prohibitive computational requirements of existing approaches. It may also be difficult to account for all possible possibilities in the offline phase. For example, such scenarios may have varying number of humans and obstacles. Our motivation lies in ensuring safety in highly constrained indoor environments, such as in Fig. 3.1 with static as well as dynamically moving agents like humans. Our idea to monitor feasible space volume thus has a high practical appeal for robotic applications.

Finally, paired with ROS2 navigation stack planners, we also provide the first application of CBF-QP controller on a large scale indoor navigation scenario by implementing our controller on AWS Hospital environment in Gazebo [2]. Several existing approaches such

as [200, 177, 48] for indoor navigation with humans rely only on fast replanning through a low-level planner based on simplified dynamics for collision avoidance, and do not employ a safety-critical controller. Our controller may thus complement existing planning frameworks. Among CBF-based approaches, [237] uses CBF-MPC, an MPC controller with constraints defined by CBF derivative conditions, for navigation in crowd, and [223] designs a novel CBF for polytopic obstacles and uses it in the CBF-MPC framework for navigation in corridor environments. From an application perspective, [237] does not consider indoor environments and [223] does not consider dynamic moving agents. While MPC does help with maintaining feasibility of CBF constraints, it is unlike our method of monitoring the volume of feasible space.

Summary of Contributions

- Introduced the notion of persistent compatibility and formulated Feasible-Space Control Barrier Function (FS-CBF) to quantify and restrict the rate-of-change of the volume of the feasible control input space.
- Demonstrated through simulations that FS-CBF enhances persistent feasibility and reduces hyperparameter sensitivity in CBF-QP controllers. We also demonstrate our controller in a large-scale indoor navigation scenario, integrated with ROS2 navigation stack planners.

3.2 Problem Formulation

We use the notation introduced in Chapter 2. Given a barrier function $h_i, i \in \{1, .., N\}$, the set of control inputs that satisfy the HOCBF derivative constraint (2.30b) for a given state x at time t are given by

$$\mathcal{U}_i(t,x) = \{u \in \mathbb{R}^m \mid \dot{\psi}_i^{r_i-1}(t,x,u) + \alpha_i^{r_i}(\psi_i^{r_i-1}(t,x)) \geq 0\}. \tag{3.1}$$

Definition 7. Feasible CBF Space: At a given time t and state $x \in \mathcal{X}$, the feasible CBF space $\mathcal{U}_c \subset \mathbb{R}^m$ is the intersection of sets $\mathcal{U}_i, i \in \{1, .., N\}$ with control input domain

$$\mathcal{U}_c(t,x) = \bigcap_{i=1}^{N} \mathcal{U}_i(t,x) \bigcap \mathcal{U}. \tag{3.2}$$

For control-affine dynamics in (2.2), the constraints (3.1) are affine in u and represent halfspaces. If the input domain is also polytopic, that is, $\mathcal{U} = \{u \in \mathbb{R}^m \mid A_u u \leq b_u\}, A_u \in$

$\mathbb{R}^{2m\times m}, b \in \mathbb{R}^{2m}$, then the feasible space $\mathcal{U}_c$ is a convex polytope. Figure 3.2 visualizes the polytope for a single integrator-modeled robot as it navigates around obstacles. For brevity, we will be denoting the feasible space in following form:

$$\mathcal{U}_c(t,x) = \{u \in \mathbb{R}^m | A(t,x)u \leq b(t,x)\}, \tag{3.3}$$

where $A : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^{(N+2m)\times m}, b : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^{(N+2m)\times 1}$. We also denote each row of A by a_i and each element of b by $b_i, i \in \{1, 2, .., N\}$. Next, we formally introduce the notion of compatibility and state the objective of this chapter.

Definition 8. (Compatible CBFs) At time t and state x, the barriers $h_i, i \in \{1, .., N\}$ are called compatible if $\mathcal{U}_c(t,x) \neq \emptyset$. Furthermore, given a Lipschitz continuous controller $\pi : \mathcal{X} \to \mathcal{U}$, the barriers h_i are called persistently compatible for dynamics (2.1) with $u = \pi(x)$ in (2.1) if $\exists E \subset \mathcal{X}$ such that $x_0 \in E \implies \mathcal{U}_c(t,x) \neq \emptyset$ for all $t \geq 0$.

Problem 1. Consider the dynamical system (2.2) subject to N barrier function constraints $h_i(t,x) \geq 0$ where each h_i, defined in (2.3), is a HOCBF for the set $\mathcal{S}_i$ respectively. If for the initial state $x_0 \in \mathbb{R}^n$, $h_i(t,x_0) > 0, i \in \{1, .., N\}$, $\mathcal{U}_c(0,x_0) \neq \emptyset$, design a controller such that $\forall t > 0$, $h_i(t,x(t)) > 0$ and $\mathcal{U}_c(t,x) > 0$.

We propose a solution to Problem 1 in Section 3.3.1. However, owing to limitations of related literature, our solution may be difficult to realize in practice. We therefore propose several approximations in Sections 3.4 and 3.4.1 and provide empirical validation of their efficacy in Sections 3.5.2 and 3.5.3.

3.3 Feasible Volume Control Barrier Function

The infeasibility of CBF-QP results when $\mathcal{U}(t,x)$ becomes empty. In this section, we propose a novel control barrier function built upon the feasible space as a means to solving Problem 1.

3.3.1 Polytope Volume Barrier Function

A measure of the feasible space is its volume. Let $\mathcal{V}(t,x) = \text{vol}(\mathcal{U}_c)(t,x)$ where vol is the volume function that, for a polytope $P = \{u | Au \leq b\}$, is defined as follows

$$\text{vol}(P) = \int_{u\in P} d\mathcal{V}, \quad d\mathcal{V} = \prod_{j=1}^{m} du_j, \tag{3.4}$$

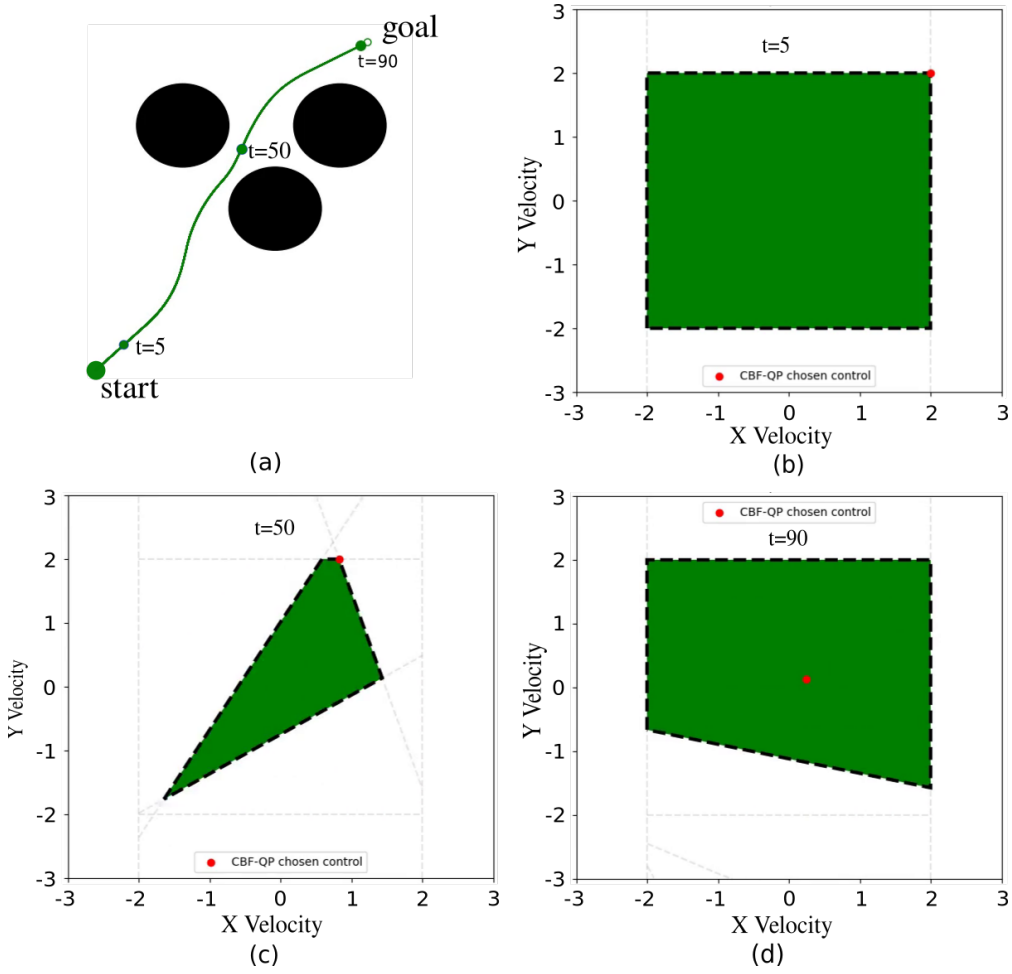


Figure 3.2: Feasible space (green) of a single integrator agent with X, Y velocity inputs as it navigates to its goal location with N=3 in (2.30) while avoiding obstacles (black). (a) The path in configuration space, (b), (c), (d) feasible control space visualization at different times. The dotted lines represent hyperplanes that constrain the control input. The 2×2 square is formed by the imposed control input bound. Other hyperplanes are formed as a result of CBF equations in (3.3). A video of animation can be found on thesis website: https://github.com/hardikparwana/PhD-Thesis

where du_j is an infinitesimal change in u_j. Note that $\mathcal{V}(t,x) = 0$ when $\mathcal{U}_c(t,x) = \emptyset$. The compatible state space is defined as the set of states $D(t) = \{x \in \mathcal{X} \mid \mathcal{V}(t,x) \geq 0\}$ and is visualized in Fig. 3.3. We then aim to find a CBF $\mathcal{V}_c : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$, with class-$\mathcal{K}$ function $\alpha_{\mathcal{V}_c}$, that renders $D(t)$ forward invariant by imposing the following condition

$$\dot{\mathcal{V}}_c(t,x,u) \geq -\alpha_{\mathcal{V}_c}(\mathcal{V}_c(t,x)). \tag{3.5}$$

We now present the following theorem that states the sufficiency of FS-CBF condition (3.5) for compatibility of h_i.

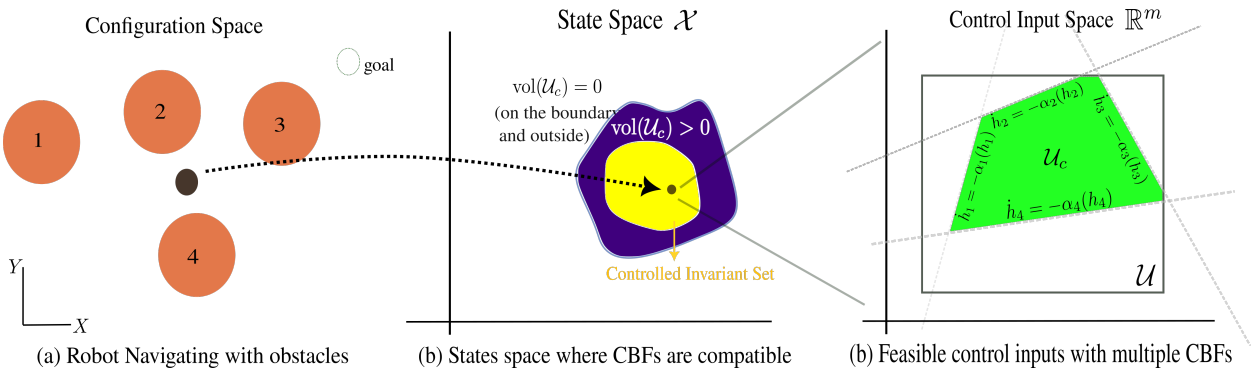
Theorem 2. Let $h_i, i \in \{1,..,N\}$ be N barrier functions corresponding to N safe sets defined as in (2.3). Let $\mathcal{V}(t,x)$ be the volume of the feasible CBF space $\mathcal{U}_c(t,x)$ in the control domain defined in (3.2). For $0 < \epsilon << 1$, let $D(t) = \{x \in \mathcal{X} \mid \mathcal{V}(t,x) \geq \epsilon\}$ be the compatible state space of $h_i, i \in \{1,..,N\}$. Then h_i are persistently compatible if there exists $\mathcal{C} \subset \mathcal{D}$ and a function $\mathcal{V}_c : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$ such that $\mathcal{V}_c$ is a control barrier function on $\mathbb{R}^+ \times \mathcal{C}$ and $x(0) \in \mathcal{C}$.

Proof. Suppose that $\mathcal{V}_c$ is a CBF with class-$\mathcal{K}$ function $\alpha_{\mathcal{V}}$ on $\mathbb{R}^+ \times \mathcal{C}$, with $\mathcal{C} \subset \mathcal{D}$. If $x(0) \in \mathcal{C}$, then $x(t) \in \mathcal{C}, \forall t > 0$ under the action of the controller that satisfies (3.5). Therefore, $\mathcal{V}_c(t,x(t)) \geq 0, \forall t > 0$ and hence $\mathcal{V}(t,x(t)) \geq \epsilon$ for all $t > 0$, implying persistent compatibility. □

Note that sets of measure 0 have zero volume. Therefore, in Theorem 3.3 we enforce $\mathcal{V} \geq \epsilon$ to distinguish between cases $\mathcal{U}_c = \emptyset$ from the case when $\mathcal{U}_c$ is a set of measure 0.

Computing $\mathcal{V}(t,x)$ requires computing the volume of the feasible space $\mathcal{U}_c$. This is not a trivial problem and even when feasible space is shaped as a convex polytope as in (3.3), it does not admit analytical expressions for $\mathcal{V}$. In the next section, we introduce some existing approximate approaches that are suitable for our application.

Contrary to intuition, only imposing (3.5) in a controller does not ensure that the chosen control input lies inside the feasible space. Since instantaneously violating a CBF constraint does not lead to an immediate violation of safety, choosing a control input outside of $\mathcal{U}_c(t,x)$ (thereby violating at least one out of N CBF constraints) may help in increasing $\mathcal{V}_c$ and therefore satisfies the condition (3.5). $\mathcal{V}_c$ is strictly used to enforce compatibility of h_i. To ensure that chosen control input maintains compatibility of h_i as well as lies inside the



(a) Robot Navigating with obstacles (b) States space where CBFs are compatible (b) Feasible control inputs with multiple CBFs

Figure 3.3: (a) Workspace of black robot with illustrative orange obstacles, (b) The state space. In the violet region, CBFs are compatible, i.e., the volume of the polytope $\mathcal{U}_c$ is > 0. The yellow region is the largest control invariant set contained in the violet region. (c) The feasible control space $\mathcal{U}_c$ defined by multiple CBFs at the current state and time. It's volume is given by the area of the green region in this 2D example.

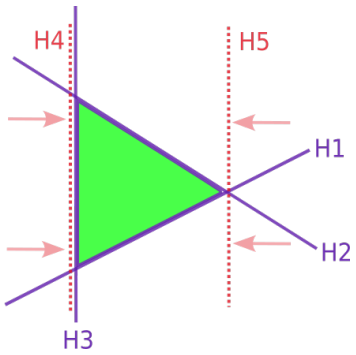


Figure 3.4: Feasible space (green) defined by hyperplanes H1-H5. H3, H4 are overlapping. Suppose H4, H5 are free to move horizontally. It can be shown that the gradient of the area of the green region exists w.r.t position of H5 but not w.r.t H4. This is because if H4 is moved to the right, it decreases the green area substantially (non-zero right side gradient) whereas if moved to the left, the green area is unaffected. H5 on the other hand, although causes the volume to decrease if moved to the left, due to its intersection at a single point, has a left-side gradient zero.

feasible space $\mathcal{U}_c$, we design the following controller

$$\pi(t,x) = \arg\min_{u\in\mathcal{U}} \quad ||u - u_{ref}(t,x)||^2 + M\delta^2 \tag{3.6a}$$

$$\text{s.t.} \quad \dot{\mathcal{V}}_c(t,x,u) \geq -\alpha_{\mathcal{V}_c}(\mathcal{V}_c(t,x)) + \delta \tag{3.6b}$$

$$\psi_i^{r_i-1}(t,x,u) \geq \alpha_i^{r_i}(\psi_i^{r_i-1}(t,x))$$

$$i \in \{1,..,N\}, \tag{3.6c}$$

where δ is a slack variable used to relax (3.6b) and $M >> 1$. The relaxation is only necessary when, as discussed in next section, an approximation of FS-CBF is used in practice.

Remark 6. Note that by nature of optimization in (2.30), if u_{ref} lies outside $\mathcal{U}_c$, then the QP solution $\pi(t,x)$ must lie on boundary of $\mathcal{U}_c$. We will demonstrate later in Section 3.5.2 and Fig. 3.6 that imposing constraint (3.6) results in a control being chosen in the interior of $\mathcal{U}_c$, even when the nominal control input is outside the feasible space. This is done to prevent the volume from shrinking too fast. The controller (2.30) on the hand always chooses a point on the boundary of feasible space whenever u_r is outside $\mathcal{U}_c(t,x)$. A mapping from u_{ref} to the interior of feasible space was also designed in [260] through a gauge map. The gauge map was designed in a different way than ours and focused on obtaining a differentiable control design with a single barrier constraint only.

3.4 An online method to monitor feasibility - Polytope Volume as Candidate CBF

In Section 3.5.1, we show an application of an existing CBF synthesis method to construct $\mathcal{V}_c$ for a simple system. For highly dynamic and constrained environments though, as previously mentioned, it is usually computationally prohibitive to consider all possible scenarios in the design of CBFs. Therefore, as is typically done in practice, we assume the constraint function is the CBF, that is, $\mathcal{V}_c = \mathcal{V}$ and use the condition (3.5) to promote persistent compatibility. When imposing $\mathcal{V}_c = \mathcal{V}$, we impose linear class-$\mathcal{K}$ functions in our simulation and experiment results. We also relax the constraint (3.6b) with a slack variable

3.4.1 Polytope Volume Estimation

In this section, we describe several ways to approximately compute the volume of feasible space.

3.4.1.1 Monte Carlo

A common approach to estimating polytope volume is to evaluate the integral with Monte Carlo (MC) sampling [129, 83]. We randomly sample K particles $p_i \in \mathcal{U}$ and evaluate the volume as follows

$$\mathcal{V}(t,x) = \frac{\sum_{j=1}^{K} \mathbb{1}(p_i \in \mathcal{U}_c)}{\text{vol}(\mathcal{U})}, \tag{3.7}$$

where $\text{vol}(\mathcal{U})$ is assumed to be given and easy to compute practically as it is often defined as a symmetric polygon.

3.4.1.2 Chebyshev Ball of a Polytope

The center c and radius r of the largest sphere contained inside a polytope, called Chebyshev ball, can be computed with the following Linear Program (LP) [40, Chapter 8]

$$\min_{c,r} \quad -r \tag{3.8a}$$

$$\text{s.t.} \quad a_i c + ||a_i|| r \leq b_i, \ \forall i \in \{1, 2, .., N\}, \tag{3.8b}$$

where a_i and b_i are the i^{th} rows of polytope matrices A, b.

3.4.1.3 Inscribing Ellipsoid

An ellipoid E is an image of the unit ball under affine transformation (B, d), $E = \{Bu + d \mid ||u|| \leq 1\}$ where $B \in S^m_{++}$ and $d \in \mathbb{R}^{m \times 1}$. Using the fact that the ellipsoid volume is proportional to $\det B$, the maximum volume ellisoid can be obtained by solving the following convex optimization problem [40, Chapter 8]

$$\min_{B,d} \quad \log\det B^{-1} \tag{3.9a}$$

$$\text{s.t.} \quad ||Ba_i|| + a_i d \leq b_i, \ \forall i \in \{1, 2, .., N\}, \tag{3.9b}$$

In practice, Monte Carlo provides most accurate estimates of polytope volume in our simulation studies. The simple MC method in (3.7) however is known to perform poorly in high-dimensional spaces. Since Monte Carlo estimates are prone to be noisy, this method is also not very suitable for evaluating gradients, which are needed to enforce CBF conditions. While more accurate but relatively expensive MC variations for volume computations [5] are known to exist, we leave the investigation of their suitability for control applications to

future work. The implementation with circle approximation is fastest since it only requires solving an LP however it suffers from non-uniqueness of solutions and therefore the LP solver may converge to different solutions for the same state configuration (We comment more on the practical implications of this in Section 3.5). Moreover, the circle may severely underapproximate the polytope volume. The ellipsoid approximates the volume much more accurately than the circle but is slower to implement. In Section 3.5, we observe that even with 10 constraints, the SDP can be solved fast enough for real-time implementations.

3.4.2 Smooth Approximation of Polytope Volume

The polytope volume $\mathcal{V}(t,x)$ is not a differentiable function in general. An example configuration that is non-differentiable is shown in Fig. 3.4. Therefore, $\mathcal{V}(t,x)$ should ideally be formulated as a non-smooth CBF [96] and generalized gradients of the exact volume given by (3.4) as well as of the ellipse and circle approximations should be used. We defer such an extensive non-smooth analysis to future work. In this work, we utilize gradients of circle and ellipse approximations that are computed using existing tools, details of which are mentioned in Section 3.5. In the remainder of this section, we propose a smooth underapproximation of (3.4).

Theorem 3. Consider the feasible space $\mathcal{U}_c(t,x)$ defined in (3.3) and its volume $\mathcal{V}(t,x) = \text{vol}(\mathcal{U}_c(t,x))$. Let $H : \mathbb{R} \to \{0,1\}$ be the Heaveside unit step function such that $H(y) = 0$ for $y < 0$ and $H(y) = 1$ otherwise. Let H_s be a differentiable approximation of H such that $H_s(z) \leq H(x), \forall z \in \mathbb{R}$. Define $\mathcal{V}_s : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}$ as the following differentiable function

$$\mathcal{V}_s(t,x) = \int_{u\in\mathbb{R}^m} \left[\prod_{i=1}^{N} H_s(b_i(t,x) - A_i(t,x)u)\right] d\mathcal{V}. \tag{3.10}$$

Then $\mathcal{V}(t,x) \geq \mathcal{V}_s(t,x)$ for all $t \in \mathbb{R}^+, x \in \mathbb{R}^n$.

Proof. For brevity, we skip the explicit dependency on time in the following. The volume of the polytope (3.3) is given by

$$\begin{aligned}
\mathcal{V}(x) &= \text{vol}(\mathcal{U}_c(x)) && \text{(3.11a)}\\
&= \int_{u\in\mathcal{U}_c(x)} d\mathcal{V} = \int_{u\in\{u\in\mathbb{R}^m | A(x)u\leq b(x)\}} d\mathcal{V} && \text{(3.11b)}\\
&= \int_{u\in\mathbb{R}^m} \left[\prod_{i=1}^{N} H(b_i(x) - A_i(x)u)\right] d\mathcal{V}, && \text{(3.11c)}
\end{aligned}$$

where $d\mathcal{V}$ is the infinitesimal volume (for example, $d\mathcal{V} = du_x du_y$ when $m = 2$) Note that although H is a discontinuous function, the integral (3.11c) exists as the points of discontinuity, defined by the boundary of the polytope, form a set of measure 0. Furthermore, since $H(y) \leq H_s(y)$ for all $y \in \mathbb{R}$, we have $\mathcal{V}(t, x) \geq \mathcal{V}_s(t, x)$. □

Note that, although smooth, evaluating the integral (3.11c) analytically is still non-trivial and requires further numerical techniques such as MC approximation. Due to the sensitivity of the gradient to the number of samples chosen, we observe that MC, although suitable for obtaining volume, is not suitable for gradient computation. Therefore we only present results with ellipse and circle approximations in the ensuing. Methods to provide stable gradients through (3.11c) will be explored in future work.

3.5 Simulation Results

In this section, we present two case studies to evaluate the performance of the proposed volume barrier function. All the simulations are run on a laptop with i9-13905 CPU. The dynamics are propagated using Euler integration with a time step of 0.01s. We use Python 3.11 and JAX's autodiff feature to compute all required gradients except for the circle and ellipse approximations whose details are mentioned in the next section. Before proceeding with the results, we first mention some important implementation details of our framework.

The sphere and ellipsoid underapproximations of $\mathcal{V}(t, x)$ are computed using LP and SDP respectively, which are convex programs and, therefore, we can exploit sensitivity analysis [7] to compute the gradients of their volume w.r.t the polytope matrices A, b. Libraries like cvxpy [78] readily provide an interface to differentiate through convex programs and we use cvxpy's JAX interface in our simulations. While LP and SDPs are not always differentiable, we do not encounter observable issues in our simulations. We hypothesize that this is because the configurations that are non-differentiable, which are related to instances when a new hyperplane enters or exits the feasible space, are rarely encountered exactly in a discrete-time simulation. A more thorough analysis though is subject to future work. Note that, if any non-differentiable configuration is encountered, cvxpy returns heuristic gradients.

In all our simulations, we first obtain the polytope volume $\mathcal{V}$ and its gradients using one of the proposed methods: smooth approximation or circle/ellipsoid approximation. We then use these values to formulate the CBF conditions and solve the controller (3.6). Therefore, in the case of circles and ellipsoids, we solve two optimization problems sequentially: first, an LP/SDP, and then the FS-CBF-QP.

3.5.1 Case Study 1: Valid FS-CBF computation

In this case study, we demonstrate the concept of FS-CBF for navigating in the obstacle environment shown in Fig. 3.2. A Dubins car model is considered: $\dot{p}_x = v\cos\theta, \dot{p}_y = v\sin\theta, \dot{\theta} = u$ where p_x, p_y are position, θ is heading, v is fixed speed, and the angular velocity $u \in [-0.5, 0.5]$ is the control input. We consider the constraint functions, also called candidate CBFs, $c_i = d_i^2 - d_{min}^2$, with d_i being the distance to center of i^{th} obstacle and d_{min} the radius of obstacle. Next, we find valid CBFs h_i to each c_i using the refineCBF framework from [226] that leverages Hamilton-Jacobi (HJ) reachability analysis. Note that each valid CBF h_i does not enforce collision avoidance with an obstacle $j \neq i$. Now we present results for two cases: In Case 1, we start with valid CBFs h_i and show their compatible state space in Fig. 3.5(a). The feasible space volume (of one-dimensional control space in this example) is computed using 100 MC samples using method in Section 3.7. We once again use refineCBF to find a valid CBF on this compatible state space. The resulting FS-CBF is visualized in Fig. 3.5(b). In Case 2, we start with the candidate CBFs c_i and visualize their compatible state space in Fig. 3.5(c),(e) for two different choices of linear class-$\mathcal{K}$ function parameters. Note that c_i are higher-order barriers and their feasible space is defined by (3.1) for $r_i = 2$. Next, we find a valid CBF on compatible state space of c_i using the refineCBF framework and visualize the resulting FS-CBF in Fig. 3.5(d),(f). Observe that compatibility is highly sensitive to chosen class-$\mathcal{K}$ parameters. This presents a significant challenge in practice wherein manually tuned candidate CBFs are often employed as finding a valid CBF is not always feasible computationally.

Note that the unsafe region (function value of zero in Fig.3.5) is larger in Fig.3.5(b),(d),(f) compared to Fig.3.5(a),(c),(e). This implies that some of the states with non-zero feasible space volume are outside the found forward invariant set and may be unsafe. We thus reiterate that CBFs designed independently do not possess any guarantee of compatibility when enforced together.

3.5.2 Case Study 2: Improving Feasibility and Sensitivity

Consider the scenario shown in Fig. 3.6 for a robot navigating to its goal location g while avoiding black prohibited zones and performing collision avoidance with humans. This case study serves two purposes. First, we show that the addition of a feasible space volume barrier function enhances the compatibility of candidate CBFs. Second, Since the N CBFs are not designed to be compatible, the time at which they become infeasible depends heavily on the nominal controller that guides the QP to certain parts of state space. We show that the proposed method also helps reduce sensitivity to the hyperparameters of the nominal

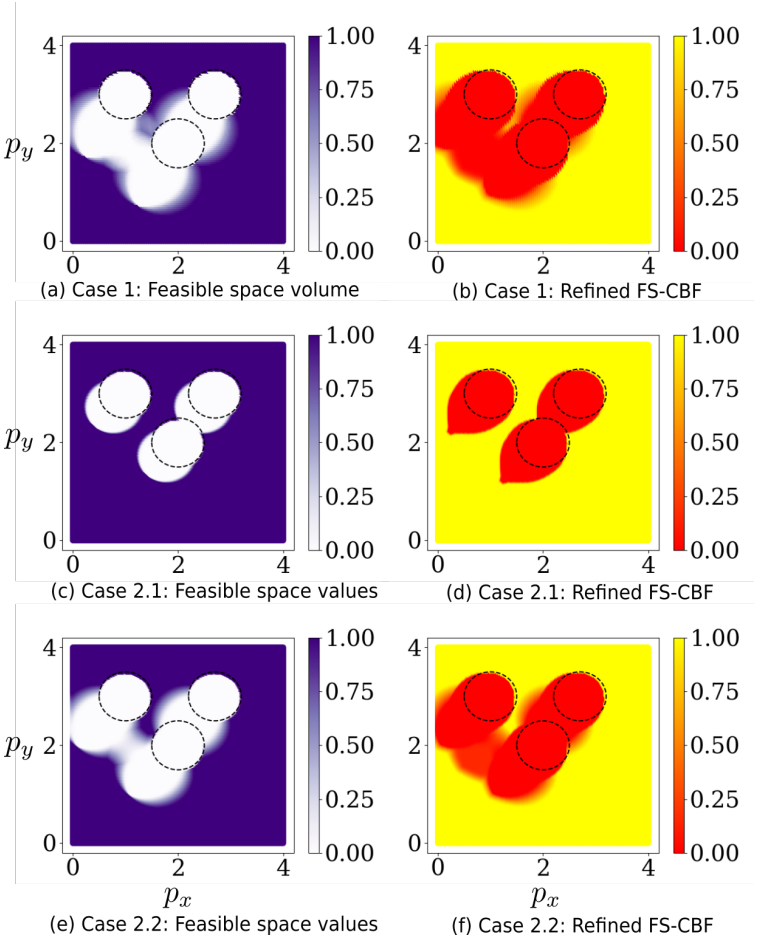
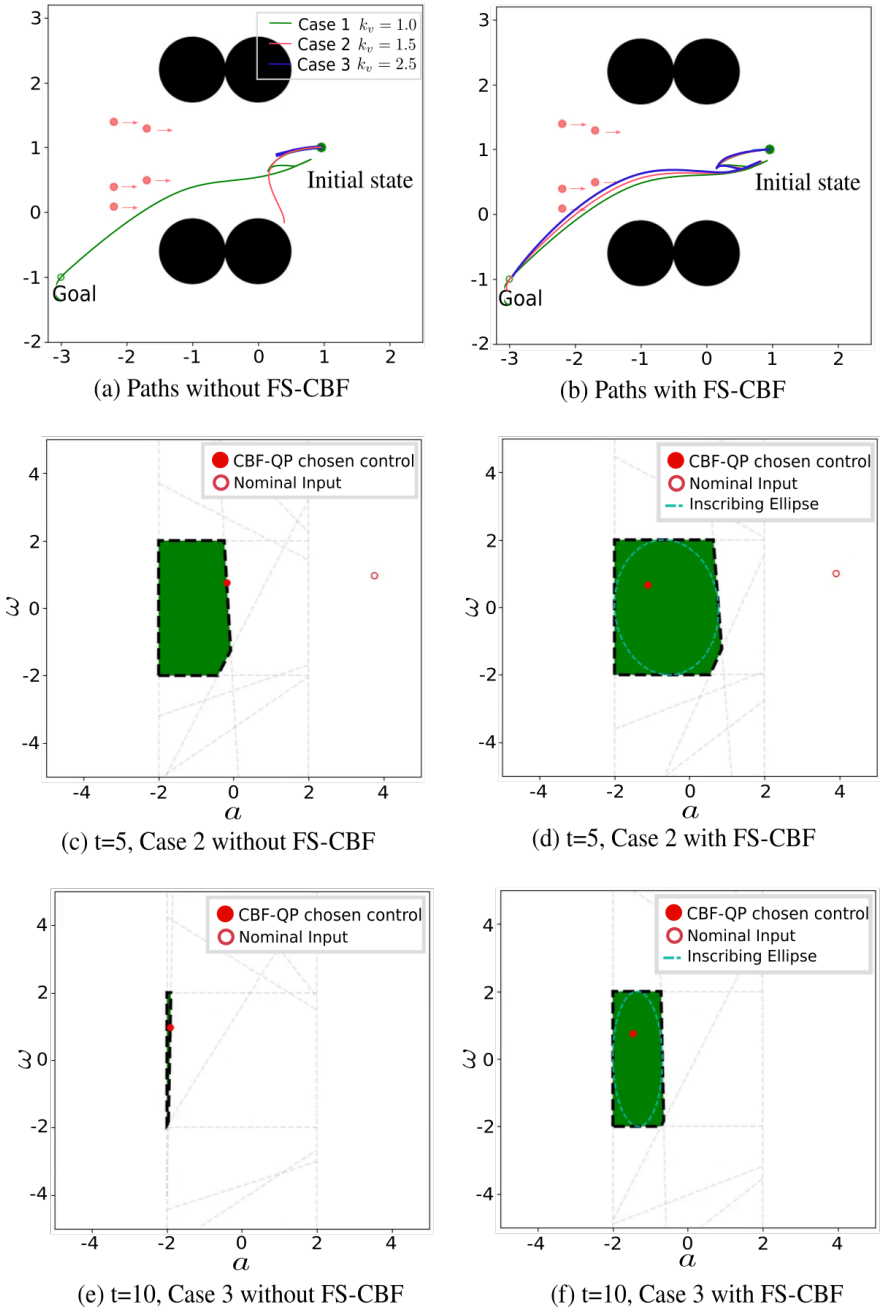


Figure 3.5: Compatible space vs refined FS-CBF values at different X, Y and heading $\theta = 45^0$. The color bar represents the numerical values of FS volume and FS-CBF clipped to the interval [0,1]. (a),(b) Case 1 - initiating with three *valid* CBFs. (c)-(f) initiating with three *candidate* CBFs with aggressive (Case 2) and conservative (Case 3) class-$\mathcal{K}$ function parameters for HOCBF. The compatible space (purple) is a subset of the forward invariant set (yellow) as also illustrated in Fig. 3.3.



(a) Paths without FS-CBF

(b) Paths with FS-CBF

(c) t=5, Case 2 without FS-CBF

(d) t=5, Case 2 with FS-CBF

(e) t=10, Case 3 without FS-CBF

(f) t=10, Case 3 with FS-CBF

Figure 3.6: Case study 1 - humans move towards the right. (a),(b) Paths for the three cases of $k_v = 1, 1.5, 2.5$. Blue and red paths in (a) end when CBF-QP becomes infeasible. (c), (d), (e), (f) Feasible space visualization with and without volume barrier function for $k_v = 1.5$ in (c), (d) and $k_v = 2.5$ in (e), (f). Note when nominal input is outside the feasible space, the volume barrier function maps it to the interior of feasible space whereas CBF-QP maps it to a point on the boundary.

controller.

The robot is modeled as a dynamic unicycle with position $p = [p_x, p_y]$, velocity v, heading angle ψ and the following dynamics $\dot{p}_x = v\cos\psi,\ \dot{p}_y = v\sin\psi,\ \dot{v} = a,\ \dot{\psi} = \omega$. The control inputs are the linear acceleration a and angular velocity ω. The humans move with social force model [105] [4], and consider the robot as one of the social agents. We consider the following reference (nominal) controller for the extended unicycle dynamics

$$\begin{aligned} \psi_{ref} &= \arctan(\frac{g_y - p_y}{g_x - p_x}),\ \psi_e = \psi - \psi_{ref},\ \omega_{ref} = k_\omega \psi_e \\ v_{ref} &= k_x ||p - g|| \cos\psi_e,\ a_{ref} = k_v(v - v_{ref}) \end{aligned} \tag{3.12}$$

where $k_\omega, k_p, k_v > 0$ are user-chosen gains. The three candidate CBFs are designed in similar fashion as in Section 3.5.1. Fig. 3.6 shows compares the performance of CBF-QP (2.30) and FS-CBF-QP (3.6) for three different values of the gain k_v. Fig. 3.6(a)-(b) show the trajectories of the robot, and Figs. 3.6(c)-3.6(e) show the variation of polytope volume under three different values of the gain k_v. We observe that when k_v is increased, that is, the nominal controller is made more aggressive, the CBF-QP becomes infeasible sooner. FS-CBF-QP on the other hand is able to maintain feasibility and the robot successfully reaches the goal as seen in Figs. 3.6(d), 3.6(f). The visualization in Fig. 3.6 also shows that the nominal control input was not mapped to the boundary of the feasible set, but to its interior as mentioned in Remark 6. The video is available on thesis website https://github.com/hardikparwana/PhD-Thesis.

The importance of our controller thus lies in enhancing the performance of a given CBF-QP controller. To further corroborate our claim, we perform extensive Monte Carlo simulations by randomizing the robot's initial state p_x, p_y, v and choice of gains k_x, k_v. We report the average simulation run times in Table 3.1. The run time is defined as the minimum of 7 seconds (maximum duration of the simulation) and the time at which CBF-QP or FS-CBF-QP becomes infeasible. We present results for small as well as large perturbarion in the initial state and gains k_x, k_v. In both the cases, we observe that FS-CBF-QP significantly outperforms CBF-QP. Note that infeasibility is not unexpected even for FS-CBF-QP as we only employ candidate CBFs and assume, for sake of implementation, that $\mathcal{V}_c = \mathcal{V}$. Since $\mathcal{V}$ is not a valid CBF, our claim is strictly limited to empirical demonstration of enhanced feasibility of QP and reduced sensitivity to user parameters. Designing a CBF for such a complicated scenario with multiple obstacle and heterogenerous agents is an active area of research and no generic solutions exist yet to the best of our knowledge.

	T_1	T_2	$\alpha_{\mathcal{V}}$	p_x	p_y	v	k_p	k_v
Scene 1	4.07 4.37 3.53	1.67	0.8 1.0 2.0	[0.9,1.1]	[0.9,1.1]	[1.2,1.4]	[0.9,1.1]	[1.5,1.6]
Scene 2	3.04 2.13 1.86	1.29	0.8 1.0 2.0	[0.9,1.1]	[0.9,1.1]	[1.2,1.4]	[0.9,1.1]	[2.5,2.6]
Scene 3	4.74 4.47 3.65	1.64	0.8 1.0 2.0	[0.8,1.2]	[0.0,1.5]	[1.0,1.5]	[0.5,3.0]	[0.5,4.0]

Table 3.1: Sensitivity analysis of time to incompatibility for small (Scene 1,2) and large (Scene 3) perturbations of the initial state p_x, p_y, v and gains k_p, k_v. Variables are sampled with uniform distribution within the specified [lower bound, upper bound].

3.5.3 Case Study 3: AWS Hospital

Next, we evaluate our FS-CBF for navigating turtlebot3 robot in AWS Hospital gazebo environment shown in Fig. 3.1. We first generate a map of the environment without humans using ROS2 navigation stack. This map is used by global planner to plan paths, as well as by the CBF-QP controller to perform collision avoidance with static obstacles. At each time, the robot finds the closest occupied grid cell in the directions $\theta = \beta 30^0, \beta \in \{1, 2, .., 12\}$, and formulates 12 CBF constraints correspondingly. The humans are moved in real-time using a modified version of social force model plugin [3]. Humans, however, are agnostic to robot's motion and do not contribute to collision avoidance. The position and velocity of humans is extracted from gazebo and made available to the robot, which then adds a CBF constraint for each human. The collision avoidance is enforced using distance as the higher-order barrier function of order two which is known to work well in the presence of a single constraint only. We use linear class-$\mathcal{K}$ functions $\alpha_1(h) = 2h, \alpha_2(h) = 6h$ for first and second-order HOCBF derivative conditions. The proposed FS CBF-QP is compared with CBF-QP in (2.30). We observe that CBF-QP leads to fewer collisions with humans by preventing the robot from getting too close to the humans in congested scenarios. A video of the experiment can be found on thesis website https://github.com/hardikparwana/PhD-Thesis.

3.6 Conclusion

We propose a new barrier function that restricts the rate-of-change of the volume of feasible solution space of CBF-QP. We also provide empirical evaluation to corroborate our theoretical results, and to show effectiveness in improving feasibility and in reducing sensitivity

to changes in other system modules such as nominal controller. Future work will involve performing experiments with robots and non-smooth control theoretic analysis of FS-CBF. Finally, monitoring feasible space volume may also help alleviate feasibility issues in other optimization based controllers such as MPC. This analysis will also be performed in future work.

CHAPTER 4

Adaptation for CBFs

From a theoretical perspective, a CBF, including the barrier function and its class-$\mathcal{K}$ function, capable of guaranteeing forward invariance of the safe set may not be unique. One CBF may allow for more aggressive motions than the other in certain parts of state space. From a practical standpoint, candidate CBFs are often employed owing to the difficulty of finding even one valid CBF. Candidate CBFs are often paired with manually chosen class-$\mathcal{K}$ functions. Depending on the choice of class-$\mathcal{K}$ functions, the candidate barrier functions may be used to make the QP controller more aggressive or conservative.

The above limitations are exacerbated when multiple CBFs and control input bounds are considered in addition to the performance objective (conservative or aggressive) as we also need to account for their compatibility. The method from Chapter 3, even if implemented ideally as proposed in theory, focuses only on feasibility and not on tuning the performance (aggressiveness and conservatives). Moreover, it may simply not be possible with maintain feasibility with given choice of class-$\mathcal{K}$ function and therefore method from Chapter 3 may not find any solution.

In this chapter, we introduce a new notion of CBFs that allows tuning the response of CBF controllers to circumvent the aforementioned issues. Note that this chapter is a standalone contribution to the thesis and does not require Chapter 3 as a pre-requisite.

4.1 Literature Review and Statement of Contributions

The notion of a control barrier function (CBF) has emerged as a tool to ensure constraint satisfaction for dynamical systems [14]. Despite significant extensions of the CBF concept to address various challenges (e.g., modeling/parameter uncertainty, input constraints, external disturbances, high relative-degree constraints) over the last few years, which this dissertation cannot cite in their entirety, there are still open challenges.

Arguably, two of the major open problems are 1) tuning the response of CBF controller

for aggressiveness without jeopardizing safety, and 2) finding a valid CBF when a single constraint is enforced, and finding valid compatible CBFs when multiple constraints are enforced. The difficulty in finding such valid CBFs typically prompts the consideration of candidate CBFs for the control design; as a result, the Quadratic Program (QP) that enforces the candidate CBF condition can become infeasible over the system trajectories. A number of practical approaches have been developed to remedy these challenges, a thorough review of which is given in Section 2.7. Here we provide a summary of more relevant works.

Among myopic approaches utilizing only the current state of adapt parameters, one standard approach [91, 266] aims to relax the CBF constraint by treating the class-$\mathcal{K}$ function parameters as an optimization variable. Subtle differences exist in some of the existing approaches. [91, 93, 241] only consider linear class-$\mathcal{K}$ functions and treat the class-$\mathcal{K}$ function parameter as an optimization variable. [266] on the other hand proposes an optimal-decay CBF-QP that multiplies an arbitrary nonlinear class-$\mathcal{K}$ function by a scalar variable that is optimized for (along with the control input) in the QP. A specific value of class-$\mathcal{K}$ parameter or class-$\mathcal{K}$ multiplicative penalty parameter may be promoted, for example, by appending the cost function of CBF-QP with a term to minimize deviation from a nominal parameter value [266]. [199] considers control non-affine dynamics and in light of the resulting loss of linearity of the CBF constraint, chooses to impose it through a penalty term in the cost function, thereby converting it into an unconstrained optimization problem.

The aforementioned adaptive approaches can be thought of as adapting *on a need basis*; i.e., they relax the CBF constraint whenever the original class-$\mathcal{K}$ function leads to infeasibility. However, persistent feasibility is not guaranteed for multiple CBF constraints in the presence of input bounds. Among the non-myopic approaches, [256] multiplies a scalar penalty to the class-$\mathcal{K}$ functions, and creates a dataset of persistently feasible and eventually infeasible parameters offline by randomly sampling parameters and simulating the system forward. Then they fit a differentiable classifier that is used to impose the resulting feasibility margin as another HOBCF constraint in the system. [142] performs offline learning to learn the best-performing parameters of the class-$\mathcal{K}$ function as a function of the state, although without consideration of feasibility. CBF-MPC is proposed in [265] wherein, instead of standard state constraints, MPC imposes CBF constraints for each state in its optimization. They show empirically that CBF and MPC complement each other - constraints help improve feasibility of MPC and at the same time optimal control with MPC helps improve feasibility of CBF constraint.

4.1.1 Contributions

In this chapter, we formalize a new notion of a Rate-Tunable Control Barrier Function (RT-CBF), which allows for parametric class-$\mathcal{K}$ functions and adaptation of their parameters online. Specifically, we introduce pointwise sufficient conditions on the rate of change of parameters for enforcing feasibility (Section 4.3.1).

We also introduce two algorithms to design parameter dynamics. The first one is a myopic algorithm that designs parameter dynamics based on current state. To the best of our knowledge, the work most similar to our notion of a CBF is presented in [255], wherein the authors introduce a time-varying *penalty parameter* that multiplies the fixed class-$\mathcal{K}$ function. The penalty parameter is allowed to change with time, but is constrained to be non-negative and hence is itself posed as higher-order CBF. The main difference between the two approaches is that we consider one additional derivative of all the parameters than what is considered in [255]. This allows us to propose a decoupled algorithm for designing the parameter dynamics and the control input compared to [255]. The differences will be elucidated further in Section 4.3.

Our second algorithm is a model-predictive approach in Section 4.4 that, given a performance metric defined as a summation of stage-wise cost, computes the sensitivity of predicted future states w.r.t. the parameters of the class-$\mathcal{K}$ function. We use these sensitivities for a Sequential Quadratic Programming (SQP) [38, 225] inspired gradient descent law to update the parameters online in direction of improving performance metric. The prediction and sensitivity computation is shown to be amenable to the backpropagation type of gradient computation over a computational graph. Infinitesimal improvements in future trajectory is also a common idea in Differential Dynamic Programming (DDP) [217] and thus we bring together elements from DDP and SQP to develop an auto-tuning mechanism for CBF controllers. While similar algorithms exploiting backpropagation on dynamical systems can be found in literature, our contribution lies in demonstrating online and real-time controller parameter update by using the computed gradients in SQP inspired update and also demonstrating one of the first, if not the first, applications to tuning of CBF based controllers for constrained systems.

4.2 Novel Formulation - Rate Tunable CBFs

We introduce the notion of a Rate-Tunable Control Barrier Function (RT-CBF), which allows for online tuning of the response of CBF-based controllers by adapting the parameters of the class-$\mathcal{K}$ function that is involved in the CBF condition. In contrast to most existing

approaches that use a fixed class-$\mathcal{K}$ function to ensure safety, we propose an online adaptation law for the parameters of class-$\mathcal{K}$ function of the CBF condition. The algorithm can also incorporate multiple higher-order CBFs and adapt their class-$\mathcal{K}$ parameters for feasibility, albeit without guarantees in theory. The simulation results verify that online adaptation helps improve the system's response in terms of the resulting trajectories being closer to a nominal reference while still being safe.

4.2.1 Problem Formulation

We use the control affine dynamics and notations introduced in Chapter 2. Some of the important equations and definitions are repeated below for the sake of convenience.

For each constraint function h_i, $i \in \{1, 2, \dots, N\}$, we consider r_i *linear parametric* class-$\mathcal{K}$ functions $\alpha_i^k : \mathbb{R}^+ \times \mathcal{A}_i^k \to \mathbb{R}^+$, $k \in \{1, \dots, r_i\}$

$$\alpha_i^k(h_i, \nu_i^k) = \nu_i^k h_i, \tag{4.1}$$

where $\nu_i^k \in \mathcal{A}_i^k \subset \mathbb{R}$, and the class of controllers with linear class-$\mathcal{K}$ functions in the CBF constraints:

$$u = \arg\min_{u \in \mathcal{U}} \quad \|u - u_r(x)\|^2 \tag{4.2a}$$

$$\text{s.t.} \quad \dot{\psi}_i^{r_i-1}(x, u) \geq -\nu_i^{r_i} \psi_i^{r_i-1}(x), \ i \in \{1, .., N\}, \tag{4.2b}$$

where $u_r : \mathbb{R}^n \to \mathbb{R}^m$ is the reference control input signal, typically designed without any regard to constraints. The optimization (4.2) is a QP when the dynamics is control-affine as in (2.2) and $\mathcal{U}$ can be expressed in the form of a polytope $Au \leq b, A \in \mathbb{R}^{q \times m}, b \in \mathbb{R}^{q \times 1}, q > 0$. For a given choice of $\nu_i^k, k \in \{1, 2, \dots, r_i\}$, a solution to (4.2) may not exist at a given x. Note that this can be the case even when the control input is unbounded, or even when each h_i is valid HOCBF by itself, but not compatible with other $h_j, j \neq i$.

Motivated by the above, we seek to design algorithms that adapt the parameters ν_i^k over time to promote the feasibility of the QP (4.2), and to tune the response of the resulting closed-loop system under (4.2) to meet performance objectives. To this end we utilize the condition (4.2b), which depends on the parameters ν_i^k through the derived barrier functions in (2.15) and (4.1), and seek adaptation laws that promote feasibility of (4.2). As an example,

for the case when $r_i = 2$, (2.15) under (4.1) reads:

$$\psi_i^0 = h_i, \tag{4.3a}$$
$$\psi_i^1 = \dot{\psi}^0 + \nu_i^1 \psi_i^0, \tag{4.3b}$$
$$\psi_i^2 = \ddot{\psi}^0 + \dot{\nu}_i^1 \psi_i^0 + \nu_i^1 \dot{\psi}_i^0 + \nu_i^2 (\psi_i^0 + \nu_i^1 \psi_i^0), \tag{4.3c}$$

yielding that the vector of the parameters of (4.3) to be tuned is $\Theta_{\alpha_i} = [\nu_i^1, \dot{\nu}_i^1, \nu_i^2]^T$. The dynamics of the parameters obey:

$$\underbrace{\begin{bmatrix} \dot{\nu}_i^1 \\ \ddot{\nu}_i^1 \\ \dot{\nu}_i^2 \end{bmatrix}}_{\dot{\Theta}_{\alpha_i}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Theta_{\alpha_i} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_{\ddot{\nu}_i^1} \\ z_{\dot{\nu}_i^2} \end{bmatrix}}_{f_{\alpha_i}(\Theta_{\alpha_i}, z_i)}, \tag{4.4}$$

where $z_i = [z_{\ddot{\nu}_i^1}, z_{\dot{\nu}_i^2}]^T$ is the parameter control input vector, which we design in Section 4.3.1. Before presenting the adaptation law in Section 4.3, we first present in Sections 4.2.2 and 4.2.3 the class of Rate-Tunable (RT)-CBFs to encode the parameter adaptability, as well as some formal motivation on why adaptation is needed for performance satisfaction.

4.2.2 Motivation for varying class-$\mathcal{K}$ functions

We motivate the adaptation of the class-$\mathcal{K}$ functions by illustrating how the class-$\mathcal{K}$ functions affect the response of the CBF-QP controller. The next theorem demonstrates that for different desired control inputs $u_d(x)$ (i.e., for different desired control inputs that render the system response e.g., either more conservative or more aggressive), the parameter ν of the class-$\mathcal{K}$ function can be chosen to steer the solution u_{QP} of the CBF-QP given by (4.2) close to u_d. For simplicity, we consider a constraint function with relative degree one w.r.t. the system dynamics in the following theorem.

Theorem 4. Consider the system (2.2), a first-order candidate barrier function $h : \mathcal{X} \to \mathbb{R}$ and a set $\mathcal{S}$ defined as in (2.3). Assume that $L_g h(x) \neq 0,\ \forall x \in \mathcal{S}$, and let a parameter $\nu = \nu(x) \in \mathbb{R}$, and [1] the following CBF-QP with unbounded control input:

$$u_{QP} = \min_{u \in \mathbb{R}^m} \quad \|u - u_r(x)\|^2 \tag{4.5a}$$
$$\text{s.t.} \quad L_f h(x) + L_g h(x) u \geq -\nu h(x), \tag{4.5b}$$

[1] The notation is used to signify that the value of ν depends on x.

where $u_r : \mathbb{R}^n \to \mathbb{R}^m$ is a reference control input signal. Let $u_d : \mathbb{R}^n \to \mathcal{U}$ be a desired control input signal that is by construction safe, and $u_r(x) \neq u_d(x),\ \forall x \in \mathcal{X}$. Let $\nu^* = \nu^*(x)$ for $x \in \text{Int}(\mathcal{S})$ be given as:

$$\nu^* = \frac{1}{h}\begin{cases} L_gh(u_r - u_d) - & \text{if } L_gh(u_r - u_d) < 0 \quad (4.6a) \\ -L_fh - L_gh u_r & \\ & \\ L_gh(u_r - u') - & \text{if } L_gh(u_r - u_d) \geq 0 \quad (4.6b) \\ -L_fh - L_gh u_r, & \end{cases} \tag{4.6}$$

where

$$u' = u_d + \frac{L_gh^T L_gh(u_r - u_d)}{\|L_gh\|^2}. \tag{4.7}$$

Then ν^* satisfies:

$$\nu^*(x) = \min_{\nu} \|u_d(x) - u_{QP}(x, \nu)\|. \tag{4.8}$$

Proof. The solution of the optimization problem (4.5), assuming it exists, is found analytically (see Appendix A.1):

$$u_{QP} = \begin{cases} u_r - \frac{L_gh^T}{\|L_gh\|^2}\psi^1(\nu) & \text{if } \psi^1 < 0 \quad (4.9a) \\ & \\ u_r & \text{if } \psi^1 \geq 0 \quad (4.9b) \end{cases} \tag{4.9}$$

where

$$\psi^1(\nu) \triangleq L_fh + L_gh u_r + \nu h. \tag{4.10}$$

Using (4.9), one can write $u_d - u_{QP}$ as follows:

$$u_d - u_{QP} = \begin{cases} u_d - u_r + \frac{L_gh^T}{\|L_gh\|^2}\psi^1, & \text{if } \psi^1 < 0 \quad (4.11a) \\ u_d - u_r & \text{if } \psi^1 \geq 0 \quad (4.11b) \end{cases} \tag{4.11}$$

Case 1: Let $L_gh(u_r - u_d) < 0$. Then, substituting ν^* from (4.6a) into ν in (4.10) yields that $\psi^1(\nu^*) < 0$. Therefore, $u_d - u_{QP}$ is given by (4.11a). Next, to show that (4.6a) minimizes $\|u_d - u_{QP}\|$, first note from (4.11a) that νh only affects the component parallel to L_gh. Therefore, we show that ν^* in (4.6a) minimizes the norm of the component of $u_d - u_{QP}$ along

$L_g h^T$. Multiplying both sides of (4.11a) by $\frac{L_g h}{\|L_g h\|}$ yields:

$$\frac{L_g h}{\|L_g h\|}(u_d - u_{QP}) = \frac{L_g h}{\|L_g h\|}(u_d - u_r) + \frac{1}{\|L_g h\|}\left(L_f h + L_g h u_r + \nu h\right). \tag{4.12}$$

Substituting ν in (4.12) with ν^* from (4.6a), we obtain: $\frac{L_g h}{\|L_g h\|}(u_d - u_{QP}) = 0$, and therefore $||u_d - u_{QP}||$ is minimized (again noting that component of $u_d - u_{QP}$ perpendicular to $L_g h$ cannot be affected by any choice of ν).

Case 2: $L_g h(u_r - u_d) \geq 0$: In this case, u_d cannot be a minimizer of (4.5) irrespective of the choice of ν, since if u_d satisfies (4.5b), then u_r also satisfies (4.5b) and so the solution of (4.5) is u_r. We can aim to tune the value of ν so that the solution of the QP (4.5) converges to a control input $u' \in \mathcal{U}$ such that $\|u' - u_d\| < \|u_r - u_d\|$ and $L_g h u' > L_g h u_r$. If such a vector exists, it can be found as the solution to:

$$\min_{u'} \quad ||u' - u_d||^2 \tag{4.13a}$$

$$\text{s.t.} \quad L_g h u' \geq L_g h u_r. \tag{4.13b}$$

The solution of the above QP can also be found analytically to be given as:

$$u' = \begin{cases} u_d + \frac{L_g h^T}{\|L_g h\|^2}\chi^1(\nu) & \text{if } \chi^1 \geq 0 \quad (4.14a) \\ \\ u_d & \text{if } \chi^1 < 0 \quad (4.14b), \end{cases} \tag{4.14}$$

where $\chi^1 = L_g h u_r - L_g h u_d$. Since we have assumed that $L_g h(u_r - u_d) \geq 0$, we have that $u' = u_d - \frac{L_g h^T}{\|L_g h\|^2}\chi^1(\nu)$, i.e., given by (4.7). Then, following the same reasoning as in Case 1, ν^* in (4.6b) can be shown to minimize $\|u_d - u_{QP}\|$. □

Remark 7. We clarify that (4.6) is not the proposed algorithm for tuning the RT-CBF parameters, since this formula holds only for first-order barrier functions w.r.t. the system dynamics. The theorem is only to demonstrate that tuning the parameter of the constraints affects the solution of the QP to be steered closer to a desired response.

4.2.3 Introducing RT-CBFs

We now formally introduce our notion of RT-CBFs. Consider the control-affine system dynamics in (2.2) and let the set of allowable states $\mathcal{S}_i$ be defined as the zero-superlevel set

of a function $h_i : \mathcal{X} \to \mathbb{R}$ as in (2.3). We define functions ψ_i^k as:

$$\psi_i^0(x) = h_i(x), \tag{4.15a}$$

$$\psi_i^k(x) = \dot{\psi}_i^{k-1}(x, \Theta_{\alpha_i<k}) + \nu_i^k \psi_i^{k-1}(x, \Theta_{\alpha_i<k}),$$
$$k \in \{1, 2, .., r_i\}, \tag{4.15b}$$

where Θ_{α_i} is the vector comprising the parameter ν_i^k in (4.15) and its $r_i - k$ derivatives (see (4.4)), for all $k \in \{1, 2, \ldots, r_i\}$, and $\Theta_{\alpha_i<k}$ denotes the sub-vector of parameters and derivatives with index lower than k.

Consider the augmented dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{\Theta}_{\alpha_i} \end{bmatrix} = \begin{bmatrix} f(x) + g(x)u \\ f_{\alpha_i}(\Theta_{\alpha_i}, z_i) \end{bmatrix}, \tag{4.16}$$

where $f_{\alpha_i} : \mathcal{A}_i \times \mathbb{R}^{n_{z_i}} \to \mathcal{A}_i$ results from the integrator dynamics of ν_i^k as in (4.4), with $\mathcal{A}_i \subset \mathbb{R}^{n_{\alpha_i}}$ a compact set, $n_{\alpha_i} = \frac{r_i(r_i+1)}{2}$, $z_i \in \mathbb{R}^{n_{r_i}}$ is the vector of the control inputs of the parameters and $i \in \{1, \ldots, N\}$.

Definition 9. (Rate-Tunable CBF) Assume that the function f_{α_i} in (4.16) is such that the (unique) solution $\Theta_{\alpha_i}(t) \in \mathcal{A}_i, \forall t \geq 0$ where $\mathcal{A}_i \subset \mathbb{R}^{n_{\alpha_i}+}$. Let $u(t) \in \mathcal{U}, t \geq 0$ represent any arbitrary control input trajectory. The constraint function $h_i : \mathcal{X} \to \mathbb{R}$ is a Rate-Tunable Control Barrier Function (RT-CBF) on the set $\mathcal{S}_i$ under the augmented system (4.16) if there exists an open set $\mathscr{D}_i \subset \mathcal{S}_i$ such that for every $x(0) \in \mathscr{D}_i$ there exists $\Theta_{\alpha_i}(0) = \Theta_{\alpha_i}(x(0)) \in \mathcal{A}_i$, and a Lipschitz continuous control input signal $z : [0, \infty) \to \mathbb{R}^{r_i}$ such that:

$$\sup_{u(t)\in\mathcal{U}, t\geq 0} \Big[\min_{t\geq 0} [\dot{\psi}_i^{r_i-1}(x(t), u(t), \Theta_{\alpha_i<r_i}(t)) + \nu_i^{r_i}(t)\psi_i^{r_i-1}(x(t), \Theta_{\alpha_i<r_i}(t))] \Big] \geq 0, \tag{4.17}$$

where the sup is over the space of control input trajectories.

Theorem 5 shows that a necessary condition for strict safety ($h_i(x) > 0$) is that any constraint function h must be a RT-CBF. We leave the boundary case to future work for the sake of simplicity and avoid an analysis of the system in the limit of $h_i(x) \to 0$.

Theorem 5. Consider the dynamics (2.2) and a constraint function h_i and its 0-superlevel set $\mathcal{S}_i$ as defined in (2.3). Let $x_0 \in \mathcal{S}_i$ be an arbitrary initial state. Then there exists a strictly safe trajectory $x(t), u(t), t \geq 0$ with $x(0) = x_0, h_i(x(0)) > 0, u(t) \in \mathcal{U}$ only if there

exists $r_i - k + 1$-differentiable trajectories $\nu_i^k(t) > 0, \forall t \geq 0, k \in \{1, .., r_i\}$ such that (4.17) is satisfied.

Proof. Consider any arbitrary safe trajectory $x(t), u(t), t \geq 0$, if it exists, starting at state $x(0)$. Since h_i has relative degree r_i, $h^{(r_i-1)(x(t))}$ exists. We prove the claim by presenting a procedure to construct parameter trajectory $\nu_i^k(t), t \geq 0$ such that (4.17) holds for the arbitrarily chosen safe trajectory. For $k = 1$, let $\bar{\nu}_i^1(t) : \mathbb{R}^+ \to \mathbb{R}+$ be defined as

$$\bar{\nu}_i^1(t) = \max \left\{ -\frac{\dot{h}(x(t))}{h(x(t))}, \epsilon \right\}, \tag{4.18}$$

where $\epsilon > 0$ is chosen to construct a $\psi_i^1(t) > 0, \forall t \geq 0$ as will be explained below. Let $\bar{\nu}_{i_s}^1(t)$ such that $\bar{\nu}_{i_s}^1(t) > 0, \forall t \geq 0$ be the smooth overapproximation[2] of $\bar{\nu}_i^1(t)$. $\bar{\nu}_{i_s}^1(t)$ is at least $r_i - 1$ times differentiable since $\dot{h}_i$ is at least $r_i - 1$ times differentiable (further gradients depend on regularity properties of u). Further, let $\nu_i^i(t)$ with $\nu_i^i(t) > 0, \forall t \geq 0$ be r_i differentiable overapproximation[3] of $\bar{\nu}_{i_s}^1(t)$. With this choice of ν_i^1, we have that $\psi_i^1(t) > 0, \forall t \geq 0$ and we can repeat the above procedure sequentially to construct $r_i - k + 1$ differentiable trajectories $\nu_i^k(t) > 0, \forall k \in \{2, .., r_i\}$. □

Remark 8. Note that while $h_i(x(t))$ is guaranteed to be $r_i - 1$ differentiable, $h_i^{r_i}(x(t), u(t))$ depends on control input and therefore shares its regularity properties. In the proof of Theorem 5, equality thus holds, that is $\nu_i^k(t) = \bar{\nu}_i^k(t)$, only if $u(t)$ is also a differentiable w.r.t t. Therefore, a smartly designed $\Theta_{\alpha_i}(t)$ could be used to synthesize smoother control inputs even in the presence of multiple barrier functions. Further exploration in this direction is subject to future work.

Remark 9. Note that RT-CBF constraint (4.17) reduces to the definition of HOCBF (2.30b) under the special case when $\Theta_{\alpha_i}(t) = \Theta_{\alpha_i}(0)$, where $\Theta_{\alpha_i}(0)$ is a constant independent of $x(0)$ and the set $\mathscr{D}_i = \mathscr{C}_i$ defined in (2.16).

4.3 Continuous time Pointwise Optimal Adaptation for CBFs

Finding valid parameter trajectories $\nu_i^k(t)$ for a RT-CBF is as difficult, if not more, as finding valid CBFs and HOCBFs. In the ensuing, we only provide a myopic algorithm for designing

[2]This is always possible. See [154][Theorem 4] for an example procedure to construct a smooth overapproximation with guaranteed bounds.

[3]This is also always possible as $\nu_i^i(t) = c$ where $c \geq \sup_{t \geq 0}[\bar{\nu}_{i_s}^1(t)]$ satisfies this condition.

$\nu_i^k(t)$, and more optimal designs is subject to future work. We emphasize that the usefulness of RT-CBF is in customizing the safety response of the system to different initial states by tuning the parameters ν_i^k.

4.3.1 Strategy for designing point-wise sufficient condition

Given the augmented system (4.16), the objective is to find the control inputs $u \in \mathcal{U}$ and the parameter control inputs $z_i \in \mathbb{R}^{n_{z_i}}$. Algorithm 1 presents a myopic update law z_i for the parameters Θ_{α_i} to enforce feasibility of (4.2). To provide some physical intuition before presenting the main algorithm, we present the steps of the approach when $r_i = 2$ as an example.

Example 2. Consider the system (2.2), a function $h_i : \mathcal{X} \to \mathbb{R}$ with relative degree $r_i = 2$ w.r.t. (2.2), and a set $\mathcal{S}_i$ as in (2.3). Then the derived barriers are given by (4.3). Aiming to define an algorithm that imposes (4.17) , we note the following. For $k = 0$, we have $\psi_i^0 = h_i \geq 0$. For $k = 1$, we impose the condition $\psi_i^1 \geq 0$, which from (4.3b) yields:

$$\nu_i^1 \geq -\frac{\dot{h}_i}{h_i}. \tag{4.19}$$

Similarly, for $k = 2$, we impose the condition $\psi_i^2 \geq 0$, and (4.3c) yields that ν_i^2 should satisfy:

$$\nu_i^2 \geq -\frac{\sup_{u\in\mathcal{U}} \ddot{h}_i(x,u) - \nu_i^1 \dot{h}_i - \dot{\nu}_i^1 h_i}{\dot{h}_i + \nu_i^1 h_i}. \tag{4.20}$$

We then define the system of parameter dynamics (4.4), and we seek to design the control inputs $z_{\dot{\nu}^1}$ and $z_{\dot{\nu}^2}$ so that (4.19) and (4.20) are enforced. More generally, for the r_i barrier functions (2.15), the proposed conditions read:

- For $k \in \{1, .., r_i - 1\}$, we seek to enforce that:

$$\nu_i^k \geq -\frac{\dot{\psi}_i^{k-1}(x, \Theta_{\alpha_{i<k}})}{\psi_i^{k-1}(x, \Theta_{\alpha_{i<k}})}. \tag{4.21}$$

- For $k = r_i$, the parameter depends on the control input. For each barrier constraint h_i, we impose

$$\nu_i^{r_i} \geq -\frac{\sup_{u\in\mathcal{U}} \dot{\psi}_i^{r_i-1}(x, u, \Theta_{\alpha_i<r_i})}{\psi_i^{r_i-1}(x, \Theta_{\alpha_i<r_i})}. \tag{4.22}$$

Proposition 1. Consider a r_i-th order function $h_i : \mathcal{X} \to \mathbb{R}$ defining sets $\mathcal{S}_i$ as in (2.3). Let the augmented system (4.16), and define r_i functions as in (4.15) and sets $\mathscr{C}_i$ as in (2.16). If for a set $\mathscr{D}_i \subset \mathcal{S}_i$, there exists $u \in \mathcal{U}$ and $z_i \in \mathbb{R}^{n_{z_i}}$ that enforce (4.21) and (4.22) everywhere on $\mathscr{D}_i$, then h_i is a RT-CBF on $\mathcal{S}_i$ under (4.16).

Proof. Follows directly from the definition of RT-CBF. □

The objective now is to design $z_i \in \mathbb{R}^{n_{z_i}}$ and $u \in \mathcal{U}$ to enforce the conditions (4.21) and (4.22). First, we observe the following: For the r_i-th order barrier function h_i, the $r_i - k + 1$ derivative $(\nu_i^k)^{r_i-k+1}$ of the parameter ν_i^k can be written in terms of parameter control input z_i as

$$(\nu_i^1)^{r_i} = z_{(\nu_i^1)^{r_i}}, \tag{4.23a}$$
$$(\nu_i^2)^{r_i-1} = z_{(\nu_i^2)^{r_i-1}}, \tag{4.23b}$$
$$\cdots \tag{4.23c}$$
$$(\nu_i^{r_i})^1 = z_{(\nu_i^{r_i})^1}, \tag{4.23d}$$

where $z_i = [z_{(\nu_i^1)^{r_i}}, z_{(\nu_i^2)^{r_i-1}}, .., z_{(\nu_i^{r_i})^1}] \in \mathbb{R}^{r_i}$ stands for the *parameter control input* to be designed using Algorithm 1. As an example, for $r_i = 2$, the state-space form of the system above takes the form of (4.4).

Second, to design the parameter control input vectors z_i, one approach could be to consider (4.21), (4.22) as additional candidate CBFs within the overall QP (4.2) (similar to [255]). However, such an approach would only introduce more parameters (resulting from the class-$\mathcal{K}$ functions of the new CBF candidates) to be tuned.

4.3.2 An algorithm with Zero-Order-Hold implementation

To avoid this, we propose an alternative way that enforces (4.21) and (4.22) using a sampled-data approach, namely a Zero-Order-Hold (ZOH) approach, as follows.

Let the control input u be computed by (4.2) at times t_j, $j \in \{0, 1, \dots\}$ and be held constant during a time horizon $\Delta t > 0$, i.e., $u(t) = u(t_j), \forall t \in [t_j, t_j + \Delta t), \forall j \in \{0, 1, \dots\}$. The state $x(t_j + \Delta t)$, i.e., the value of the state x at time $t_j + \Delta t$, is computed from control-affine dynamics (2.2) under the (step) control command $u(t_j)$ resulting from the solution of (4.2) at time t_j.

The objective of the parameter adaptation at each time t_j is to ensure that the value of the parameters $\nu_i^k(t_j + \Delta t)$ at time $t_j + \Delta t$ are such that either the CBF conditions (4.21) are enforced (safety), or such that they track a desired $\nu_{i,d}^k(t_j + \Delta t) \geq 0$ that is a priori specified by the user (performance).

Then, at time t_j:

- For $k \in \{1, .., r_i - 1\}$, we set the target value $\nu_i^k(t_j + \Delta t)$ as:

$$\nu_i^k(t_j + \Delta t) = \max \left\{ \underbrace{\lambda \max \left\{ -\frac{\dot{\psi}_i^{k-1}(x(t_j + \Delta t))}{\psi_i^{k-1}(x(t_j + \Delta t))}, 0 \right\}}_{B}, \nu_{i,d}^k(t_j + \Delta t) \right\}, \tag{4.24}$$

where $\lambda > 1$ is a tuning parameter to avoid satisfying the constraint (4.21) on the boundary of the constraint set. Note that the rule (4.24) for choosing the desired values for ν_i^k enforces safety via the term B in the maximization function: if the lower bound in (4.21) is positive, then it is multiplied by $\lambda > 1$ to make it even stricter, i.e., enforce the constraint in the interior of the safe set, while if the lower bound is negative then the desired value is set to zero, which has a similar effect in making the condition stricter. Then, if the user-defined value $\nu_{i,d}^k$ is greater than the lower bound B for safety, i.e., is already safe, we choose this as the target value, otherwise the lower bound B for safety is applied. We design $\nu_{i,d}^k(t_j + \Delta t)$ to promote bounded change with respect to $\nu_i^k(t_j)$, that is,

$$|\nu_{i,d}^k(t_j + \Delta t) - \nu_i^k(t_j)| \leq \Delta_i^k, \tag{4.25}$$

where $\Delta_i^k > 0$ is also an important tuning parameter in our algorithm. If a user specifies nominal $\nu_{i,nom}^k$, then $|\nu_{i,d}^k(t_j + \Delta t)$ is designed to steer towards $\nu_{i,nom}^k$ while respecting Δ_i^k bound.

Then, we design the derivative $(\nu_i^k)^{r_i - k + 1}(t_j)$ at time t_j to achieve the value $\nu_i^k(t_j + \Delta t)$ at time $t_j + \Delta t$ given by (4.24). Since the ν_i^k dynamics is defined by an integrator model, its analytical solution is known to be

$$\begin{aligned} \nu_i^k(t_j + \Delta t) =& \nu_i^k(t_j) + (\nu_i^k(t_j))^{(1)} \Delta t + (\nu_i^k(t_j))^{(2)} \frac{\Delta t^2}{2} + ... \\ &+ (\nu_i^k(t_j))^{(r_i - k + 1)} \frac{\Delta t^{(r_i - k + 1)}}{(r_i - k + 1)!}. \end{aligned} \tag{4.26}$$

This motivates the following design of $z_{(\nu_i^k)^{r_i-k+1}}$:

$$z_{(\nu_i^k(t_j))^{(r_i-k+1)}} = \frac{(r_i-k+1)!}{\Delta t^{(r_i-k+1)}}\Bigg(\nu_i^k(t_j+\Delta t) - \nu_i^k(t_j) - \sum_{p=1}^{r_i-k}(\nu_i^k(t_j))^{(p)}\frac{\Delta t^p}{p!}\Bigg). \tag{4.27}$$

- For $k = r_i$, the CBF condition for each barrier constraint h_i, $i \in \{1, 2, \dots, N\}$ is given by (4.22). Taking all constraints into account, the lower bound of $\nu_i^{r_i}(t_j + \Delta t)$ is taken as the solution to the following linear program to enforce compatibility:

$$\min_{\nu_i^{r_i}, u\in\mathcal{U}} \quad \sum_{i=1}^{N}\nu_i^{r_i} \tag{4.28a}$$

$$\text{s.t.} \quad \dot{\psi}_i^{r_i-1}(x(t_j+\Delta t), u, \Theta_{\alpha_i}(t_j+\Delta t)) \geq -\nu_i^{r_i}\psi_i^{r_i-1}(x(t_j+\Delta t), \Theta_{\alpha_i}(t_j+\Delta t)) \tag{4.28b}$$

$$\nu_i^{r_i} \geq 0, i \in \{1, .., N\}. \tag{4.28c}$$

Representing the solution of (4.28) as $\bar{\nu}_i^{r_i}(t+\Delta t)$, we formulate $\nu_i^{r_i}(t+\Delta t)$ as follows

$$\nu_i^{r_i}(t+\Delta t) = \max\left\{\bar{\nu}_i^{r_i}(t+\Delta t), \nu_{i,d}^{r_i}(t_j+\Delta t)\right\}, \tag{4.29}$$

and the final derivative is then designed as

$$z_{(\nu_i^{r_i})^1} = \frac{\nu_i^{r_i}(t+\Delta t) - \nu_i^{r_i}(t)}{\Delta t}. \tag{4.30}$$

Proposition 2. Consider N r_i-th order function $h_i : \mathcal{X} \to \mathbb{R}$, $i \in \{1, 2, \dots, N\}$, defining sets $\mathcal{S}_i$ as in (2.3). For each constraint h_i, define r_i functions as in (4.15). Define an augmented system of the form (4.16), where the subsystem of parameter dynamics is defined as the concatenation of N systems of the form (4.23) for $i \in \{1, \dots, N\}$. Denote the time instances t_j, $j \in \{0, 1, \dots\}$, the states $x(t_j), \Theta_{\alpha_i}(t_j)$ at time t_j, and let the control input $u(t_j)$ be computed using (4.2). Let $x(t_j + \Delta t)$ be the state at time $t_j + \Delta t$ computed under zero-order-hold control input $u(t) = u(t_j)$, $t \in [t_j, t_j + \Delta t)$ (Steps 4,5). For each constraint h_i, at time t_j, set the parameter control input to be given by (4.27) (Steps 6-9) and compute the corresponding components of $\Theta_{\alpha_i}(t_j + \Delta t)$ for $k \in \{1, \dots, r_i - 1\}$ (Steps 11-13). Assume further that the LP (4.28) is feasible at time t_j, and compute $z_{(\nu_i^{r_i})^1}$ by (4.30) for $k = r_i$ (Step 14). Then the functions h_i are RT-CBFs, and Algorithm 1 tunes their parameters to

enforce safety for the system trajectories, i.e., $x(t) \in \mathcal{S}_i, \quad \forall t \geq 0$.

Proof. Follows from the definition of RT-CBFs and by construction of Algorithm 1. □

Algorithm 1 RT-CBF Pointwise Optimal Controller

Require: $t, x(t), \nu(t), N$ ▷ current time, current state, current parameter, No. of constraints
1: **for** i=1 to N **do** ▷ Compute all derived barriers
2: Compute $\dot{\psi}_i^{r_i-1}, \psi_i^{r_i}$
3: **end for**
4: Compute $u(t)$ using (4.2) ▷ Implement u on system (2.1)
5: Compute $x(t+\Delta t)$ using $u(t)$ and (2.1)
6: **for** i = 1 to N **do** ▷ Update class-$\mathcal{K}$ parameters
7: **for** $k = 1$ to $r_i - 1$ **do** ▷ for all derived barriers
8: Compute $z_{(\nu_i^k)^{r_i-k+1}}$ using (4.27)
9: **end for**
10: **end for**
11: **for** i=1 to N **do**
12: Compute ν_i^k and its derivatives at time $t+\Delta t$ for $k \leq r_i - 1$ using $z_{(\nu_i^k)^{r_i-k+1}}$ from Step 8.
13: **end for**
14: Compute $\dot{\nu}_i^{r_i}$ using (4.30) and compute $\nu_i^{r_i}(t+\Delta t)$.

Remark 10. PACBF introduced in [255] considers only $r_i - k$ derivatives of the parameters of function α_i^k whereas we consider $r_i - k + 1$ derivatives. That is, for a second order barrier function and linear class-$\mathcal{K}$ functions in the example (4.3), PACBF only designs $\dot{\nu}_i^1$ and, similar to works in [265, 91, 93], allows ν_i^2 to be an optimization variable in the QP. Whereas we design $\ddot{\nu}_i^1$ and $\dot{\nu}_i^2$. As a consequence, since $\dot{\psi}_i^1$ is dependent on $\dot{\nu}_i^1$, PACBF simultaneously optimizes for the control input u and the derivative $\dot{\nu}_i^1$ in a single optimization problem. In contrast, in our case, (4.2) is independent of $\ddot{\nu}_i^1$ and $\dot{\nu}_i^2$ and hence is optimized only for u.

4.3.3 Simulation Results

We present simulation studies to show the efficacy of our proposed algorithm. Code and videos available at: https://github.com/hardikparwana/RateTunableCBFs. Videos are also available on thesis website https://github.com/hardikparwana/PhD-Thesis.

4.3.3.1 Pointwise Optimal Controller

Consider the scenario in Fig. 4.1 that is reminiscent of a traffic interaction, with an ego agent (green) being surrounded by two non-ego agents (black). Assume that the position

trajectories of the non-ego agents, denoted $x_i(t), i \in \{1, 2\}$ and represented by the two black lines in Fig. 4.1, are known to the ego agent, whose position trajectories are denoted as $x(t)$, and are such that the respective inter-agent distances $\|x(t) - x_i(t)\|$ decrease during some time interval and remain constant thereafter. We compare the results of a standard CBF-QP, with the RT-CBF-QP that implements Algorithm 1. The candidate barrier function used is $h_i = (x - x_i)^2$, where x is the position of the ego agent and x_i is the position of the non-ego agent i. We simulate two different models for the ego agent: 1) single integrator (SI), so that the barrier functions h_1, h_2 are of first-order w.r.t. the SI dynamics, and 2) double integrator (DI), so that the barrier functions h_1, h_2 are of second-order w.r.t. the DI dynamics. The reference controller is chosen as $u_r(x) = -k_x(x-6.0)$ for SI, and $u_r(x) = k_v(\dot{x}+k_x(x-6.0))$ for DI where $k_x = 1.0, k_v = 2.0$ are controller gains. In all our simulations, the fixed parameter value in CBF-QP is also the initial value of parameters for RT-CBF-QP.

Fig. 4.1 illustrates the position trajectories under the parameter adaptation. The trajectories $x_i(t)$ end either at the point of infeasibility, or when the simulation ends at $t = 5.5s$. The sensitivity of the initial location is further analyzed in Fig. 4.2 that shows the variation in time-to-failure for different initial positions but the same class-$\mathcal{K}$ parameters. For DI dynamics, we set the initial velocity equal to zero. Failure occurs when either the RT-CBF-QP becomes infeasible, or when the parameters $\nu_i^k \to \infty$ (which implies that $h_i \to 0$). Note that the region of the state space that defines valid initial conditions is larger for RT-CBFs compared to classical CBFs.

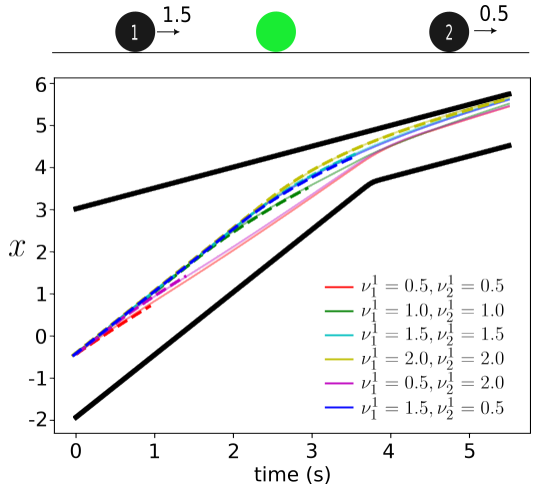


Figure 4.1: Trajectories with different initial parameter values without adaptation (dotted lines) and with adaptation (solid lines) for initial state $x = -0.5$. Dotted lines end when the underlying problem becomes infeasible. The solutions of the proposed method remain feasible until the end of the simulation horizon (5.5 seconds).

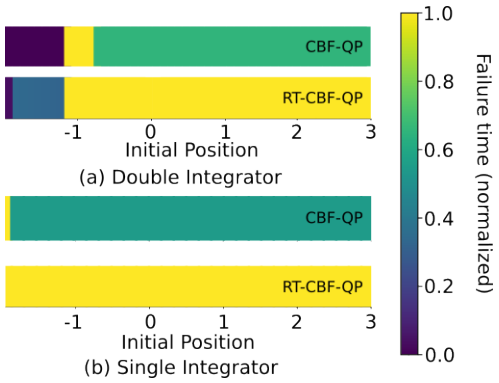


Figure 4.2: Time-to-failure as a function of the initial location $x_0 \in [-1.95, 3.0]$ of the double integrator agent. The initial velocity is zero. Algorithm 1 increases the time horizon over which the controller is feasible.

4.3.3.2 Adaptive Cruise Control

We simulate the Adaptive Cruise Control (ACC) problem introduced in [15] but with an additional constraint on the ego vehicle's velocity as in [255], and while considering a decelerating leader. Consider an ego agent moving with velocity v and a leader agent moving at velocity v_L with distance D between them. The system dynamics is:

$$\begin{bmatrix} \dot{v} \\ \dot{v}_L \\ \dot{D} \end{bmatrix} = \begin{bmatrix} \frac{-1}{M} F_r(v) + \frac{1}{M} u \\ a_L \\ v_L - v \end{bmatrix}, \tag{4.31}$$

where M denotes the mass of ego agent, $F_r(v)$ is the resistance force defined as $F_r(v) = f_0 \text{sign}(v) + f_1 v + f_2 v^2$ with $f_0, f_1, f_2 > 0$, and a_L is the acceleration of the leader. The safety objective of the ego agent is stay a minimum distance $\bar{D}$ away from the leader. The desired velocity is given by v_d. The velocity and control input constraints are:

$$v_{min} \leq v \leq v_{max} \quad , \quad -cMg \leq u \leq cMg, \tag{4.32}$$

where c is acceleration coefficient and g is acceleration due to gravity. To ensure safety, we define three barrier functions:

$$h_1 = D - \bar{D}, h_2 = v - v_{min}, h_3 = v_{max} - v, \tag{4.33}$$

where h_1 is second-order w.r.t. the system dynamics, whereas h_2, h_3 are first-order. The CLF is chosen as $V = (v - v_d)^2$. We further choose linear class-$\mathcal{K}$ functions as in (4.1) and apply Algorithm 1 to ensure safety. The following parameters are used for the simulation:

$M = 1650$ kg, $f_0 = 0.1, f_1 = 5, f_2 = 0.25$, $g = 9.81 m/s^2$, $v_{min} = 0$, $v_{max} = 0$, $v_d = 24 m/s$, $c = 0.4$, $\bar{D} = 10$. The leader is designed to slow down during the first 10 sec:

$$a_L = \begin{cases} -0.6(1 - \tanh\left(\frac{1}{2(10-t)}\right) & \text{if} \quad t \leq 10\text{sec}, \\ 0 & \text{if} \quad t > 10\text{sec}. \end{cases} \tag{4.34}$$

The objective of the CBF-QP to ACC is designed in the same way as [15]. We compare CBF-QP with RT-CBF-QP in Fig. 4.3. The initial value of the parameters are $\nu_i^1 = 0.5, \nu_1^2 = 0.7, \nu_2^1 = 0.5, \nu_2^2 = 0.5$. The exponential rate of convergence for CLF is $k = 10$. The initial state is $v = 20 m/s, v_L = 10 m/s, D = 100 m$. The simulation is run for 50 seconds. The standard CBF-QP (i.e., with fixed parameters) becomes infeasible before the simulation finishes as shown in Fig. 4.3. RT-CBF-QP on the other hand maintains feasibility and safety at all times. The variations of the barrier function h_1, the parameter ν_1^2, and the control input u over time are shown in Fig. 4.4. Note that ν_1^2 starts increasing as the control-input bound is approached. We design the desired parameter value $\nu_{i,d}^1$ in (4.24) to be a P controller with gain 1 and saturated at 1 to steer ν_i^2 to 0.7. As seen in Fig. 4.4(a), ν_i^2 returns to its nominal value of 0.7 eventually. Other parameters do not change in this example, and their variation is thus not shown. This example illustrates that, for a chosen barrier function h, the feasibility of CBF-QP controllers is highly dependent on the (fixed) parameters, and that online adaptation can help circumvent this issue.

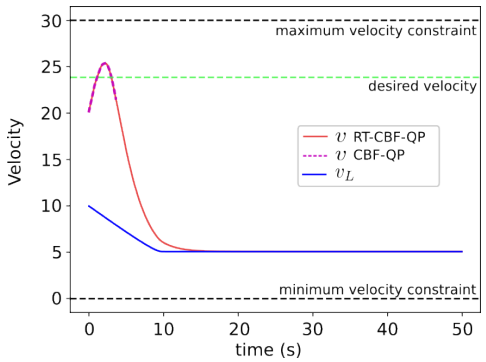


Figure 4.3: Velocity variation for the ACC problem. The dotted pink line ends when CBF-QP becomes infeasible. The leader decelerates for the first 10 seconds and then moves at constant velocity. The ego-agent approaches the boundary of the safe set (see Fig. 4.4(b)) and hence does not have enough bandwidth to steer back towards its desired velocity of $24 m/s$.

4.3.3.3 Baseline Comparisons

In this section, we compare our method to an existing and similar myopic method for adapting parameter ν. Consider the multi-agent scenario shown in Fig. 4.5 where the ego agent

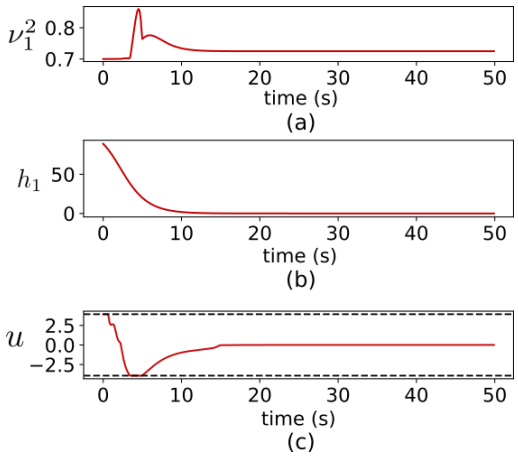


Figure 4.4: The variation with time of (a) parameter ν_i^2: ν_i^2 increases as the control input approaches its bounds, and eventually reduces towards the nominal value of 0.7 due to the effect of the $\nu_{1,d}^1$ in (4.24), (b) barrier h_1: The ego agent approaches and maintains the minimum distance w.r.t. the other agent., and (c) normalized control input u: Normalization is done as v/M^2

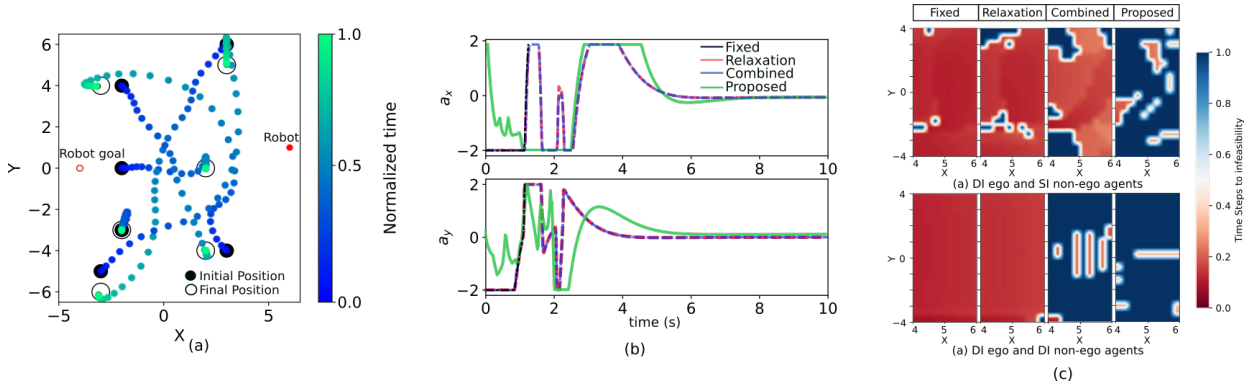


Figure 4.5: (a) Environment setup for baseline comparison. Trajectories of non-ego agents are shown from their start position to the final position. The robot (red) navigates to the goal location while avoiding other agents. (b) Control input variation for DI ego-agent under different methods. (c) Time to infeasibility (normalized) as a function of initial X, Y position of the robot in meters. Fixed corresponds to fixed-parameter case.

(red) navigates to its goal location $p_{x,d} \in \mathbb{R}^2$ while avoiding the black obstacles. To provide a fair comparison among all the methods, all the non-ego agents move agnostic to the existence of the ego agent, and the dynamics, as well as control inputs of the non-ego agents, are made available to the ego agent. Each non-ego agent designs its control input and uses a CBF-QP to perform collision avoidance with other non-ego agents. Note that although non-ego agents also use (vanilla) CBF-QP, this choice of controller is arbitrary and has no implications for the controller that the ego agent uses. This example serves to compare methods on a system with multiple constraints, without delving into a certified controller design for a multi-agent system which is outside the scope of this dissertation.

The ego-agent is modeled as a double integrator and uses one of the following controllers: 1) *proposed method*, 2) *relaxation method* where $\nu_i^k, k \in \{1, ..r_i - 1\}$ are kept fixed and $\nu_i^{r_i}$ is relaxed, akin to [266], by making it an optimization variable and adding $Q \sum_{i=1}^{N} (\nu_i^{r_i} - \nu_{i,nom}^{r_i})^2$ to the objective function (2.30a) with $Q > 0$ being a user-chosen constant weighting factor, and finally 3) *combined method* where $\nu_i^{r_i}$ is adapted by relaxation akin to [266] and $\nu_i^k, k \in \{1, ..r_i - 1\}$ are adapted with proposed method. *Combined method* is implemented to improve upon the *relaxation method* while still keeping a quadratic program structure. A direct extension of [266] to HOCBF by treating $\nu_i^k, k < r_i$ as an optimization variable would make it a nonlinear optimization problem that is implementable but more expensive than a QP.

The second order barrier function used by ego-agent is $h_i = ||p_x - p_{x,i}||^2 - d_{min}^2, i \in \{1, .., N\}$ where $p_x \in \mathbb{R}^2$ is the position of ego agent, N is the number of non-ego agents, and p_{x_i} is the position of the i^{th} non-ego agent, and d_{min} is the collision avoidance radius. The nominal values of $\nu_{i,nom}^1$ and $\nu_{i,nom}^2$ are taken as 3, 6.0. The nominal controller u_r of the ego-agent is designed as $u_r = -k_v(\dot{p}_x - v_d), v_d = -k_x(p_x - p_{x,d})$ with $k_x = 1.5, k_v = 3.0$. For the proposed method, the initial value of ν_i^2 is taken as 0 to start conservatively and then become aggressive with adaptation towards $\nu_{i,nom}^k$ using the proposed algorithm. This, in general, helps maintain feasibility better across different scenarios.

We simulate different dynamics of non-ego agents; single integrator (SI) and double integrator (DI). Since myopic algorithms are sensitive to the initial state of the robot, we further show the simulation time at which the ego-agent controller becomes infeasible as a function of the initial position of the robot. The initial velocity is set to zero across all runs. The results are shown in Fig. 4.5(c), both for SI non-ego agents (above) and DI non-ego agents (below). When the non-ego agents are SI, their acceleration is taken as zero by the ego-agent to evaluate required derivatives. Note that all methods being myopic cannot guarantee feasibility in a multi-constrained system. We simulate for $\lambda = 1.5$ in (4.24) and $\Delta_i^1, \Delta_i^2 = 1\Delta t$ for SI non-ego agents and $\Delta_i^1 = 10.0\Delta t$, $\Delta_i^2 = 1.0\Delta t$ or DI non-ego agents where $\Delta t = 0.02$ is the simulation time step. Finally, note that although in theory *Relaxation* method (and ours) is guaranteed to be feasible when $\psi_i^{r_i - 1} > 0$ in (2.30b) since it completely relaxes $\nu_i^{r_i}$, in practice this requires $\nu_i^{r_i} \to \infty$ as $\psi_i^{r_i - 1} \to 0$ which, due to limited floating point precision of the solver, results in infeasible or unbounded error by the solver both of which are interpreted as being infeasible in our results.

4.3.4 Trust-based adaptation for Decentralized Multi-agent Systems

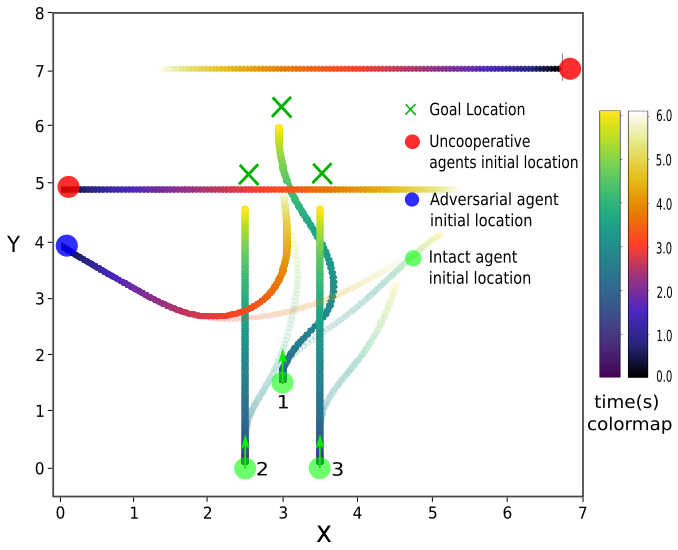
As part of an illustrative case study, we show application of RT-CBFs for decentralized uncooperative multi-agent systems. In presence of non-cooperative or adversarial agents, several works assume worst-case scenario and design a robust controller to navigate the agent. However, this leads to overly conservative behavior. In this case study, we adapt parameters of class-$\mathcal{K}$ function to reduce this conservatism. We design the parameter dynamics based on a *trust factor* that is computed based on the instantaneous ease of satisfaction of the CBF constraints. The contribution of other agents to the satisfaction of CBF constraints can be computed from state observations. Detailing the exact design of trust-factor would result in a significant detour from the progression made in this chapter and therefore the algorithmic details are mentioned in Appendix A.2. In the ensuing, we show some results showcasing the ability of our algorithm to reason about the motions of other agents.

Videos and code available on the thesis website https://github.com/hardikparwana/PhD-Thesis. We consider a decentralized multi-robot system. The behavior of the neighbors of each agent can be either cooperative or non-cooperative as defined in the earlier section. For the simulation case study, we consider the non-cooperative neighbors of each agent (called ego agent) to be of two behaviors, namely: 1) adversarial; if a neighbor is adversarial, then it means that it is simulated to move head-on towards the ego agent; and 2) uncooperative; if a neighbor is uncooperative then it means that it is simulated to move along an arbitrary direction (resembling a dynamic obstacle). We provide results for two scenarios as shown in Fig. 4.6. The initial locations of each ego agent are marked with green circles, uncooperative agents with blue circles, and adversarial agents with red circles. Note that since an agent can be adversarial with respect to some agent but uncooperative to some other, we clarify later on in each case study which agent we are referring to.

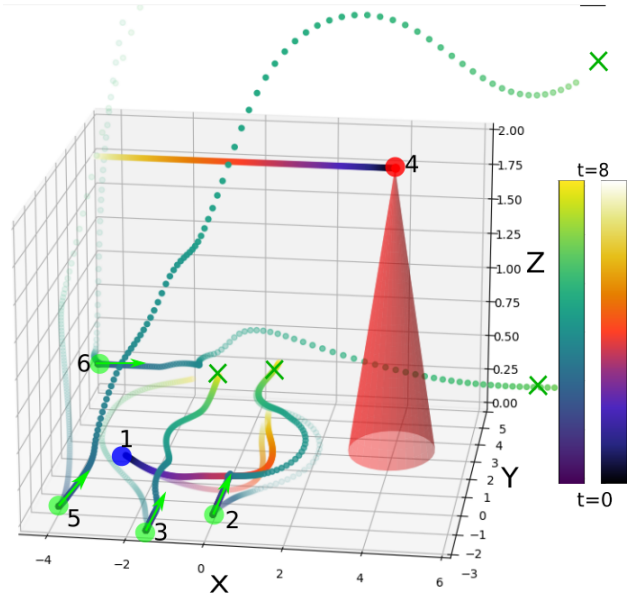
For scenario 1, there are three ego agents $N = 1, 2, 3$ that are modeled as unicycles; note however that none of them assumes that the other ego agents are cooperative. The agents $1, 2, 3$ use the first-order barrier functions from [251] in order to enforce safety. The agent depicted in blue (Agent 5) is adversarial with respect to Agent 1, and is modeled as a single integrator, and the agent in red (Agent 4) is uncooperative (w.r.t. all agents) and moves to the left at constant velocity. None of the ego agents know the identities of any other agent in the system, including other ego agents. The initial values of $\nu_{\alpha_{ij}}$ are all chosen to be 0.8 and the resulting trajectories and variations in barrier function value and class-$\mathcal{K}$ function parameters are shown in Figs. 4.6a and 4.7.

We also compare our results with fixed parameter case. No solution exists to QP(2.30) after some time when all ν_{ij} are initialized to 0.8, where ν_{ij} is the class-$\mathcal{K}$ parameter that robot i uses to enforce collision avoidance with robot j. Hence, assuming identities are known, we choose θ_{ij} 2.0 when j is an ego agent and 0.8 otherwise. The fixed parameter CBF controller fails to lead the robots to their goal locations because of the conservative response wherein it steers the ego agents too far away from other agents to ensure safety. In the RT-CBF implementation on other hand, the value of parameters increases (making controller aggressive) when the trust is positive, that is when it is easy to satisfy the CBF constraint, as shown in Fig.4.7. The parameters decrease (making controller conservative) when trust is negative which happens when an agent is moving too fast towards the ego agent or is very close. Specifically, when agent 1 deviates to the right ($t = 0$ to 1.5s), it increases the trust for agents 2,3 and one of the noncooperative agents as they do not seem to chase after agent 1 and therefore leads to positive trust. Whereas since the adversary and another uncooperative agent move towards agent 1, their trust is 0 or a small negative and does not affect the parameters much. Next, when agent 1 turns back to the left, it perceives a negative trust for agents 2,3 as they seem to be moving towards agent 1. Eventually, since the adversary slows down as it approaches agent 1 (the adversary's controller is based on Lyapunov function) and because the uncooperative agents steer far away, agent 1 is more capable of evading the adversary and therefore positive trust to it. This leads to the transient increase in class-$\mathcal{K}$ parameter (red) for collision avoidance with an adversary in Fig. 4.7a. In our simulation, the beliefs that each ego agent employs are straight-line motions parallel to Y axis for other ego agents and parallel to X axis for other agents.

In scenario 2, shown in Fig. 4.6b, the ego agents labeled 2 and 3 are modeled as unicycles and use the first-order barrier function from [251], and the agents labeled as 5,6 are modeled as double integrators and use the second-order barrier function $h_{ij} = s^2 - smin^2$ where s is the distance between agents i, j and s_{min} is minimum allowed distance. The adversarial agent chases the unicycle labeled as 2. The uncooperative agent moves to the left at constant velocity but has a visibility cone that must be avoided by ego-agents. We again choose linear class-$\mathcal{K}$ functions with $\theta_{\alpha_{ij}}$ initialized to 0.8 for first order barrier function and $\theta_{\alpha^0_{ij}}$ to 0.4 and $\theta_{\alpha^1_{ij}}$ to 0.8 for second order barrier functions. The response of the fixed parameter case is again to be very conservative compared to the proposed method.



(a) Scenario 1

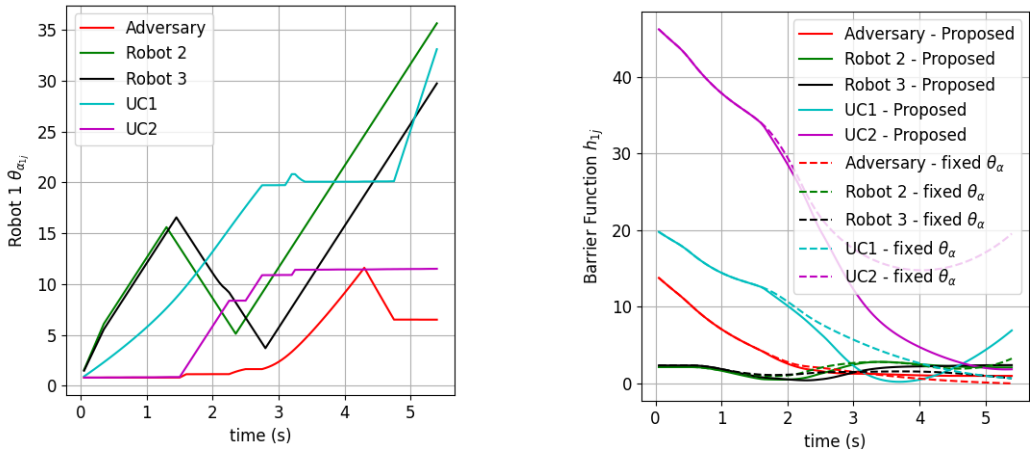


(b) Scenario 2

Figure 4.6: The timestamp is given by the colormap. Paths with bold colors result from the proposed method. Paths with increased transparency result from the application of CBFs with fixed class-$\mathcal{K}$ function parameter.

4.4 Discrete time Predictor-Corrector Approach to Parameter Optimization

In this section, we propose another method to adapt parameter ν for a discrete-time dynamical system by employing a model-predictive framework. In a nutshell, we predict the future



(a) Scenario 1 Parameter Variation

(b) Scenario 1 Barriers

Figure 4.7: Scenario 1: Variation of class-$\mathcal{K}$ function parameters (a) and barrier functions (b) of Robot 1 with time. UC1, UC2 are the uncooperative agents starting from left and right respectively in Fig. 4.6a. Trust-based relaxation allows agents to go closer to the safe set boundary ($h_{ij} = 0$) compared to fixed α case, and hence leads to a less conservative response while still guaranteeing safety.

states using system dynamics and QP controller (4.2) for a fixed parameter, then compute the sensitivity (gradient) of a user-specified performance metric w.r.t the parameter, and then perform online constrained gradient descent inspored by sequential quadratic programming. Therefore, we pose the control synthesis problem as a differential policy whose parameters are optimized for performance over a time horizon at high level, thus resulting in a bi-level optimization routine (lower level optimization being the QP controller itself). In the absence of knowledge of the set of feasible parameters, we develop a Recursive Feasibility Guided Gradient Descent approach for updating the parameters of QP so that the new solution performs at least as well as the previous solution. By considering the dynamical system as a directed graph over time, this work presents a novel way of optimizing the performance of a QP controller over a time horizon for multiple CBFs by (1) using the gradient of its solution with respect to its parameters by employing sensitivity analysis, and (2) backpropagating these as well as system dynamics gradients to update parameters while maintaining feasibility of QPs.

4.4.0.1 Sequential Quadratic Programming

We briefly describe Sequential Quadratic Programming (SQP) [38, 225] that we use extensively in Section 4.4 to derive real-time parameter adaptation scheme. Consider an optimization problem in $z \in \mathbb{R}^n$ with cost function $J : \mathbb{R}^n \to \mathbb{R}$, and constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$

$$\min_{z} \quad J(z), \tag{4.35a}$$

$$\text{s.t.} \quad c(z) \leq 0, \tag{4.35b}$$

In SQP, given an estimate $\bar{z}$ of the solution of the optimization problem (4.35), gradients or sensitivities of the cost and constraint functions at $\bar{z}$ are used to compute infinitesimal update directions d so that new estimate becomes $\bar{z}^+ = \bar{z} + d$. The update direction d is computed based on two factors

(1) Improving-cost direction $-\nabla J|_{z=\bar{z}}$: this is obtained by computing the gradient of the cost function.

(2) Feasible update directions: these are the directions for which constraints that are satisfied with the current estimate $\bar{z}$ continue to be satisfied with the new estimate $\bar{z}$. The set of feasible directions $\mathcal{F}$ up to first-order approximation is given as:

$$\mathcal{F}(\bar{z}) = \{d \mid c(\bar{z}) + \nabla\langle c|_{z=\bar{z}}, d\rangle \leq 0\}. \tag{4.36}$$

The set $\mathcal{F}$ need not be computed explicitly. The update direction d is computed by projecting the improving-cost direction into the set of feasible directions by solving the following linear program

$$\min_{d} \quad \langle \nabla J_{z=\bar{z}}, d\rangle \tag{4.37a}$$

$$\text{s.t} \quad c(\bar{z}) + \langle \nabla c_{z=\bar{z}}, d\rangle \leq 0. \tag{4.37b}$$

Note that the notion of feasible directions is needed because moving only in the direction of improving cost does not guarantee that if $c(z) \leq 0$ then $c(\bar{z}^+) \leq 0$.

4.4.1 System Description

In this section, we abuse the notations used for continuous-time systems in Chapter 2 to define the discrete-time system. This choice is made to not confuse readers with many different notations.

Consider a control affine nonlinear dynamical system with state $x \in \mathcal{X} \subset \mathbb{R}^n$, control input $u \in \mathcal{U} \subset \mathbb{R}^m$ given in discrete time as

$$x_{t+1} = f(x_t) + g(x_t)u_t, \tag{4.38}$$

where $f(x) : \mathcal{X} \to \mathbb{R}^n, g(x) : \mathcal{X} \to \mathbb{R}^{n\times m}$ are Lipschitz continuous and t is the time index. The safety of the system is specified in terms of, possibly time varying but intersecting, safety sets $\mathcal{S}_i(t), i \in 1, 2, .., N$ that encode the allowable states of the system and are defined as 0-superlevel set of a smooth function $h : \mathcal{X} \times \mathbb{R}^+ \to \mathbb{R}$ as follows

$$\mathcal{S}_i(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) \geq 0\}, \tag{4.39}$$

$$\partial\mathcal{S}_i(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) = 0\}, \tag{4.40}$$

$$\text{Int}(\mathcal{S}_i)(t) \triangleq \{x \in \mathcal{X} : h_i(t, x) > 0\}. \tag{4.41}$$

The set $\mathcal{S}_i(t)$ is forward invariant if and only if $\dot{h}_i(t, x) \geq 0 \; \forall x \in \partial\mathcal{S}_i(t)$ [36] in continuous-time systems. With slight abuse of notation, we will use $\mathcal{S} = \cap S_i$. If h_i is a zeroing-barrier function for a class-$\mathcal{K}_\infty$ function ν_i, then the following condition in discrete-time on h_i, called CBF condition, is sufficient for the invariance of the set $\mathcal{S}_i$:

$$h_i(t + 1, x_{t+1}) - h_i(t, x_t) \geq \nu_i(h_i(t, x_t)). \tag{4.42}$$

For simplicity, we will be using $\nu_i(x) = \alpha_i x, \alpha_i \in \mathbb{R}^+$ in remainder of dissertation but any other parametric function can also be used.

Problem 2. Given the dynamical system (4.38) and initial state $x_0 \in \mathcal{S}(0)$, design a controller $u(x, t)$ such that $\forall t > 0$, the closed loop trajectory satisfies $x(t) \in \mathcal{S}(t)$ and $||x_t - x_t^d|| \to 0$ as $t \to \infty$.

In order to achieve this objective, the following CBF-CLF-QP, a discrete-time variant of the continuous HOCBF-QP (2.30), which we shall refer to as policy $\pi_{QP}(x_t; \alpha_0, \alpha_1, ...\alpha_N)$, will be used: Let the state and input at time step $\tau \in \mathbb{Z}^+$ be denoted as x_τ, u_τ and let $\tau + 1$ denote the next sampling time instant. The HOCBF-QP controller (2.30), augmented with control Lyapunov function, is given in discrete-time implementation as

$$\min_{u_t \in \mathcal{U}, \delta} \quad J(u_\tau) = (u_\tau - u_{d_\tau})^T Q(u_\tau - u_{d_\tau}) + M\delta^2, \tag{4.43a}$$

$$\text{s.t.} \quad V(\tau + 1, x_{\tau+1}) \leq (1 - \nu_0)V(\tau, x) + \delta, \tag{4.43b}$$

$$\psi_i^k(\tau + 1, x_{\tau+1}) \geq (1 - \nu_i^k)\psi_i^k(\tau, x_\tau), \tag{4.43c}$$

$$\forall i \in \{1, 2, .., N\}, k \in \{1, 2, .., r - 1\},$$

$$Au_\tau \leq b, \tag{4.43d}$$

Here $\nu_i, i \in 0, 1, ..., N$ are parameters of the policy, u_{d_t} is the desired or nominal control input, $P \in \mathbb{R}^{m\times m}, Q \in \mathbb{R}^+$ are the weight factors, $V(x)$ is a given Lyapunov function, and

$\delta \in \mathbb{R}$ is slack variable used to relax CLF condition. In the subsequent, we will refer to all parameters together in a vector $\theta = [\alpha_0, ...\alpha_N]^T$. Note that u_{d_t}, P, and Q may also be treated as parameters if desired so.

Remark 11. Refer to [9], [42] for a thorough discussion of discrete time CLF and CBF conditions and their variants. We use a simple extension of their work that ensures constraint satisfaction only at sampling times. Any other variation with stronger inter-sample guarantees may be used too.

Example 3. A follower modeled as a unicycle is supposed to follow a leader and maintain it within its Field-of-View(FoV). The Lyapunov function is defined on desired radial distance only, while three barrier functions h_1, h_2, h_3 encode minimum distance, maximum distance, and maximum angle w.r.t camera axis constraints. A control policy can be synthesized as

$$\begin{aligned} \min \quad & u^T u + Q\delta^2 \\ \text{s.t.} \quad & V(t+1, x_{t+1}) \leq -(1-k)V(t, x_t) + \delta \\ & h_1(t+1, x_{t+1}) \geq (1 - nu_1)h_1(t, x_t) \\ & h_2(t+1, x_{t+1}) \geq (1 - nu_2)h_2(t, x_t) \\ & h_3(t+1, x_{t+1}) \geq (1 - nu_3)h_3(t, x_t). \end{aligned} \tag{4.44}$$

Here, $k, \alpha_1, \alpha_2, \alpha_3$ are parameters of the problem.

Most implementations choose the parameters θ based on manual tuning and assume that h_i are valid CBF that can be satisfied simultaneously at all states. However, this assumption is not true in general, and system trajectories may violate the above constraints even in presence of unbounded control inputs. Even when constraints do remain feasible, they may not allow desirable performance and there exists no automatic tuning procedure for θ. The following example shows how even in simple scenarios, having multiple constraints can lead to infeasibility in future.

Example 4. Consider the motion of an autonomous car modeled as one-dimensional integrator system $\dot{x} = u$ with state x and velocity input u. Now consider a common road scenario in which the car is caught between two vehicles. The vehicle in front is moving slower than the vehicle at back and therefore the safe set, described as no collision zone between the vehicles, is shrinking with time and vanishes in future. The objective is to stay safe for maximum possible time. Let the shrinking set be given by the constraint $x \geq t$, and $x \leq 1 + ct$ where t is time and $c < 1$ is a constant. We can encode these two constraints with two barrier functions $h_1 = x - t$, $h_2 = 1 + ct - x$ and solve the following optimization

problem

$$\begin{aligned} \max \quad & u \\ \text{s.t.} \quad & h_1(t+1, x_{t+1}) \geq (1-a)h_1(t, x_t) \\ & h_2(t+1, x_{t+1}) \geq (1-b)h_2(t, x_t). \end{aligned} \tag{4.45}$$

Here, $a, b \in \mathbb{R}$ are parameters. Linear Programs(LP) satisfy the same assumptions introduced for QPs in Section 4.4.2 and therefore are still amenable to our approach. Note that eventually the safe set completely vanishes here but the response of the system before that critical time is still relevant to real scenarios. The closed loop system for different values of parameter is simulated. Fig.4.8 shows the resulting trajectories and Fig.4.9 shows how time to infeasibility changes with parameter choice for two different set shrinkage rates c. Both figures are asymmetric w.r.t a, b which is a consequence of difference in rate of change of both constraints. Figs.4.8,4.9 also show why it is important to choose parameter proactively as a wrong parameter choice may lead to infeasibility and possible crash of the system.

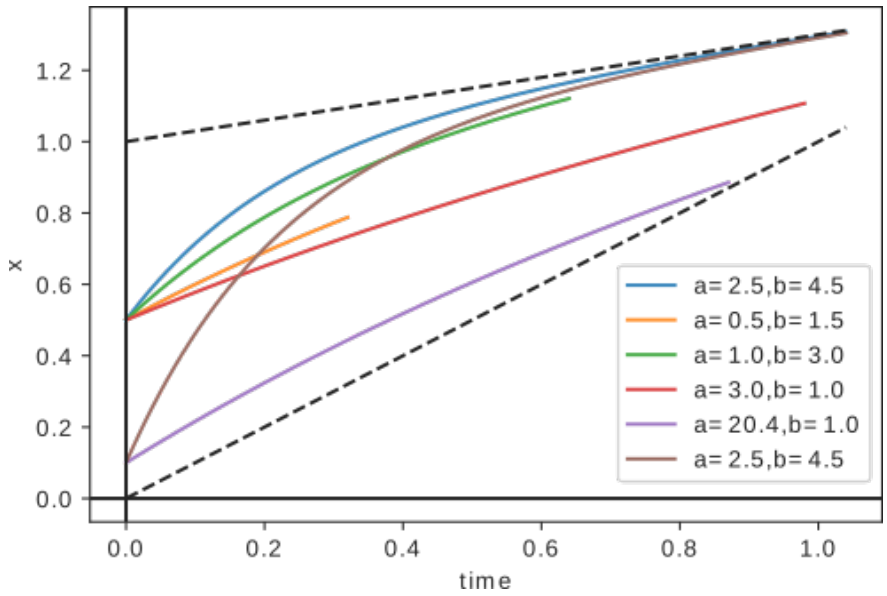


Figure 4.8: Plot of trajectories for different values of parameter. Trajectories end at point of infeasibility. The values of a and b shown are normalized by sampling time Δt.

4.4.2 Differentiable Convex Programs

Finally for sake of completion, we mention this important result that allows us to differentiate solution of QPs with respect to its parameters. This validates our use of gradients in subsequent sections. We first make following assumptions concerning KKT conditions.

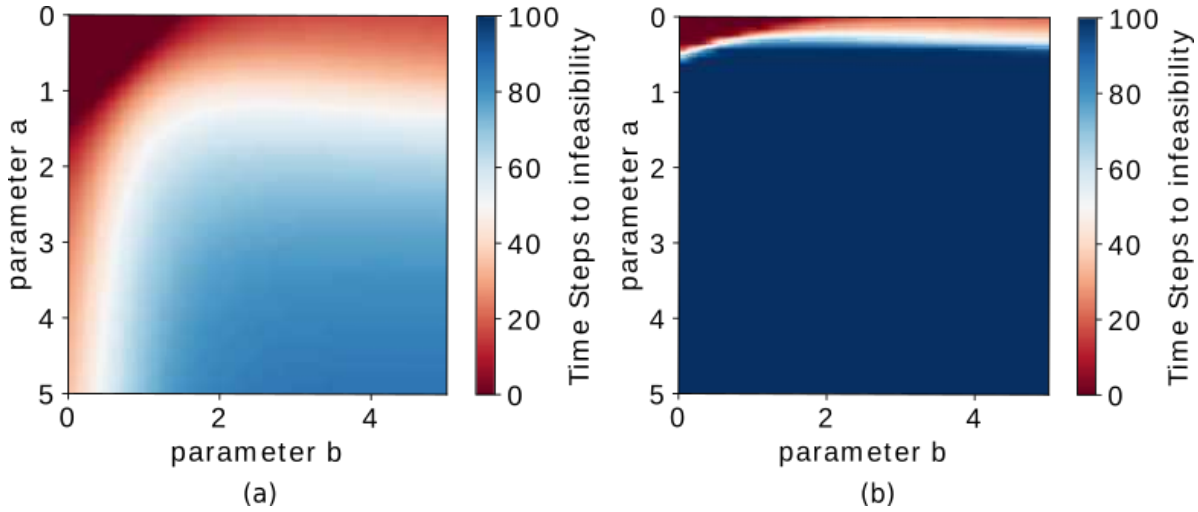


Figure 4.9: Time to Infeasibility: X and Y axis represent parameter values(normalized by sampling time Δt) for which trajectory is simulated. The color gradation represents the time after which problem becomes infeasible. A value of 100 implies it remains feasible for maximum possible time and value of 0 implies it is infeasible from the start. (a)c=0.3, (b)c=0.7 in Example 4.

Assumption 1. (Strong Duality) Slater's condition holds for (4.43)

Assumption 2. Let $e(t, u_t, x_t, \theta) \in \mathbb{R}^{N+1} \leq 0$ represent the vector of all inequality constraints in (4.43). The functions e_i and the objective $J(\pi)$ are twice differentiable in u_t, and $e(u_t, x_t, \theta), \nabla_u e(u_t, x_t, \theta)$ are continuously differentiable in (x_t, θ). Note that for QP, (x_t, θ) are parameters and u_t is a variable. This assumption would usually be satisfied for most systems of interest that have differential dynamics and a smooth objective function.

Assumption 3. Strict complementarity holds for (4.43).

Remark 12. Note that in many control problems, the solution of QP may lie exactly on the boundary of constraint inequalities and Assumptions 1,3 may not hold and gradient may be undefined. However even in these cases, relaxed KKT conditions [30] can be used to get gradient in the limiting case [99]. Alternatively, a more thorough analysis based on directional differential along a parameter trajectory may be possible but is out of scope of this dissertation. A simple way to ensure differentiability of QP w.r.t its parameters is to ensure that $u_t = u_{d_t}$ does not make the constraints active.

Lemma 2. ([8], [24]Theorem3.1) If Assumptions 1,2, and 3 hold, then the QP solution is a locally single valued function around its solution u_t and thus is continuously differentiable in the neighborhood of (x_t, θ).(Refer to [24] for a more insightful condition).

4.4.3 Problem Statement

A typical objective to be achieved over a time horizon is to maximize the summation of stage-wise cost, which we call the *Reward* r, subject to constraints (4.42b,4.42c).This second optimization routine at the high-level is given as follows:

Now suppose the reward of choosing action u_t at state x_t is given by the function $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. The reward r can, for example, encode either of stability and safety objectives or try to minimize control input. A finite-horizon optimal control problem to find (fixed) parameters θ_α that maximize the reward accumulated for T time steps starting at time τ can be formulated as follows:

$$\begin{aligned}
\max_{\theta_\alpha} \quad & R(x_\tau, \theta) = \sum_{s=\tau}^{\tau+T} r(x_s, u_s) + \Phi(x_{\tau+T+1}), && \text{(4.46a)}\\
\text{s.t.} \quad & u_s = \pi_{QP}(x_s; \theta_\alpha), && \text{(4.46b)}\\
& x_{s+1} = f(x_s) + g(x_s)u_s, && \text{(4.46c)}\\
& \forall s \in \{\tau, \tau+1, .., \tau+T\},
\end{aligned}$$

where $\Phi : \mathcal{X} \to \mathbb{R}$ is the terminal cost, which ideally is representative of the residual infinite-horizon cost. The QP controller (4.43) is said to be implemented at *low-level*, and the optimization (4.46) at *high-level*. The optimization (4.46) is a nonlinear program whose global optimum is difficult to compute in real time. Therefore we only focus on methods that help improve the estimates of (4.46) but not necessarily lead to convergence to the global optimum.

Since we are looking for a real-time receding horizon approach, we resort to gradient descent, specifically SQP, based solutions of optimization problems and propose to use the sensitivity w.r.t θ_α of cumulative reward (4.46a) that is achieved for a fixed θ_α and given initial state x_τ. That is, at time τ we predict the future states by propagating the system (4.38) forward in time with control at each time step chosen in (4.46b) as $u_s = \pi_{QP}(x_s; \theta_{\alpha_\tau})$ and obtain the sensitivity (gradient) of predicted reward w.r.t θ_α. This sensitivity is then used to design $\dot{\theta}_\alpha$ at time τ and the procedure is repeated in receding horizon fashion at every time $\tau' > \tau$.

When predicting future states up to a horizon of T for a given parameter vector θ_α, the controller (4.43) may become infeasible at any time $s \leq \tau + T$ or it may never become infeasible. Therefore we propose different update laws depending on whether the current choice of the parameter $\theta_{\alpha\tau}$ at time τ leads to a feasible or infeasible control input trajectory at future times $s \in [\tau, \tau + T]$. For this purpose, we first introduce the following definitions.

Definition 10. Consider the system (4.38) evolving under the policy $\pi_{QP}(x;\theta_\alpha)$ given by (4.43) with fixed parameter θ_α.

- The parameter θ_α is called feasible if starting at state x_τ at time τ, the barrier functions $h_i(s, x_s)$ are compatible for all time $s > \tau$, that is, the controller π_{QP} admits a solution for all $s > \tau$.
- A parameter θ is called T-feasible for state x_τ and time τ, if the barrier functions $h_i(s, x_s)$ are compatible for all time $s < \tau + T$, but incompatible at time $\tau + T$.

In practice, it is sufficient to call a parameter feasible if (4.43) remains feasible for the period of interest T, which could be the time horizon we expect our autonomous agent to take to finish the task or get away from unsafe regions. Depending on whether we have feasibility or T-feasibility, we now specify two objectives to be solved in this section:

Objective 1. Given a feasible θ, design an update rule $\theta^+ = F(x, \theta)$, where $F : \mathbb{R}^n \times \mathbb{R}^{n_\alpha}$ is a lipschitz continuous function, such that θ^+ is feasible, and $R(x_\tau, \theta^+) \geq R(x_\tau, \theta)$.

Objective 2. Given a T-feasible θ, design an update rule $\theta^+ = F(\theta)$ such that θ^+ is $(T+1)$-feasible.

4.4.3.1 Methodology

We present two update rules inspired by Sequential Quadratic Programming (SQP) for solving Objectives 1 and 2.

Case 1: Feasible θ: First, we design an update rule for improving the performance of a feasible parameter. A gradient-based update rule computes infinitesimal changes in the current estimate of parameter θ_{α_τ}. We design the update rules based on SQP as introduced in section 4.4.0.1. In the context of our problem, the improving-cost direction is simply obtained as the gradient of the reward function $R(x_\tau, \theta)$ in (4.46a) w.r.t θ_{α_τ}. The set of feasible directions is computed such that if the parameter θ_{α_τ} is feasible (or T-feasible), then $\theta^+_{\alpha_\tau}$ is also feasible (respectively T-feasible). Let $e(t, u_t, x_t, \theta_{\alpha_\tau}) \in \mathbb{R}^{N+1+q} \geq 0$ represent all inequality constraints at time t in (4.46b) which are respectively given by (4.43b), (4.43c), and (4.43d). The update direction d, following the SQP rule in Section 4.4.0.1, is thus obtained as a result of the following linear program.

$$d = \arg\max_{d_\theta} \quad \langle \nabla_\theta R(x_\tau, \theta_{\alpha_\tau}), d_\theta \rangle \tag{4.47a}$$

$$\text{s.t.} \quad e(s, u_s, x_s; \theta_{\alpha_\tau}) + \nabla e(s, u_s, x_s; \theta_{\alpha_\tau}) d_\theta \geq 0,$$
$$\forall s \in \{\tau, \tau+1, .., \tau+T\} \tag{4.47b}$$

Remark 13. The rule (4.47) uses the first-order approximation of the change in the value of function $e(t, u_t, x_t, \theta_\alpha)$ when the parameter θ_α is perturbed. This is equivalent to linearizing the solution of (4.38) about the current trajectory $(x_s, u_s), s \in \{\tau, \tau+1, .., \tau+T\}$. This rule employing SQP is also inspired by Differential Dynamic Programming literature [217] that iteratively improves estimates of the optimal trajectory by using the sensitivity of future trajectory w.r.t a given sequence of control input. However, unlike DPP, we use a parameterized controller rather than considering a sequence of control inputs, and we use sensitivities for the purpose of automated tuning of controller parameters.

Case 2: T-feasible θ: Second, we design an update rule to perturb a T-feasible parameter with the goal of making it a $(T+1)$-feasible parameter. Given that the QP (4.43) is infeasible at $\tau+T$, we add slack variables to all the hard constraints so that the imposed constraint is

$$e(\tau+T, u_{\tau+T+1}, x_{\tau+T+1}; \theta) + \bar{\delta} \geq 0. \tag{4.48}$$

The slack $\bar{\delta} > 0$ is thus the infeasibility margin of the hard constraints and should be minimized to zero. The high-level objective to find the update direction d thus becomes

$$d = \min_{d_\theta} \quad \langle \nabla_\theta(\bar{\delta}^T \bar{\delta}), d_\theta \rangle \tag{4.49a}$$

$$e(s, u_s, x_s; \theta_{\alpha_\tau}) + \nabla e(s, u_s, x_s; \theta_{\alpha_\tau}) d_\theta \geq 0,$$
$$\forall s \in \{\tau, \tau+1, .., \tau+T-1\}. \tag{4.49b}$$

Note that the only independent variables in (4.47), (4.49) are the initial state x_τ and the parameter θ, and therefore the gradients have to be computed w.r.t. to these independent variables.

The next two subsections discuss how we can compute gradients by backpropagating. These gradients can then be used in (4.47), (4.49) to find d. In the next section, we describe how automatic differentiation can be used to obtain the required gradients.

4.4.4 Dynamical System as a Computational Graph

Firstly, we design an update rule for improving performance of a feasible parameter. This is done by computing gradient of the objective $\nabla_\theta J(x, \theta)$, projecting it to the set of feasible directions of θ, and then updating in the resulting direction with a learning rate β. Here, the set of feasible directions is computed such that a change in this direction will maintain feasibility of the QPs and guarantee existence of trajectory at all times (or upto a large

horizon in practice). This is done in manner similar to FSQP where parameters are updated so that each iterate of gradient descent, not just the last one, is a feasible solution.

Secondly, we design a rule to update a T-feasible parameter to (T+1)-feasible parameter. This is done by first identifying the constraint at T+1, that needs to be relaxed the least of all other constraints so that the QP at T+1 becomes feasible. A direction that reduces the infeasibility margin of this constraint is then computed with respect to θ. This direction is then projected to set of feasible directions of QPs from t=1 to t=T and the resulting direction is used for update with learning rate β. The next two subsections discuss how we can compute gradients by backpropagating and then how we can use them in FSQP framework to update parameters.

4.4.5 Gradient through Backpropagation

The dynamical system (4.38) can be visualized as a computational graph (see Fig.4.10) where the nodes either represent independent variables or functions, and incoming edges represent the arguments to the function and the flow of information. The gradient of each function node can be obtained either from the system dynamics or through the sensitivity analysis of the QP. Since the dynamics is deterministic and known, we can roll-out a trajectory to the future and compute the gradient of the objective function with respect to θ and initial state x_τ at time τ. As an example, for $T = 2$,

$$\frac{\partial R(x_\tau, \theta)}{\partial \theta} = \frac{\partial r(x_\tau, u_\tau)}{\partial \theta} + \frac{\partial r(x_{\tau+1}, u_{\tau+1})}{\partial \theta}. \tag{4.50}$$

Here, we have

$$\frac{\partial r(x_{\tau+1}, u_{\tau+1})}{\partial \theta} = \underbrace{\frac{\partial r_{\tau+1}}{\partial x_{\tau+1}}}_{known} \frac{\partial x_{\tau+1}}{\partial \theta} + \underbrace{\frac{\partial r_{\tau+1}}{\partial u_{\tau+1}}}_{known} \frac{\partial u_{\tau+1}}{\partial \theta}, \tag{4.51}$$

with $r_{\tau+1} = r(x_{\tau+1}, u_{\tau+1})$. Since $x_{\tau+1} = f(x_\tau) + g(x_\tau)u_\tau$ is a function of x_τ and u_τ

$$\frac{\partial x_{\tau+1}}{\partial \theta} = \underbrace{\frac{\partial x_{\tau+1}}{\partial x_\tau}}_{\text{from dynamics}} \underbrace{\frac{\partial x_\tau}{\partial \theta}}_{x_\tau \text{ constant}} + \underbrace{\frac{\partial x_{\tau+1}}{\partial u_\tau}}_{\text{from dynamics}} \frac{\partial u_\tau}{\partial \theta}. \tag{4.52}$$

$\frac{\partial u_\tau}{\partial \theta}, \frac{\partial u_{\tau+1}}{\partial \theta}$ are gradients of QP's solution w.r.t. the parameters which can be obtained by performing sensitivity analysis of QPs which is readily implemented in libraries such as cvxpy [78]. The chain-rule based recursive analysis that we employ in (4.50)-(4.52) to compute gradients of stage-wise reward w.r.t the parameter θ is amenable to backpropagation-based

derivative computation and is also readily implementable in libraries like pytorch and tensorflow, that are capable of constructing computational graphs. The gradient of other quantities of interest, such as barrier function, can be obtained similarly.

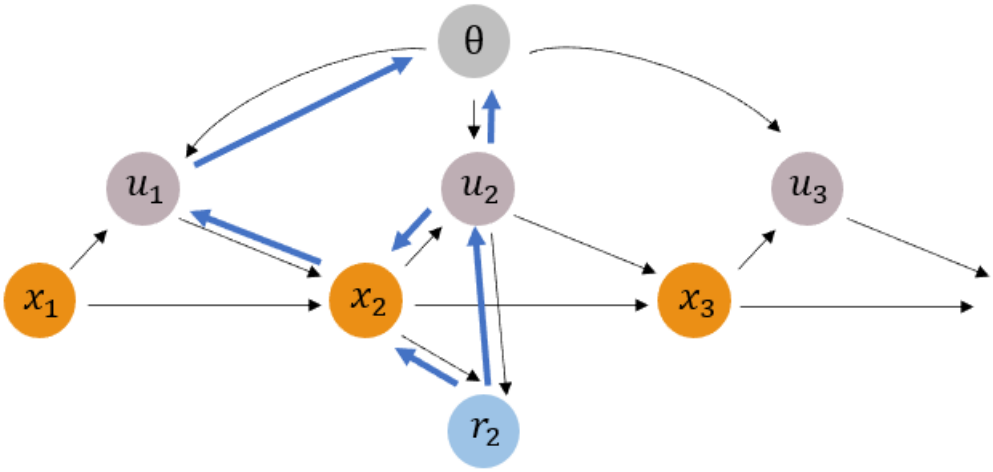


Figure 4.10: Nodes represent functions, specified here by dynamics or QP controller, and edges represent arguments to the function. Thick blue arrows show the flow of information when computing gradient of second reward $r_2 = R(x_2, u_2)$ with respect to θ through backpropagation.

4.4.6 SQP based Parameter Update - Back-propagation with Differential Optimization

We introduce a constraint aware update law - Recursive Feasibility Guided Gradient Descent(RFGGD) to design the class-$\mathcal{K}$ function parameter dynamics.

Using the gradients obtained through the backpropagation procedure we can solve the optimization problems (4.47) and (4.49) to obtain the update direction d. We now use the following update

$$\theta_{t+1} = \theta_t + \beta d, \tag{4.53}$$

where $\beta \in (0, 1]$ is the learning rate of gradient descent. Since d was computed based on linearization about the rolled-out trajectory corresponding to the parameter θ, we use a small learning rate to ensure approximation errors due to linearization are suppressed. Note that choosing a proper learning rate to suppress linearization errors has been an active research area due to the popularity of gradient descent-based updates for training highly nonlinear neural network models [27, 269] and is outside the scope of this dissertation. In

practice, some manual tuning is required for our approach when applied to an agent with new nonlinear dynamics.

Remark 14. While SQP provides constraint-maintenance guarantees only up to first-order approximation due to linearization, one can employ Feasible SQP [225] algorithms that can guarantee constraint satisfaction for higher-order approximations. However, since these methods also require computation of higher-order gradients such as Hessians, which can also be obtained through backpropagation, they add to the computational complexity of the algorithm.

Remark 15. Since each update is based on a single trajectory roll-out and backpropagation and solving the linear program, (4.47) or (4.49), it can be performed very fast and is suitable for online implementation. In practice, in our receding horizon framework, for Case 1, a single update based on (4.47) can be done at each time τ. For Case 2, we can perform multiple updates at each time τ based on (4.49) until we reach a parameter θ that is $T+1$ feasible. Since SQP in general does not provide guarantees for finite time convergence to local optimum for generic nonlinear optimization problems, a thorough analysis of conditions on system dynamics and initial parameter choice under which we can guarantee convergence is outside the scope of this dissertation and will be addressed in future work. We show the efficacy of our method through an extensive simulation study in Section 6.5. Our work thus brings together elements from SQP, DDP, and backpropagation based gradient computation for computational graphs to propose an automated tuning framework for controllers.

4.4.7 Simulation Results

We present results for two case studies. The first one corresponds to the autonomous car model of Example 4, and applies our algorithm to increase the horizon over which the system is feasible. The second one corresponds to the leader-follower problem of Example 3, and shows how parameter adaptation can improve the high-level objective. The optimization problems are solved in python with cvxpy [78] interface and SCS solver. cvxpy also returns the gradient of QP solution with respect to its parameters. The dynamics of robotic agent and cvxpy optimization are made part of PyTorch layers and once the computation graph is made, backpropagation is done with PyTorch's Autograd feature[4].

[4]The videos and the code implementing the algorithm can be found at https://github.com/hardikparwana/Safe-Learning-DASC and on thesis website https://github.com/hardikparwana/PhD-Thesis

4.4.7.1 Autonomous car with shrinking safe set

Fig.4.11 shows the results from applying algorithm from Section 4.4.6 to autonomous car problem. The constrained GD is able to update parameter while ensuring that the system remains feasible for the same or longer time horizon.

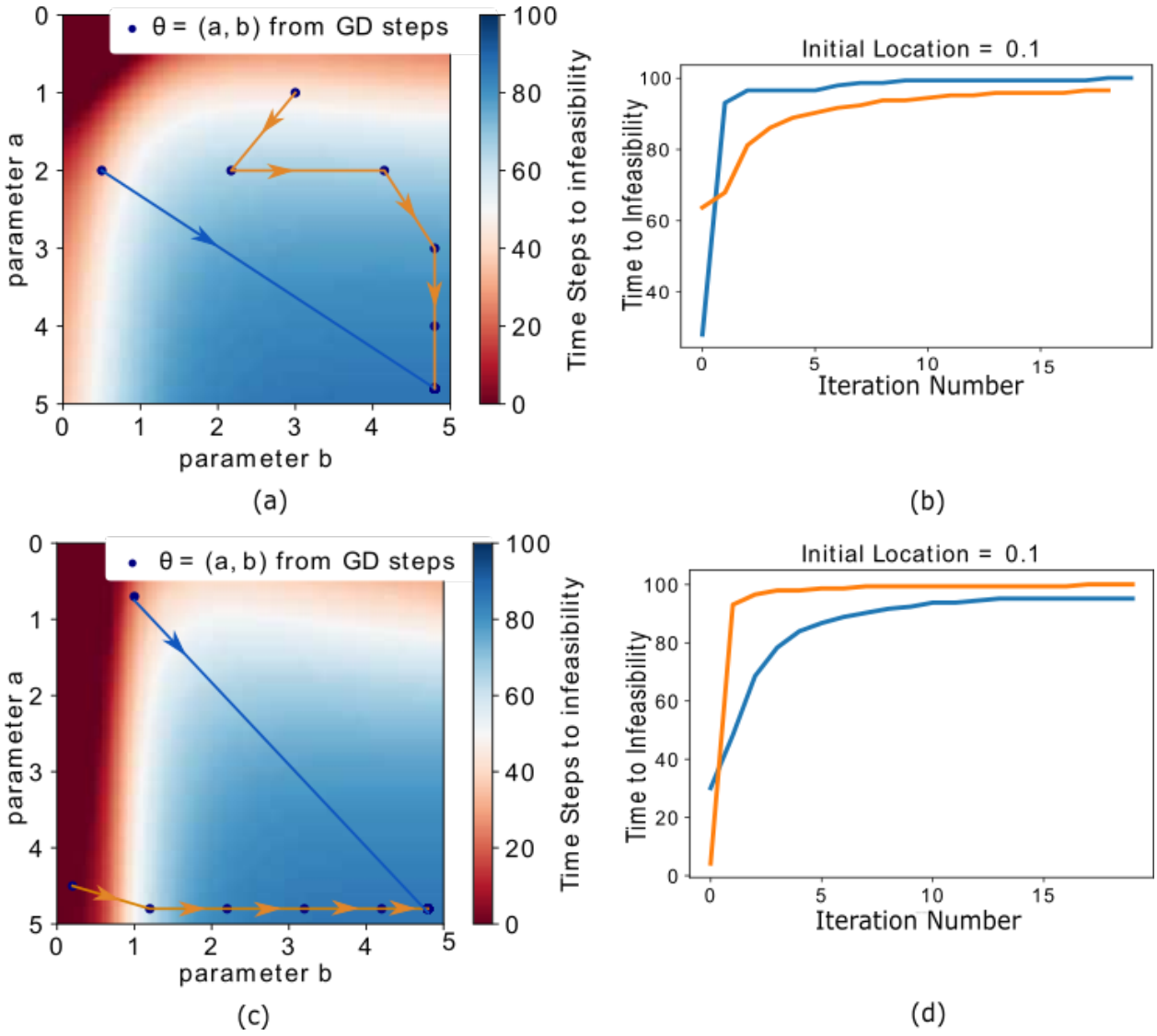


Figure 4.11: GD iterates: All figures are for c=0.3 in Example 4. Initial state is 0.5 for (a),(b), and 0.1 for (c),(d). Each shows results for two different initialization of θ. (a), (c) show the change in θ. Points on same GD process are connected by line and direction of update marked with arrows. (b), (d) show plots of time to infeasibility with GD iteration number. Parameter values have been clipped to range (0,5) to better visualize infeasible parameters near top and left edges of the box. The values of a and b shown are normalized by Δt.

4.4.7.2 Predictive Framework vs MPC

Here we do a comparison of trajectories obtained from the proposed receding horizon approach for tuning CBF controllers and MPC. We validate our approach on two different robots. The first, a single integrator, has following dyamics

$$\dot{p}_x = u_1, \dot{p}_y = u_2, \tag{4.54}$$

and second, a mobile robot with the following dynamic unicycle dynamics:

$$\dot{p}_x = v\cos\phi,\ \dot{p}_y = v\sin\phi,\ \dot{v} = u_1,\ \dot{\phi} = u_2, \tag{4.55}$$

where p_x, p_y are the location of the robot in horizontal plane, ϕ is the heading angle measured w.r.t X axis, u_1, u_2 are the control inputs. The objective is to navigate around two obstacles to reach a goal location $x_g = [g_x\ g_y]^T$ as shown in Fig.4.12 with input bounds $|u_1|, |u_2| \leq 3$ for single integrator and $|u_1| \leq 2, |u_2| \leq 5$ for dynamic unicycle. For collision avoidance with the two circular obstacles, the barrier function is designed as $h_i(x) = s_i^2 - s_{min}^2$ where s_i is the distance of the robot from the center of i^{th} circular obstacle and s_{min} is the collision avoidance radius. The stage-wise reward for high-level optimization (4.46) is chosen as $r(x_t, u_t) = -10\left((p_x - g_x)^2 + (p_y - g_y)^2\right) - ||u||^2$. The nonlinear MPC problem is also solved to optimize $\sum r(x_t, u_t)$ the same stage-wise cost to find a path that maintains $h_i(x) \geq 0$. The simulation time is $4s$ with time steps of $0.01s$ and the prediction horizon for the proposed method is taken as 20 with an initial learning rate of 0.07. and is solved with IPOPT using cyipopt library in Python. Our method always improves the solution resulting from initial choice of parameters. Note that in Fig.4.12b, Case 1 nominal parameters fail to ensure the existence of a solution to QP and so we relax the CBF constraints by adding a slack variable with a corresponding large weight in the objective function. The resulting solution thus does not ensure safety but gives minimum violating trajectory. The proposed method, on the other hand, is able to tweak the gains to ensure persistent compatibility.

4.4.7.3 Predictive Framework for Leader-Follower Problem

We consider a robot, called follower, modeled by unicycle kinematics, equipped with a forward-looking sensor (e.g., camera). The follower's objective is to keep another robot, called leader, within its field-of-view (FoV). The leader robot is modeled by single-integrator dynamics and is assumed to be moving independently of the follower robot. The FoV constraint specifies a time-varying safe set obtained as the intersection of the following three

barrier conditions

$$h_1 = s^2 - s_{min}^2 \geq 0, \tag{4.56a}$$

$$h_2 = s_{max}^2 - s^2 \geq 0, \tag{4.56b}$$

$$h_3 = b - \cos(\gamma) \geq 0, \tag{4.56c}$$

where s is the distance between the follower and the leader, s_{min} is the minimum allowed distance for collision avoidance, s_{max} is the maximum allowed distance for accurate detection using the onboard camera, b is the bearing vector from the follower to the leader, and γ is the FoV angle of the follower's camera. The CBF-CLF-QP (4.43d) uses the aforementioned three first-order barrier functions and the Lyapunov function

$$V(x) = (s - \frac{s_{min} + s_{max}}{2})^2, \tag{4.57}$$

The linear and angular velocity of the unicycle follower is constrained to be under 2.5 m/s and 4.0 rad/s in magnitude. We use linear class-$\mathcal{K}$ functions with first-order barriers h_1, h_2, h_3 having parameter $\nu_1, \nu_2\nu_3$. The predictive framework from Section 4.4.6 is used to auto-tune the controller gains ν_1, ν_2, ν_3, k. The objective is to keep the controller (4.43d) feasible for all $t > 0$, and to maximize the reward in (4.46) where $r(x_t, u_t) = V(x) - 10h_3$, that is, the reward is maximum when the leader is at the center of field-of-view. The leader is made to move aggressively with a constant horizontal velocity of $\dot{p}_x = 1$ m/s and sinusoidal vertical velocity of $\dot{p}_y = 12 \sin\sin(4\pi t)$ m/s. s_{min}, s_{max}, γ are chosen to be $0.3, 2.0, \pi/3$ respectively. We simulate for two cases: unbounded and bounded control input and compare the proposed approach with a fixed parameter case, leading to a total of four comparisons. The videos and code can be found on our website. The results are shown in Fig.4.13. We see that unbounded input with online adaptation leads to the lowest cost in Fig. 4.13b. Despite not having input bounds, it demands smaller control inputs compared to fixed parameter case as seen in Fig.4.13a. When the control input is bounded, the fixed parameter case leads to infeasible QP and the simulation ends after 0.1 seconds. Our proposed method is able to quickly adapt parameters to maintain input bounds without violating barrier constraints as is evident in Figs.4.13a and 4.13c.

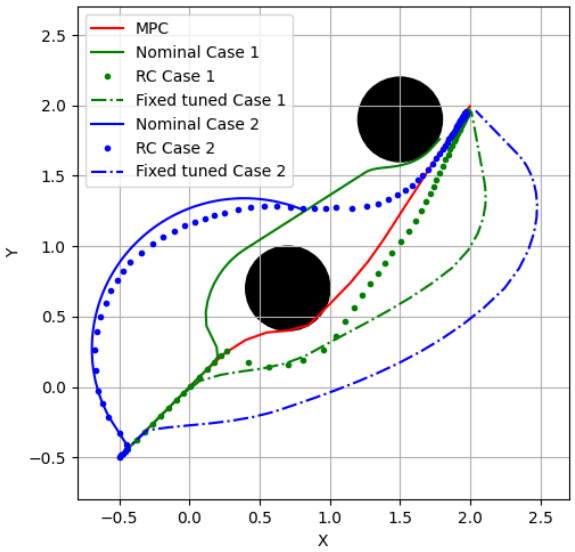
We also show the variation in performance as a function of prediction horizon. At each time instant, a single step of GD is done for a look-ahead horizon that ranges from 0 to 30 in simulations where a horizon of 0 implies the non-adaptive case. Fig. 4.14 shows the rewards received in Eq.(4.46) for the proposed algorithm. Videos can be found on the thesis website and on thesis website https://github.com/hardikparwana/PhD-Thesis.

4.4.7.4 Computational Complexity for factored graphs

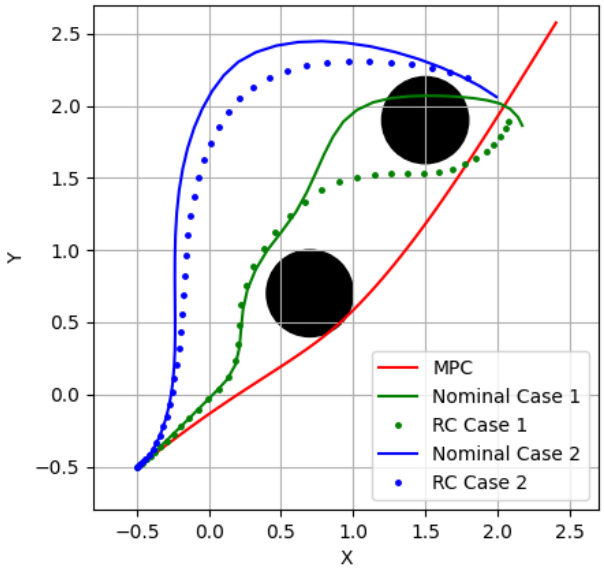
A thorough analysis of computational complexity is outside the scope of this chapter. For simplicity, we only evaluate number of gradient computations. For every function dependent on x_t, u_t, the number of gradients in backpropagation can be easily deduced to be $4t-1$ by counting edges that need to be traversed when going back to initial state x_1 and parameter vector θ. Since we have N hard constraints and one reward at each time $t \in \{1, 2.., T\}$, the total number of gradients required are $\sum_{t=1}^{T}(N+1)(4t-1) = (N+1)T(T+1)$. In practice, however, the constraints that are already satisfied by a large margin need not be considered for feasibility maintenance as small changes in parameters will not affect them. Also, the updates may be performed based on linearization about past trajectory instead of predicted one as it would offer huge boost in speed.

4.4.8 Conclusion and Future Work

This chapter details a framework to update parameters of QP controller by employing back-propagation on a computational graph over time. Sensitivity analysis was used for the first time and combined with SQP algorithms to ensure that feasibility of the process over the horizon of interest is maintained or improved while simultaneously achieving performance gains. Future work will investigate experimental verification on different robotic platforms, convergence rates guarantees, and the effect of uncertain dynamics. Since differentiating through conic programs is not limited to QPs and LPs, more generic optimization based controllers will also be investigated. While the unicycle example showed how algorithm would perform for a nonlinear dynamical system, the algorithm needs to be tested for feasibility on more systems and for infeasibility resulting from control input bounds. We also plan to use more general class-$\mathcal{K}$ functions for imposing barrier and lyapunov conditions.

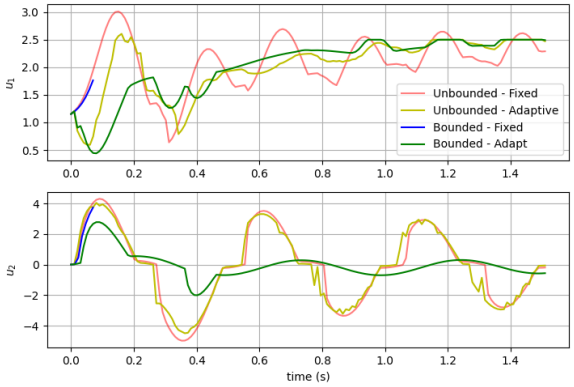


(a) Single Integrator: Initial parameter: Case 1: $k = 1, \nu_i = 3$; Case 2: $k = 0.5, \nu_i = 0.5, i = \{1, 2\}$ in (4.43)

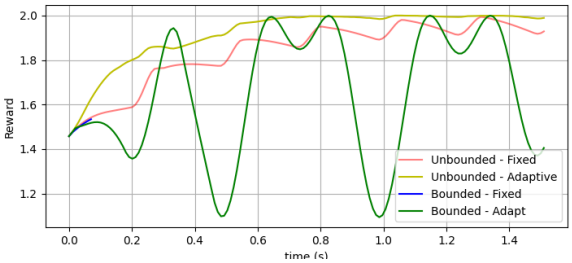


(b) Dynamic Unicycle: Initial parameter: Case 1: $k = 1, \nu_i^0 = 5, \nu_i^1 = 20$; Case 2: $k = 1, \nu_i^0 = 1, \nu_i^1 = 3$ in (4.43). MPC solution first overshoots but comes back to the goal at (2,2).

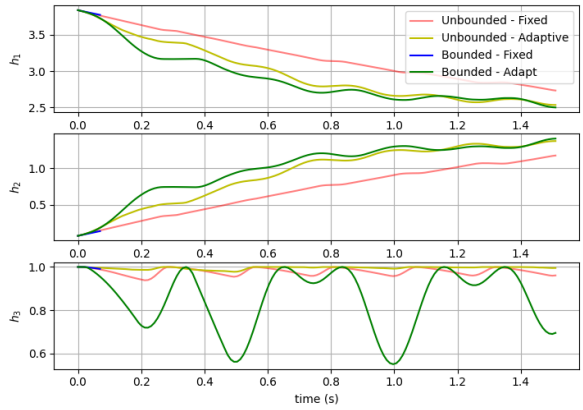
Figure 4.12: Comparison of MPC with the proposed method for two different initial parameter values given by blue and green trajectories. Receding horizon(RC) corresponds to gradient descent steps (4.53) applied as robots move in the environment. Fixed-tuned trajectories result from performing 20 iterations of (4.53) before actually moving the robot in the environment. Goal is at (2,2). Start location is at (-0.5,-0.5).



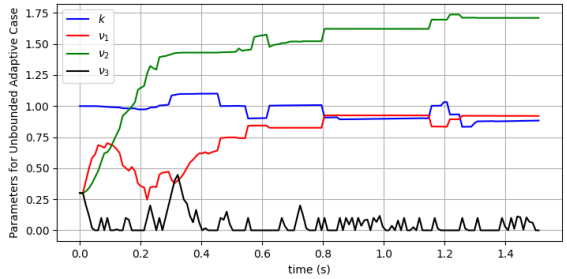
(a) Follower's input with time.



(b) Variation of moving horizon reward with time. Adaptation with unbounded input is able to achieve the highest reward as expected.



(c) Constraint functions h_1 (minimum distance), h_2 (maximum distance) and h_3 (maximum angle w.r.t. camera axis) with time. Constraint maintenance corresponds to $h_i > 0, i = 1, 2, 3$.



(d) Variation in the four parameters for leader-follower problem

Figure 4.13: Leader-Follower Problem

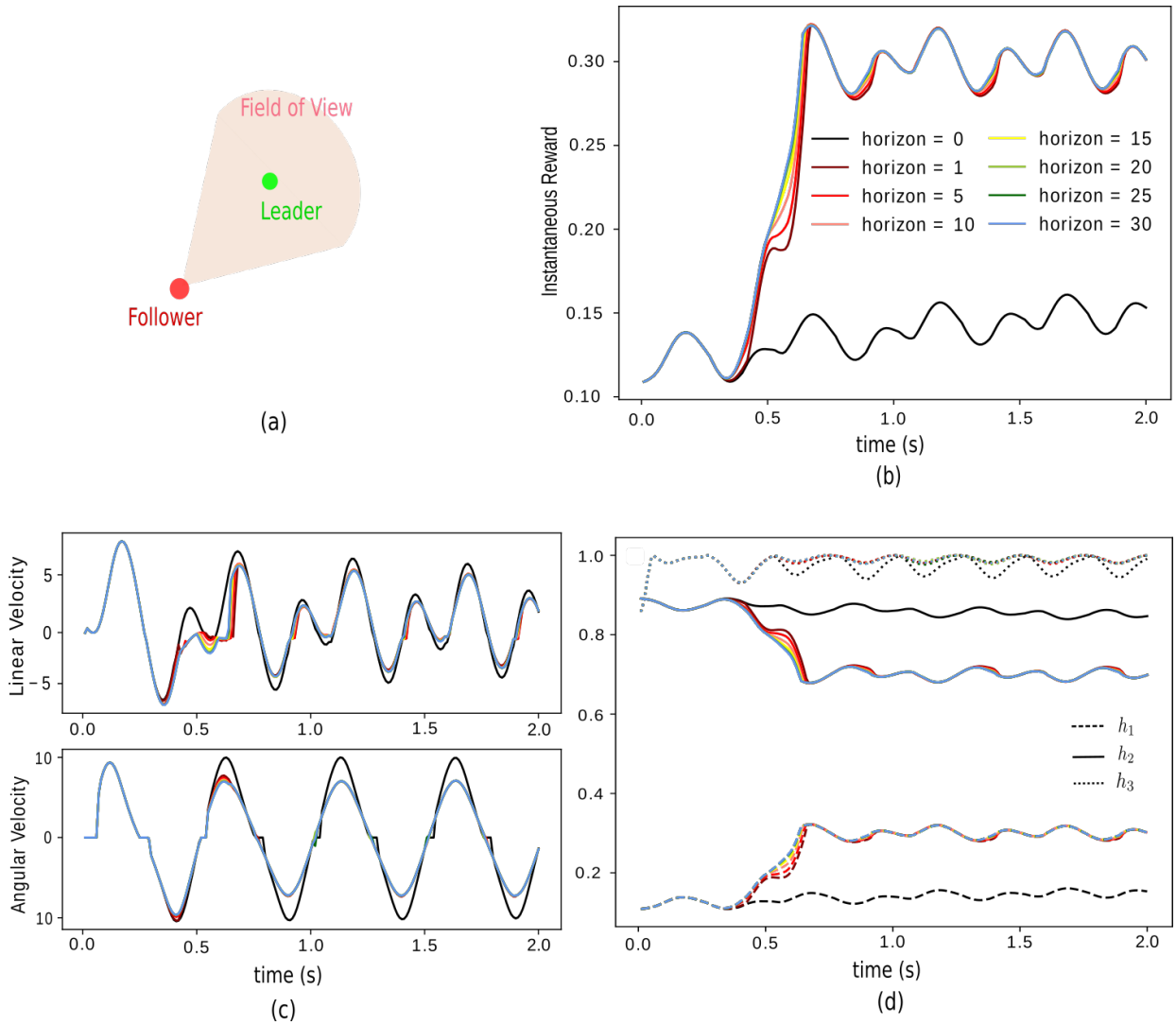


Figure 4.14: Leader-follower simulation comparing different horizons for high-level optimization. (a) shows the scenario considered and (b) shows the rewards received with time. The adaptive parameter case manages to achieve higher rewards. (c) shows the control input variation and (d) shows variation of barrier functions with time. h_1 and h_2 represent minimum and maximum distance barriers respectively and converge to the value corresponding to desired relative location of leader.

CHAPTER 5

Enter the Uncertainties FORESEE - State Prediction under Uncertainties

In the previous chapters, we assumed that dynamics is known and deterministic. In this chapter, we relax this assumption and build up upon our predictive framework in Section 4.4 for auto-tuning controller (policy) parameters for a dynamical system with generic state-dependent stochastic uncertainties. Note that this chapter as well as the predictive framework introduced in Section 4.4 are applicable irrespective of the choice of controller (policy). These methods may use any parametric controller like neural networks, PID, or CBF-QP.

The main challenge in using the predictive frameworks for policy tuning/optimization is to predict the distribution of future states. This problem admits no analytical solution under generic probability distributions. This chapter introduces a new numerical scheme based on the Unscented Transform (popularly known for its application in Unscented Kalman Filter (UKF)) to approximate state distributions.

5.1 Literarture Review and Contributions

5.1.1 State Prediction under Uncertainties

The ability to adapt the parameters of the control policy of an autonomous system (e.g., a robot) by quantifying the uncertainties present in the system, and evaluating the performance of the policy as a function of its parameters, is crucial to enabling system autonomy. We introduce a policy optimization framework that we call FORESEE (4C): Foresee for Certified Constrained Control, which uses stochastic estimates of system dynamics to predict the future state distribution, and then adapts policy parameters to improve performance while ensuring the probabilistic satisfaction of state and control input constraints. To this end, we

consider the problem of online optimizing the parameters of a given policy under constraints and uncertain states. We propose a novel, computationally-efficient method for the online prediction of nonlinear state distributions using the Unscented Transform, and interface it with a constraint-aware gradient-descent policy that adapts the policy parameters online for performance improvement.

5.1.1.1 Earlier work on state prediction for nonlinear dynamics

Predicting a nonlinear stochastic distribution is, in general, analytically intractable [76]. Several approximate analytical and numerical methods have therefore been proposed. A popular one that is still prevalent was proposed in PILCO [75], which at each time step propagates a Gaussian distribution, approximates the resulting non-Gaussian distribution with a Gaussian distribution, and repeats this procedure. The idea since then has also been used in several other works on stochastic Model Predictive Control (MPC) [115], and Differential Dynamic Programming (DDP) [222, 168] approaches. However, the higher-order moments of the distribution are ignored, which can lead to significant errors when the original distributions have high variance, and when the transition dynamics is highly nonlinear. Furthermore, repeated approximations over multiple time steps causes the prediction to drift away from the true distribution [236]. Also, the analytical expressions hold only for Gaussian kernels in GP, they are restrictive in practice as different kernels are preferred depending on the environment [198]. Model learning algorithms other than GP have also been used. Deep PILCO [88] uses Bayesian Neural Network (NN), but it still relies on the same moment-matching principle as PILCO.

Some works have tried to alleviate this problem by maintaining higher-order moments, or by approximating distributions with more generic parametric representations. [205] and [13] employ Monte Carlo (MC) to estimate future rewards and their gradients. The authors in [166] used Unscented Transform (UT) to propagate uncertainty in the presence of Gaussian noise, and improve an initially-feasible trajectory with DDP. The approach in [236] represented distributions as a mixture of Gaussians, maintaining coefficients for basis functions, and used Gaussian quadrature rule to propagate the coefficients over multiple time steps.

The above methods, while leading to more accurate results than the Gaussian approximation, have their own disadvantages. MC requires an excessively large number of samples and is computationally expensive [163, 95]. [166] used the standard Unscented Transform, which ignores moments higher than the covariance when choosing sigma points. The approach in [166] considers random perturbations that are independent of the state, and hence does not exhibit the scalability issue associated with an increasing number of sigma points in presence of state-dependent perturbations that we discuss later. [236] requires an offline analysis to

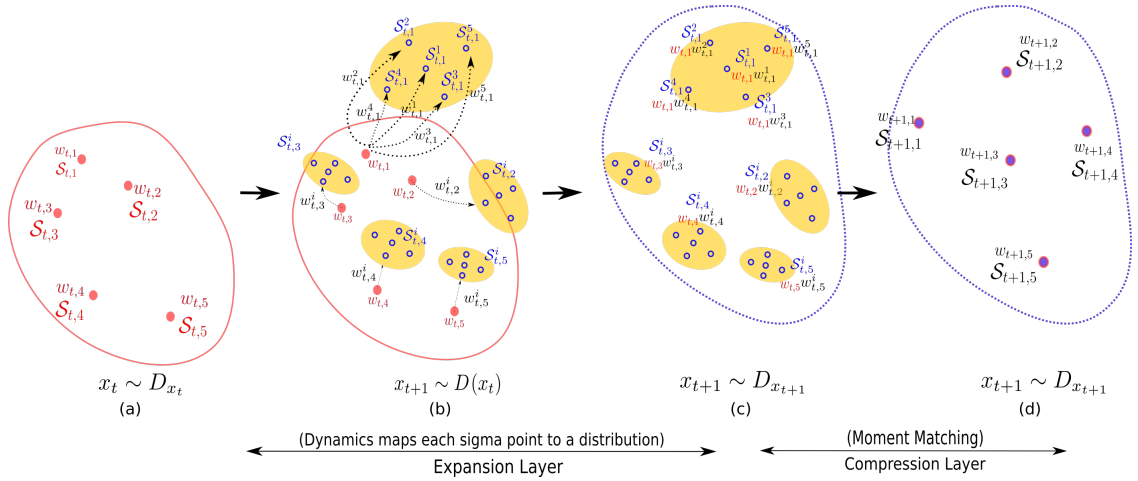


Figure 5.1: The proposed Expansion-Contraction mechanism for predicting the distribution of state x_{t+1} given the distribution of state x_t. In this abstract 2D ($n = 2$) example, UT and GenUT choose 5 ($= 2n + 1$) points to represent a distribution (a) The red particle (sigma point) $\mathcal{S}_{t,i}$ and their weights $w_{t,i}$ represent the distribution of state x_t. (b) Since the uncertainty is state-dependent, each red point $\mathcal{S}_{t,i}$ gives rise to the states in the yellow region, each of which is represented by 5 blue points $\mathcal{S}^j_{t,i}$ and weights $w^j_{t,i}$ in the expansion layer. (c) The overall weight of $\mathcal{S}^j_{t,i}$ is obtained by multiplying the weight of the edge $w^j_{t,i}$ with the weight of the corresponding root node $w_{t,i}$. (d) The 25 points resulting from the expansion layer are then compressed back to five red-blue particles in the compression layer to represent the blue distribution of x_{t+1}.

design the quadrature rule.

5.1.1.2 Earlier gradient-based approaches for parameter update

A key element in most of the aforementioned works is the use of gradient-based optimization for policy update. There are other approaches [56, 130, 88] that perform gradient-free optimization. While most of these gradient-free methods are also able to incorporate models for the system dynamics other than GPs, they either rely heavily on parallelization for efficient implementations [56, 130], or require extensive offline optimization [88]. Although these methods present many advantages, here we focus on gradient-based methods that are amenable to online implementation on less powerful (computationally) platforms, e.g., small aerial robots.

The system, and hence its control policy, is often required to satisfy strict state and input constraints. [268] adds a safety layer based on control barrier functions, and [61] poses safe RL as a stochastic MPC problem. Some works directly solve the Constrained Markov Decision Problem [212] to learn a safety-critical policy. In general, model predictive frameworks can be used in tandem with nonlinear optimization techniques to find a constraint-respecting policy. For example, SafePILCO [190] extends PILCO to satisfy state constraints. However, few

techniques are suitable for online implementation. The approaches in [19, 233] use DDP to incrementally improve the trajectory online and solve the constrained RL problem. However, they only consider the mean of learned model dynamics when predicting the future.

5.1.1.3 Contributions and comparison to the authors' earlier work

The contribution of this chapter is the integrated framework on state prediction and policy parameter adaptation, which we call "FORESEE". The framework comprises two components, which can be viewed as a primary contribution and a secondary contribution, respectively. As the primary contribution, we propose a novel method for predicting future state distributions for nonlinear dynamical systems with state-dependent uncertainty using the Unscented Transform (UT) and its variants [82]. Note that although GPs and BNNs have been prevalent in robotics, state-dependent probability distributions also arise in several other emerging uncertainty quantification schemes such as conformal prediction [21], deep ensembles [185], variational inference [16] and hence have significant interest from the application standpoint. UT methods maintain a finite number of weighted particles or sigma points that have the same sample moments as the true distribution. However, propagating a finite number of sigma points through a model with state-dependent uncertainty requires an increase in the number of points that need to be maintained with each time step to keep track of the evolving state distribution. *To the best of our knowledge, a method on propagating a given finite number of particles through a dynamics model with state-dependent perturbation distribution is missing, and we address it in this chapter* . In order to keep our algorithm scalable, we add a moment-preserving sigma-point compression layer that preserves the number of sigma points constant across the time steps. An illustration is shown in Fig. 5.1. Our empirical results show that with a finite number of sigma points, our method performs satisfactorily, comparable to MC simulations but at a much lower computational cost.

As the secondary contribution, inspired by DDP approaches [166], we show how our proposed method for state prediction using the Expansion-Compression Unscented Transform can be used in combination with a gradient-descent approach based on Sequential Quadratic Programming (SQP) for *online* policy optimization, and validate our approach both in simulations and experiments. More specifically, the proposed state prediction allows us to develop a model predictive framework that can be implemented as a computational graph whose nodes represent states and control inputs. Backpropagating through this graph allows us to use the policy gradients for policy optimization. For the policy gradients to be well-defined, we use UT methods [113, 82] that admit analytical expressions and are differentiable almost everywhere, except for the case when the covariance matrix is singular. Furthermore,

by adopting and building upon the SQP approach, we define an online tuning mechanism for controllers, which is desirable in the face of uncertainties and constraints. The efficacy of the method is demonstrated by case studies addressing online controller tuning both in simulation and hardware experiments. Notably, the proposed state prediction scheme can be used with any nonlinear optimizer, similarly to other approaches, to obtain an optimal policy; a case study addressing offline trajectory optimization is included in the simulation results.

This chapter builds upon our earlier work in Chapter 4 (published in [175]), in which we introduced a preliminary version of the online parameter optimization framework for *deterministic* systems. This chapter thus includes the method of handling *uncertain* dynamics, which corresponds to the introduction of the novel, data-efficient state prediction method that we call the Expansion-Compression Unscented Transform. The proposed expansion-compression layers of our UT-based algorithm constitute, to the best of our knowledge, an important and standalone contribution to state prediction for online policy optimization under uncertainty, and the major contribution of the current chapter. The UT-based method does not appear in [175].

5.1.1.4 Chapter Organization

Section 5.1.2 describes the class of systems considered and the problem statement. Section 6.4 introduces our methods for state prediction, as well as for policy optimization. Section 5.4.1 includes a number of simulation case studies. We compare the empirical performance of our expansion-compression prediction method against Monte Carlo simulations and a successive approximation method for *state-dependent* distributions. Next, we show an application of our expansion-compression layer to solve a stochastic nonlinear MPC problem with IPOPT, which is an off-the-shelf available solver. We then show the efficacy of our framework for online tuning of policy parameters on two different case studies. First, we consider a leader-follower problem, where the follower's policy is computed as the solution to a Quadratic Program (QP) subject to multiple Control Barrier Function (CBF) constraints and input constraints. Second, we use our algorithm for a quadrotor stabilization task on safe-control-gym [264] and compare its performance and computational characteristics with the baseline of GP-MPC [106]. Finally, in Section 5.4.6 we present experimental evaluation of the framework on quadrotor tracking control, and specifically the efficacy of FORESEE in auto-tuning the controller of a real quadrotor platform in the presence of wind disturbances.

5.1.2 Problem Statement and Approach

Consider a nonlinear system with state $x \in \mathcal{X} \subset \mathbb{R}^n$, input $u \in \mathcal{U} \subset \mathbb{R}^m$. The discrete-time state dynamics is given by

$$x_{t+1} = f(x_t, u_t), \tag{5.1}$$

where $t \in \mathbb{Z}^+$ denotes the time index, $x_t \sim D_{x_t}$ is the current state that follows a known nonlinear distribution D_{x_t}, x_{t+1} is the predicted state, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a deterministic function for the system dynamics. When f is unknown, e.g., when the system dynamics are hard to model with physics-based principles, we assume that the state-transition model

$$x_{t+1} \sim D(x_t, u_t) \tag{5.2}$$

is available where x_t is the current state, x_{t+1} is the predicted state, $t \in \mathbb{Z}^+$ denotes the time index, and D is a nonlinear probability distribution. For example $D = N(\mu(x_t, u_t), \Sigma(x_t, u_t))$, a $n-$dimensional probability distribution whose mean and covariance may depend on the state and control input at time t. In practice, D can be obtained via model-learning algorithms such as Gaussian Process (GP) or Bayesian Neural Network. We assume that the true dynamics is a sample of the learned model, $f(x_t, u_t) \sim D(x_t, u_t)$.

We furthermore consider K constrained sets $\mathcal{C}_i \subset \mathcal{X}$ defined as the zero-superlevel sets of corresponding smooth constraint functions $c_i : \mathcal{X} \to \mathbb{R}, i \in \mathcal{K} := \{1, 2, .., K\}$, such that

$$\mathcal{C}_i \triangleq \{x \in \mathcal{X} \mid c_i(x) \geq 0\}. \tag{5.3}$$

Each state-action pair (x_t, u_t) results in a reward from the environment, given as $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$. A policy $\pi : \mathcal{X} \to \mathcal{U}$ that maximizes the expectation of the cumulative reward for $t \in \mathcal{T} := \{0, 1, 2, \ldots, H\}$ while forcing the probabilistic satisfaction of the constraints at each time step t is given as:

$$\pi^* = \arg\max_{u_t \in \mathcal{U}} \quad \mathbb{E}\left(\sum_{t=0}^{H} r(x_t, u_t)\right), \tag{5.4a}$$

$$\text{s.t.} \quad P[c_i(x_t) \geq 0] \geq 1 - \epsilon, \quad \forall i \in \mathcal{K}, \forall t \in \mathcal{T}, \tag{5.4b}$$

where $u_t = \pi(x_t)$ and $\epsilon \in [0, 1]$ is a user-defined risk.

In this chapter , we mainly focus on the case when the policy u_t is a parameterized policy in terms of parameters $\theta \in \mathbb{R}^\kappa$, such as a neural network (NN), or a P controller in which NN weights or the P gain are viewed as a tunable parameter. With slight abuse of notation,

we also use $D(x_t)$ to refer to the closed-loop dynamics wherein u_t either takes a specific value or is given by a state-feedback policy π. Note that in this case, D is understood to change with parameter θ. In that case, the optimization (5.4) is reformulated in terms of θ as shown below. We are furthermore motivated by problems where optimization over arbitrarily large horizons might not be available or desirable (e.g., due to computational constraints). Hence, we seek to develop receding-horizon policies that satisfy the given constraints with high probability, as well as a method for state prediction that is computationally-efficient for implementation along with the online policy optimization. In summary, we consider the following problem:

Problem Statement: Given a system modeled as in (5.2), and a control policy $\pi : \mathcal{X} \times \mathbb{R}^{\kappa} \to \mathcal{U}$ such that $u_t = \pi(x_t, \theta)$, we seek at each time $t \in \mathbb{Z}^+$ to
(**P.1**) find the parameter θ^* that maximizes the expected reward over a finite time horizon H subject to probabilistic safety constraints, i.e., solves for

$$\theta^* = \arg\max_{\theta} \quad \mathbb{E}\left(\sum_{\tau=0}^{H} r(x_{t+\tau}, u_{t+\tau})\right), \tag{5.5a}$$

$$\text{s.t.} \quad P\left[c_i(x_{t+\tau}) \geq 0\right] \geq 1 - \epsilon, \quad \forall i \in \mathcal{K}, \forall \tau \in \mathcal{T}, \tag{5.5b}$$

where $u_{t+\tau} = \pi(x_{t+\tau}; \theta)$, τ is the prediction time step, and $x_{t+\tau}$ is the predicted future state at time $t + \tau$, and
(**P.2**) develop a method for obtaining computationally-efficient predictions of $x_{t+\tau}$.

Approach: To solve **P.2** we propose a novel prediction technique based on Unscented Transform (UT) that is accurate and data-efficient due to the proposed Expansion-Compression (EC) layer. To solve **P.1**, we convert the stochastic optimization problem (5.5) to an approximate deterministic optimization problem that can be solved by any available nonlinear solver, e.g., IPOPT. Additionally, when offline optimization is too expensive to perform, we propose an online policy update algorithm that is inspired by Sequential Quadratic Programming and employs backpropagation through the computational graph generated by the states of the dynamical system (5.1). The overall method, called FORESEE, comprising the prediction and online optimization phases, is described in detail in Section 6.4. In the next subsections we provide some preliminaries that help establish the notation and framework for FORESEE.

5.2 State Distribution Prediction with Unscented Transform

Unscented Transform [113] is a method to represent a probability distribution through a finite set of weighted particles, called sigma points and sigma weights from here forth.

Let a given state $x_t \sim D_{x_t}$ at time t. Consider a set $\mathcal{S}_t$ of N representative sigma points, $\mathcal{S}_{t,i}$, $i = 1, \ldots, N$, denoted $\mathcal{S}_t = \{\mathcal{S}_{t,1}, \mathcal{S}_{t,2}, \ldots, \mathcal{S}_{t,N}\}$, and the set $w_t = \{w_{t,1}, w_{t,2}, \ldots, w_{t,N}\}$ comprising their associated weights $w_{t,i}$. The sample (or empirical) mean and covariance of x_t are computed as follows:

$$\mathbb{E}_s[x_t] = \sum_{i=1}^{N} w_{t,i} \mathcal{S}_{t,i}, \tag{5.6}$$

$$\Sigma_s[x_t] = \sum_{i=1}^{N} w_{t,i} (\mathcal{S}_{t,i} - \mathbb{E}_s[x_t])(\mathcal{S}_{t,i} - \mathbb{E}_s[x_t])^T. \tag{5.7}$$

Given the moments of a distribution of x_t, UT provides a principled way of selecting the sigma points $\mathcal{S}_t, w_t$ such that their empirical mean, covariance, and possibly other higher-order moments are equal to the moments of the given distribution. The standard UT algorithm [113] is presented in Algorithm 2 and, for a given mean and covariance of a n-dimensional distribution, it selects the best $N = 2n + 1$ sigma points $\mathcal{S}_{t,i}$ and weights $w_{t,i}$, , $i \in \{1, 2, \ldots, 2n + 1\}$, whose empirical mean and covariance $\mathbb{E}_s[x_t], \Sigma_s[x_t]$ respectively, as defined above, match the given moments. Note that the standard UT in Algorithm 2 requires a free parameter $k \in \mathbb{R}$ that is chosen by the user. In theory, the optimal choice of k depends on the original distribution and is used to better represent higher-order moments [113]. In practice, this is usually computed for specific distributions beforehand. For a priori unknown distributions, sigma point adaptation algorithms can be used, see [228].

5.2.1 Generalized Unscented Transform

Several variations of vanilla UT have been proposed over the years to better preserve higher-order moments of a probability distribution. In general, a system of polynomial equations needs to be solved which is not amenable to analytical solutions. In this section, we introduce a specific variation, known as Generalized Unscented Transform (GenUT) [82], that we use extensively in our work. GenUT also admits analytical solutions, uses only $2n + 1$ points (same as vanilla UT), and can preserve diagonal components of skewness and kurtosis tensors of a distribution. Unlike vanilla UT, the ability to maintain components of skewness tensor renders it capable of representing asymmetric distributions. Algorithm 3 presents GenUT

Algorithm 2 generate_UT_points

Require: μ, Σ ▷ mean, covariance of the distribution

1: Choose a free parameter $k > 0$
2: Calculate $2n + 1$ sigma points

- $\mathcal{S}_0 = \mu$
- $\mathcal{S}_i = \mu + \sqrt{(n+k)\Sigma_i}, \mathcal{S}_{i+n} = \mu - \sqrt{(n+k)\Sigma_i}$

for $i \in \{1, .., n\}$

3: Calculate weights for the sigma points

- $w_0 = k/(n+k)$
- $w_i = 1/(2(n+k)),\ w_{i+n} = 1/(2(n+k))$

for $i \in \{1, .., n\}$

4: Return $\mathcal{S}, w$

for selecting sigma points and weights.

Algorithm 3 generate_GenUT_points: Generalized Unscented Transform [82] for finding $2n + 1$ sigma points and weights.

Require: $\mu \in \mathbb{R}^{n\times 1}, \Sigma \in \mathbb{R}^{n\times n}, S \in \mathbb{R}^{n\times 1}, K \in \mathbb{R}^{n\times 1}$ ▷ mean, covariance, diagonal skewness, diagonal kurtosis of the distribution

1: $u = 0.5(-S + \sqrt{4K - 3S^2})$ ▷ $n \times 1$ vector
2: $v = u + S$ ▷ $n \times 1$ vector
3: $w_2 = (u + v)./v$ ▷ n weights. Division done elementwise
4: $w_1 = (w_2 . * v)./u$ ▷ n weights. Multiplication and division done elementwise
5: $w0 = 1 - \text{sum}(w_1) - \text{sum}(w_2)$
6: $U = \text{diag}(u), \quad V = \text{diag}(v)$ ▷ two $n \times n$ matrices
7: $\mathcal{S} = \text{Append}(\mu, \mu - \sqrt{\Sigma}U, \mu + \sqrt{\Sigma}V)$ ▷ $2n + 1$ vectors
8: $w = \text{Append}(w_0, w_1, w_2)$ ▷ $2n + 1$ weights
9: Return $\mathcal{S}, w$

5.3 Proposed Approach: FORESEE

This section presents our method (Algorithm 4) on state prediction by propagating sigma points across multiple time steps, and on policy-parameter adaptation using constrained gradient descent to respect state and input constraints. The state prediction is presented in lines 5-10 of Algorithm 4. Constrained gradient descent implemented in lines 11 and 12 is then detailed in the next subsection.

Algorithm 4 FORESEE

Require: x_0, H, β, π_θ ▷ initial state, time horizon, learning rate for policy update, parameterized policy

1: Initialize $\mu_0 = \mathbb{E}[x_0]$
2: **for** N = 1 to num_trials **do**
3: J = 0 ▷ Cost function
4: $\mathcal{S}_0, w_0$ = generate_UT_points($\mu_0, 0$)
5: **for** t = 1 to T **do** ▷ Predicting future
6: $\mathcal{S}_{next}, w_{next} \leftarrow$ expand_sigma_point$(\mathcal{S}_{t-1}, w_{t-1}, D(\cdot; \pi_\theta))$
7: $J \leftarrow J + \mathbb{E}(r(\mathcal{S}_{next}, \pi(\mathcal{S}_{next})))$
8: $\mathcal{S}_t, w_t \leftarrow$ compress_sigma_points$(\mathcal{S}_{next}, w_{next})$
9: **end for**
10: **if** Online Update **then**
11: Find update direction d_θ using (5.14) or (5.15)
12: Update $\theta \leftarrow \theta + \beta d_\theta$
13: **end if**
14: **end for**

5.3.1 Expansion-Compression Layers

Given N sigma points and weights $\mathcal{S}_t, w_t$ representing the state x_t at time t, line 6 predicts the sigma points representing the state x_{t+1} based on the dynamics function $D(x_t)$. Each sigma point $\mathcal{S}_{t,i}$ is mapped to a distribution $D(\mathcal{S}_{t,i})$ based on dynamics $x_{t+1}|x_t = D(x_t)$ and Algorithm 2 is used to sample new sigma points from this distribution. This leads to an increase in the total number of sigma points that is not scalable for multi-step prediction. Therefore, after evaluating the cost in Line 7, Line 8 performs an operation based on Algorithm 5 to compress the sigma points $\mathcal{S}_{new}$ to obtain N new sigma points. The procedure is repeated for H time steps.

Algorithm 5 compress_sigma_points

Require: $\mathcal{S}, w$ ▷ Collection of sigma points and weights

1: μ_s, Σ_s = Sample mean, sample covariance (sigma points)
2: Return generate_UT_points(μ_s, Σ_s)

Expansion Layer Here we describe our sigma-point expansion layer in Algorithm 6. For every sigma point $\mathcal{S}_{t,i}$ of x_t, we obtain the moments of the distribution of $D_t(\mathcal{S}_{t,i})$. Using these moments, for each sigma point $\mathcal{S}_{t,i}$, we use the UT algorithm in line 4 to generate N' new sigma points and weights denoted as $\mathcal{S}^j_{t,i}$ and $w^j_{t,i}$, $j \in \{1, .., N'\}$, $N' = 2n + 1$ for UT, as illustrated in Fig. 5.1. The weight of new sigma points $\mathcal{S}^j_{t,i}$ is obtained equal to $w^j_{t,i}\, w_{t,i}$, and the resulting NN' new sigma points are returned. The next theorem shows that this scheme correctly represents the true distribution of x_{t+1} up to mean and covariance.

Algorithm 6 expand_sigma_points

Require: $\mathcal{S}, w, D()$ ▷ Collection of sigma points and their weights, closed-loop dynamics distribution

Initialize $\mathcal{S}_{new} = \{\}, w_{new} = \{\}$
for i in size($\mathcal{S}$) **do**
    mean, cov = moments($D_c(\mathcal{S}_i)$)
    $s', w' =$ generate_UT_points(mean, cov, skewness, kurtosis)
    $\mathcal{S}_{new}$.append(s')
    w_{new}.append($w' * w_i$)
end for
Return $\mathcal{S}_{new}, w_{new}$

Theorem 6. Suppose the sigma points $\mathcal{S}_{t,i}, w_{t,i}, i \in \{1, 2, .., N\}$ have sample moments equal to moments of random variable $x_t \in \mathbb{R}^n$. For each sigma point $\mathcal{S}_{t,i}$, consider the N' new sigma points and weights denoted by $\mathcal{S}^j_{t,i}, w^j_{t,i}$, where $j \in \{1, 2., , .N'\}$ that have the same sample moments equal to distribution $D(\mathcal{S}_{t,i})$. Then, (1) the set of points $\mathcal{S}^j_{t,i}$ with corresponding weights $w_{t,i}w^j_{t,i}$ have a sample mean and covariance equal to those of the true distribution of x_{t+1}, $\mathbb{E}[x_{t+1}] = \mathbb{E}_s[x_{t+1}]$. (2) Furthermore, suppose there exists N sigma points $\mathcal{S}_{t+1,i}, w_{t+1,i}, i \in \{1, 2, .., N\}$ whose sample moments are equal to moments of x_{t+1}. Then, the sigma points $\mathcal{S}_{t+1,i}, w_{t+1,i}$ and $\mathcal{S}^j_{t,i}, w^j_{t,i}$ have the same sample mean and covariance.

Preliminary Knowledge for Proof At time t, consider the random variable $\hat{x}_t$, which has to be propagated through a random function $D(\cdot)$ to obtain the random variable $\hat{x}_{t+1}$. The mean of $\hat{x}_{t+1}$ is: $\mathbb{E}[\hat{x}_{t+1}] = \mathbb{E}[D(\hat{x}_t)] = \mathbb{E}\big[\mathbb{E}[D(\hat{x}_t)|\hat{x}_t]\big]$, where we use the law of iterated expectation [**?**] in the second equality, while by the law of total variance [**?**] we have: $\Sigma[\hat{x}_{t+1}] = \mathbb{E}\big[\Sigma[\hat{x}_{t+1}|\hat{x}_t]\big] + \Sigma\big[\mathbb{E}[\hat{x}_{t+1}|\hat{x}_t]\big] = \mathbb{E}[\Sigma(D(\hat{x}_t)|\hat{x}_t)] + \Sigma[\mu(D(\hat{x}_t)|\hat{x}_t)]$.

Proof. For brevity, we remove the explicit dependence on t for sigma points and weights. For example, $\mathcal{S}_i$ stands for $\mathcal{S}_{t,i}$. The sample expectation defined by sigma points $\mathcal{S}^j_i$ and weights $w_i w^j_i$ is given as $\mathbb{E}_s[\hat{x}_{t+1}] = \sum_i^N \sum_j^{N'} w_i w^j_i \mathcal{S}^j_i = \sum_i^N w_i \sum_j^{N'} w^j_i \mathcal{S}^j_i$. Since $\mathcal{S}^j_i, w^j_i$ are sigma points for $D(\mathcal{S}_i)$ and since we have assumed that they have sample moments equal to true moments of $D(\mathcal{S}_i)$, we have

$$\mathbb{E}[D(\mathcal{S}_i)] = \sum_j^{N'} w^j_i \mathcal{S}^j_i, \tag{5.8}$$

$$\Sigma[D(\mathcal{S}_i)] = \sum_j^{N'} w^j_i (\mathcal{S}^j_i - \mathbb{E}[D(\mathcal{S}_i)])(\mathcal{S}^j_i - \mathbb{E}[D(\mathcal{S}_i)])^T. \tag{5.9}$$

Then, we have

$$\mathbb{E}_s[\hat{x}_{t+1}] = \sum_i^N w_i \sum_j^{N'} w_i^j \mathcal{S}_i^j \tag{5.10}$$

$$\overset{(5.8)}{=} \sum_i^N w_i \, \mathbb{E}[D(\mathcal{S}_i)] = \sum_i^N w_i \mathbb{E}[D(\hat{x}_t)|\hat{x}_t = \mathcal{S}_i]$$

$$= \mathbb{E}[\mathbb{E}[D(\hat{x}_t)|\hat{x}_t = \mathcal{S}_i]] = \mathbb{E}[\hat{x}_{t+1}], \tag{5.11}$$

$$\Sigma_s[\hat{x}_{t+1}] = \sum_i \sum_j w_i w_i^j (\mathcal{S}_i^j - \mathbb{E}[\hat{x}_{t+1}])(\mathcal{S}_i^j - \mathbb{E}[\hat{x}_{t+1}])^T$$

$$= \sum_i w_i \left(\sum_j w_i^j (\mathcal{S}_i^j - \mathbb{E}[\hat{x}_{t+1}])(\mathcal{S}_i^j - \mathbb{E}[\hat{x}_{t+1}])^T \right) \tag{5.12}$$

$$= \sum_i^N w_i \sum_j^{N'} w_i^j \left((\mathcal{S}_i^j - \mathbb{E}[D(\mathcal{S}_i)])(\mathcal{S}_i^j - \mathbb{E}[D(\mathcal{S}_i)])^T \right)$$

$$+ \sum_i^N w_i \sum_j^{N'} w_i^j (\mathbb{E}(D(\mathcal{S}_i)) - \mathbb{E}[\hat{x}_{t+1}])(\mathbb{E}(D(\mathcal{S}_i)) - \mathbb{E}[\hat{x}_{t+1}])^T$$

$$+ 2 \sum_i^N w_i \sum_j^{N'} w_i^j \left(\mathcal{S}_i^j - \mathbb{E}[D(\mathcal{S}_i)]\right)(\mathbb{E}[D(\mathcal{S}_i)] - \mathbb{E}[\hat{x}_{t+1}])^T \tag{5.13}$$

$$\overset{(5.9)}{=} \sum_i^N w_i \Sigma[D(\mathcal{S}_i)]$$

$$+ \sum_i^N w_i \left(\mathbb{E}[D(\mathcal{S}_i)] - \mathbb{E}[\hat{x}_{t+1}]\right)\left(\mathbb{E}[D(\mathcal{S}_i)] - \mathbb{E}[\hat{x}_{t+1}]\right)^T + 0$$

$$= \mathbb{E}[\Sigma[D(\hat{x}_t)\big|\hat{x}_t = \mathcal{S}_i]] + \Sigma[\mathbb{E}[D(\hat{x}_t)\big|\hat{x}_t = \mathcal{S}_i]] = \Sigma[\hat{x}_{t+1}],$$

where the second expression in (5.13) equals $\Sigma[\mathbb{E}[D(x_t)]]$ and third expression is 0 because $\sum_j w_i^j \mathbb{E}[D(\mathcal{S}_i)] = \mathbb{E}[D(\mathcal{S}_i)] \left(\sum_j w_i^j\right) = \mathbb{E}[D(\mathcal{S}_i)]$ as $\sum_j w_i^j = 1$ and therefore $\sum_j w_i^j \mathcal{S}_i^j = \mathbb{E}[D(\mathcal{S}_i)]$.

The second argument now trivially follows since both the set of sigma points, $\mathcal{S}_{t+1,i}$, $w_{t+1,i}$ (given) and $\mathcal{S}_{t,i}^j$, $w_{t,i}^j$ (from first argument), are known to have sample mean and covariance equal to that of x_{t+1} and therefore equal to each other. The second argument forms the basis for the compression layer which is presented in the next section.

□

Note that Theorem 6 does not assume a specific distribution, but relies on the assumption

that $\mathcal{S}_i^j, w_i^j$ represents the true distribution. In this regard, our approach is not limited to GPs with Gaussian kernel as in PILCO [75] and other studies that use PILCO-inspired state prediction [190, 115, 222, 168].

Compression Layer Line 1 of Algorithm 5 first computes the sample moments of NN' sigma points $\mathcal{S}_{t,i}^j$ and weights $w_{t,i}w_{t,i}^j$. Then using these moments, we generate new N sigma points in line 2 using UT.

5.4 Application to Online Stochastic Optimization - Constraint Aware Gradient Descent

In this section, we propose a gradient-descent approach for lines 11 and 12 of Algorithm 4 for solving Problem **P.1**. This section is an extension of online tuning framework proposed in Section 4.4 to stochastic dynamics.

Without loss of generality, we divide the online policy update design into two cases and denote the prediction time with τ.

Case 1: At time t, suppose that the policy $\pi(x_{t+\tau}, \theta)$ given out of (5.5) satisfies the constraints (5.5b) over the time steps $\tau \in \mathcal{T}$, but is suboptimal w.r.t. the performance objective. For $R = \sum_{\tau=0}^{H} r(x_{t+\tau}, \pi(x_{t+\tau}, \theta))$, the standard RL update $\theta^+ = \theta - \beta(\nabla_\theta \mathbb{E}[R])$, where $\theta^+ \in \mathbb{R}^\kappa$ is the updated parameter at time t, $\beta \in [0, 1]$ is the learning rate, and $\nabla_\theta \mathbb{E}[R] \in \mathbb{R}^\kappa$ is the gradient of the expectation of the reward, does not ensure that the policy $\pi(x_{t+\tau}; \theta^+)$ continues to satisfy (5.5b) for all $\tau \in \mathcal{T}$. therefore, similar to the procedure presented in Section 4.4, following constrained gradient descent update rule is used:

$$\theta^+ = \theta + \beta d_\theta, \tag{5.14a}$$

$$d_\theta = \arg\max_d \; d^T \nabla_\theta \mathbb{E}[R] = \arg\min_d \; -d^T \nabla_\theta \mathbb{E}[R] \tag{5.14b}$$

$$\text{s.t.} \quad P[c_i(x_{t+\tau}) + d^T \nabla_\theta c_i(x_{t+\tau}) \geq 0] \geq 1 - \epsilon, \tag{5.14c}$$

$$\forall \tau \in \mathcal{T}, \forall i \in \mathcal{K},$$

where θ^+ is the updated parameter at time t, $\beta \in [0, 1]$ is the learning rate, $\epsilon \in (0, 1]$ is the risk tolerance, $\nabla_\theta \mathbb{E}[R]$, is the gradient (wrt θ) of the expectation of the reward, i.e., a deterministic value, but $\nabla_\theta c_i(x_{t+\tau})$ and $c_i(x_{t+\tau})$ are functions of the random variable $x_{t+\tau}$. This update rule forces the new parameter θ^+ to satisfy the constraints that θ satisfied up to first-order approximation about the current trajectory.

We note that (5.14c) is a probabilistic, or chance, constraint. Chance-constrained optimization problems are in general NP-hard, and only some cases are known to have analytical

solutions. When the chance-constrained problem is intractable, several approximations (safe, convex, tractable approximations) of the original problem are usually considered, such as in [158, 26]. We articulate two such approaches to approximate constraints (5.5b) and (5.14c) using existing approaches in Appendix B.1 and B.2. Notably, the approximate constraint (B.7) captures, consistently with one's intuition, the statistics of the state prediction, and the behavior of the constraint functions around the estimates of the state and the parameter vectors. In our ensuing though, for the sake of computational efficiency (reasons mentioned in Appendix B.1), we employ the approximation in (B.9) from Appendix B.2.

Remark 16. While SQP provides aforementioned guarantees only upto first-order approximation about current trajectory, Feasible SQP [225] algorithms, although more computationally expensive, can also be used to provide guarantees up to higher orders. Choosing a small β helps in ensuring that the first-order approximation is valid. Also, note that (5.14) may give unbounded values of d_θ. We therefore clip the solution of (5.14) to a small value in addition to multiplying it with learning rate β. One could also formulate a quadratic objective in (5.14b) to promote smaller values of d (resulting in a QP) but that would increase the optimization solve time, which is undesirable in practice as we wish to apply (5.14) as frequently as possible.

Case 2: At time t, suppose that the policy $\pi(x_{t+\tau}, \theta)$ given out of (5.5) satisfies the constraints (5.5b) over the time steps $\tau \in \{0, 1, \ldots, H-1\}$, but not at the $\tau = H$ time step when predicting the future states $x_{t+\tau}, \tau \in \mathcal{T}$. In this case, we relax the j-th constraint function at the H time step so that $c_j(x_{t+H}, u_{t+H}) = -\delta_j,\ j \in \mathcal{J}$, where $\mathcal{J}$ is the set of constraints that could not be satisfied in the original constraint (5.5b) and $\delta_J > 0$ are called slack variables and are minimized through an objective function that minimizes δ_J^2 instead of the original objective (5.5a). $-\delta_J$ is also referred to as the infeasibility margin. Let $\delta_H = [\delta_{\mathcal{J}_1}, \delta_{\mathcal{J}_2} ...,]$ be the vector of all slack variables. The update rule is then designed to be:

$$\theta^+ = \theta + \beta d_\theta, \quad \text{where} \tag{5.15a}$$

$$d_\theta = \arg\max_d \ -d^T \nabla_\theta \mathbb{E}[{\delta_H}^T \delta_H] \tag{5.15b}$$

$$\text{s.t. } P[c_i(x_{t+\tau}) + d^T \nabla_\theta c_i(x_{t+\tau}) \geq 0] \geq 1 - \epsilon, \tag{5.15c}$$

$$\forall \tau \in \{0, 1, .., H-1\}, \forall i \in \mathcal{K}$$

$$P[c_j(x_{t+H}) + d^T \nabla_\theta c_j(x_{t+H}) \geq 0] \geq 1 - \epsilon, \tag{5.15d}$$

$$.\forall j \in \mathcal{K} \setminus \mathcal{J}$$

The above update rule minimizes the slack, and as a result the infeasibility margin, while enforcing that the constraints that were already satisfied continue to be satisfied up to first-

order approximation. The approximation of the probabilistic constraints with robust ones can be done as in the earlier case.

5.4.1 Simulation Results

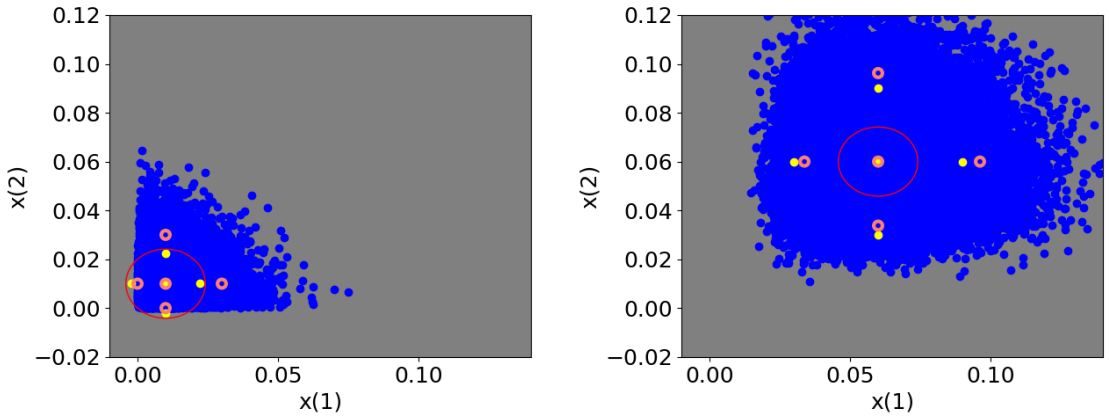
We present a number of case studies to show the efficacy of the proposed method. The first case study demonstrates the accuracy and sample efficiency of the Expansion-Compression UT compared to state-of-the art methods on uncertainty propagation. The second case study demonstrates the application of FORESEE in offline trajectory optimization, while the third and fourth show the applicability of the method for online controller tuning. The code and videos for our simulations can be found at https://github.com/hardikparwana/FORESEE. The videos can also be found on thesis website https://github.com/hardikparwana/PhD-Thesis.

5.4.2 Benchmark: Accuracy of State Prediction

We compare the accuracy of our state distribution prediction scheme to other methods. To evaluate the performance, we perform Monte Carlo with 500, 5000, and 50,000 particles, as well as successive Gaussian approximation for updating the mean and covariance of the distribution. Note that PILCO's application is limited to GPs with Gaussian kernel and therefore cannot be applied directly to any of the presented case studies in this section. Therefore, given the dynamics, $x_{t+1} \sim D(x_t)$, the successive Gaussian approximation based on updating the mean and covariance as follows $\mathbb{E}[x_{t+1}] = \mathbb{E}[D(x_t)] \sim \mu_D(\mathbb{E}[x_t]), \Sigma[x_{t+1}] \sim \Sigma_D(\mathbb{E}[x_t])$ where μ_D, Σ_D are the mean and covariance functions of distribution D. We also perform two versions of our proposed algorithm: one with successive expansion (E) operations only, leading to a surge in the number of sigma points, and another with successive expansion-compression (EC) operations, which keep the number of sigma points constant. The performance of the compression-expansion layer is evaluated with two different UT algorithms: the standard UT (Algorithm 2) and the generalized UT [82]. The free parameter k in Algorithm 2 is chosen to be 1, which is known to be optimal for Gaussian distributions.

In the first case study, we propagate states through a non-Gaussian dynamics function, specifically, the gamma distribution Γ. Given the shape and scale parameters, k and β respectively, the first four moments of the distribution are: $\mu_\Gamma = k\beta, \Sigma_\Gamma = k\beta^2, S_\Gamma = \frac{2}{\sqrt{k}}, K_\Gamma = \frac{6}{k} + 3$. We design the following dynamics function

$$x_{t+1} = x_t + \Delta t\ \Gamma(k\beta_t, k\beta_t^2, \frac{2}{\sqrt{k}}, \frac{6}{k} + 3), \tag{5.16}$$



(a) Horizon = 1, UT with EC

(b) Horizon = 6, UT with EC

Figure 5.2: The predicted state distribution under different methods for dynamics given by (5.16). Blue samples correspond to 50,000 MC particles. Yellow and salmon-colored samples are the sigma points generated using successive Expansion-Compression (EC) operations. Yellow samples employ UT (Alg. 2) and salmon samples employ Generalized UT [82] in its expansion and compression operation. Red ellipse represents the 95% confidence ellipse of the distribution obtained by successive Gaussian approximation.

where we choose $k = 2, \beta_t = 0.01x_t^2 + 0.1$, and $\Delta t = 0.05$ and gamma distribution is applied elementwise. The predicted state distributions and moments as a function of the prediction horizon are shown in Figs. 5.2 and 5.3. Monte Carlo (MC) simulations are run 20 times and sample sizes of 500 and 5000 are observed to have increasing variance in their estimates with the horizon. Successive Gaussian approximation performs well in estimating the mean but performs poorly in estimating higher-order moments. EC with GenUT is observed to better represent all the moments compared to UT as expected owing to its capability to retain more higher-order moments.

In the second case study, we consider the dynamics:

$$x_{t+1} \sim D(x_t), \tag{5.17}$$

$$D(x) = N(g(x), 0.01 + \text{diag}(g(x)^2)), \tag{5.18}$$

$$g(x) = a \begin{bmatrix} \cos(x(1)) + 0.01 \\ \sin(x(2)) + 0.01 \end{bmatrix}, \tag{5.19}$$

where $x = \begin{bmatrix} x(1) & x(2) \end{bmatrix}^T$ and $a \in \mathbb{R}$. Starting at time $t = 1$ with state $x_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, we predict the value of the state x_3 at time $t = 3$. Fig. 5.4 depicts the predicted states obtained under different methods (proposed, Monte Carlo, and successive Gaussian approximation) for $a = 20$. Fig. 5.5 shows the corresponding predicted moments for different values for $a = 20$,

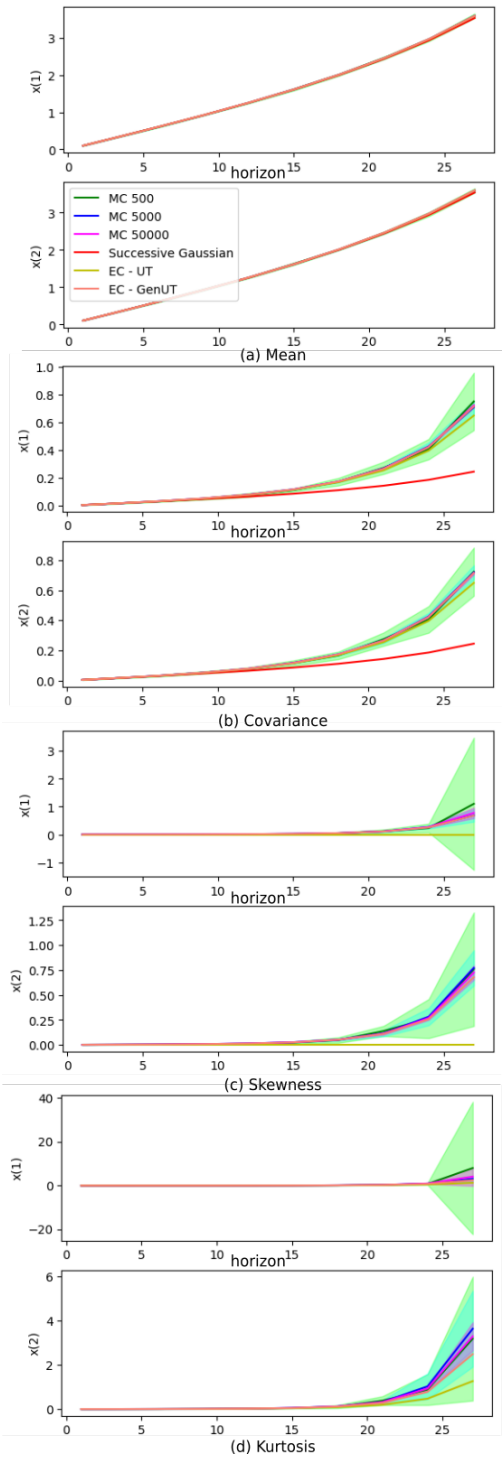


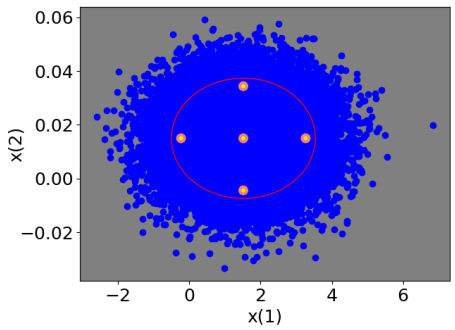
Figure 5.3: First four moments with different methods: Monte Carlo (MC), successive Gaussian, Expansion-Compression (EC) with UT, and with Generalized UT (GenUT). The successive Gaussian approximation is only shown for the first two moments as it is identically zero and constant for skewness and kurtosis. The shaded areas show two standard deviation variations in MC runs.

and Fig. 5.6 shows the computation times for a naive Python implementation. Monte Carlo (MC) simulations are run 20 times and sample sizes of 500 and 5000 are observed to have high variance in their estimates. Successive Gaussian approximation performs poorly in all scenarios, and its covariance identically tends to zero. It also exhibits oscillatory behavior in Fig. 5.5(a). On the other hand, in agreement with previous works [82], we note that UT-based methods lead to a similar accuracy as that of MC with a large number of particles, but at a much lower computational cost. Also, as evident from Fig. 5.4 and 5.6, UT with expansion only is non-scalable, as within a horizon of 6, the number of sigma points is much larger than the number of MC particles, and hence is no longer computationally cheap. On the contrary, the proposed UT with the Expansion-Compression (EC) layer keeps the number of sigma points fixed across a time step and is computationally the most efficient method. Finally, generalized UT [82] leads to more accurate estimates of moments than the standard UT, which is also unable to convey information on the asymmetry of the resulting distributions. Generalized UT on the other hand chooses sigma points asymmetrically to preserve the diagonal elements of skewness and kurtosis moment matrices of the sigma points in compression operation and is therefore better represents the true distribution.

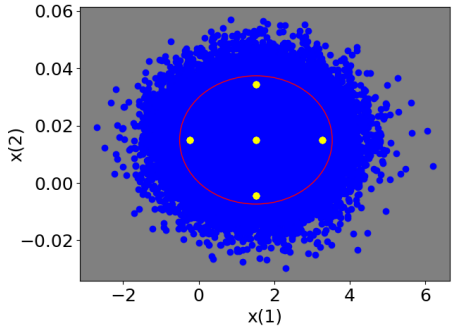
Note that Theorem 6 relied on the assumption that sigma points accurately represent all the moments of the distribution. In practice, we hypothesize that the degree to which this assumption is violated is dependent on the complexity of the distribution under consideration and the specific UT method employed. For example, we observe here that most methods perform well with gamma distribution. However, all methods including ours experience performance degradation (EC with UT degrades more than GenUT) in the second case study as seen in Fig. 5.5, where (5.19) introduces oscillatory dynamics function that causes drastic changes to the state distribution at every time step. A thorough theoretical analysis of error induced by UT algorithms and their propagation through multiple time steps is out of the scope of this chapter and will be addressed in future work.

5.4.3 Application to Stochastic Trajectory Optimization

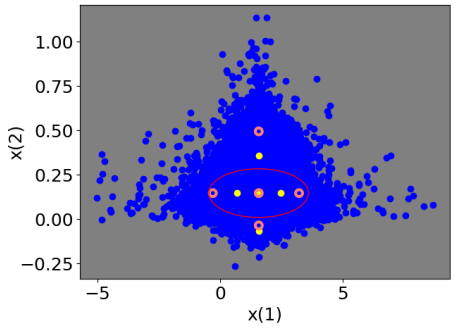
In this section, we show an application of state prediction to stochastic trajectory optimization. Consider a dynamic unicycle model with state $x_t = [p_{xt}\ p_{yt}\ \psi_t\ v_t]^T$ and the following



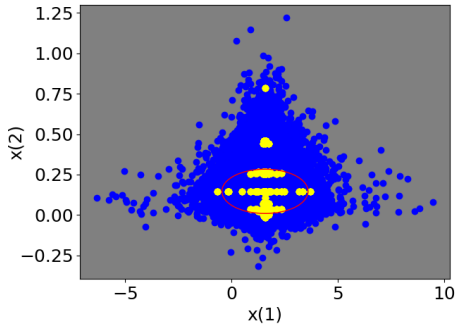
(a) Horizon = 1, UT with EC



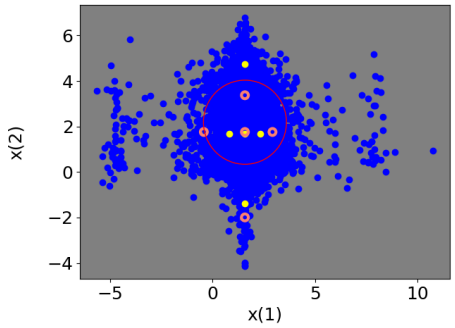
(b) Horizon = 1, UT with E only



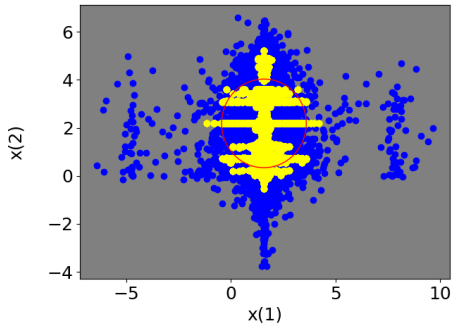
(c) Horizon = 3, UT with EC



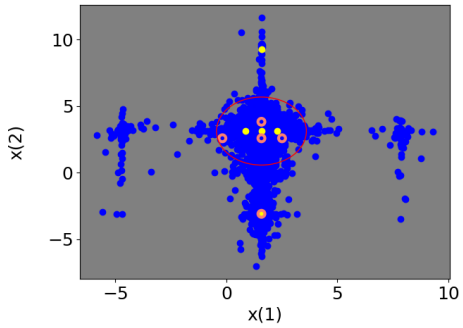
(d) Horizon = 3, UT with E only



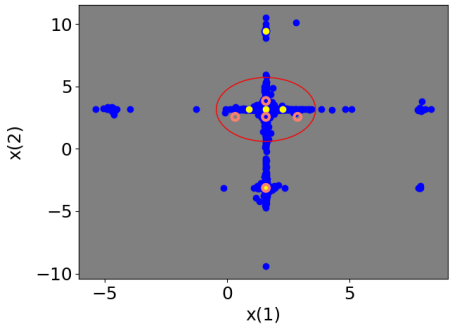
(e) Horizon = 6, UT with EC



(f) Horizon = 6, UT with E only



(g) Horizon = 8, UT with EC



(h) Horizon = 10, UT with EC

Figure 5.4: The predicted state distribution under (5.19) for different dynamics. The colors of samples follow the notation in Fig. 5.2. E stands for expansion.

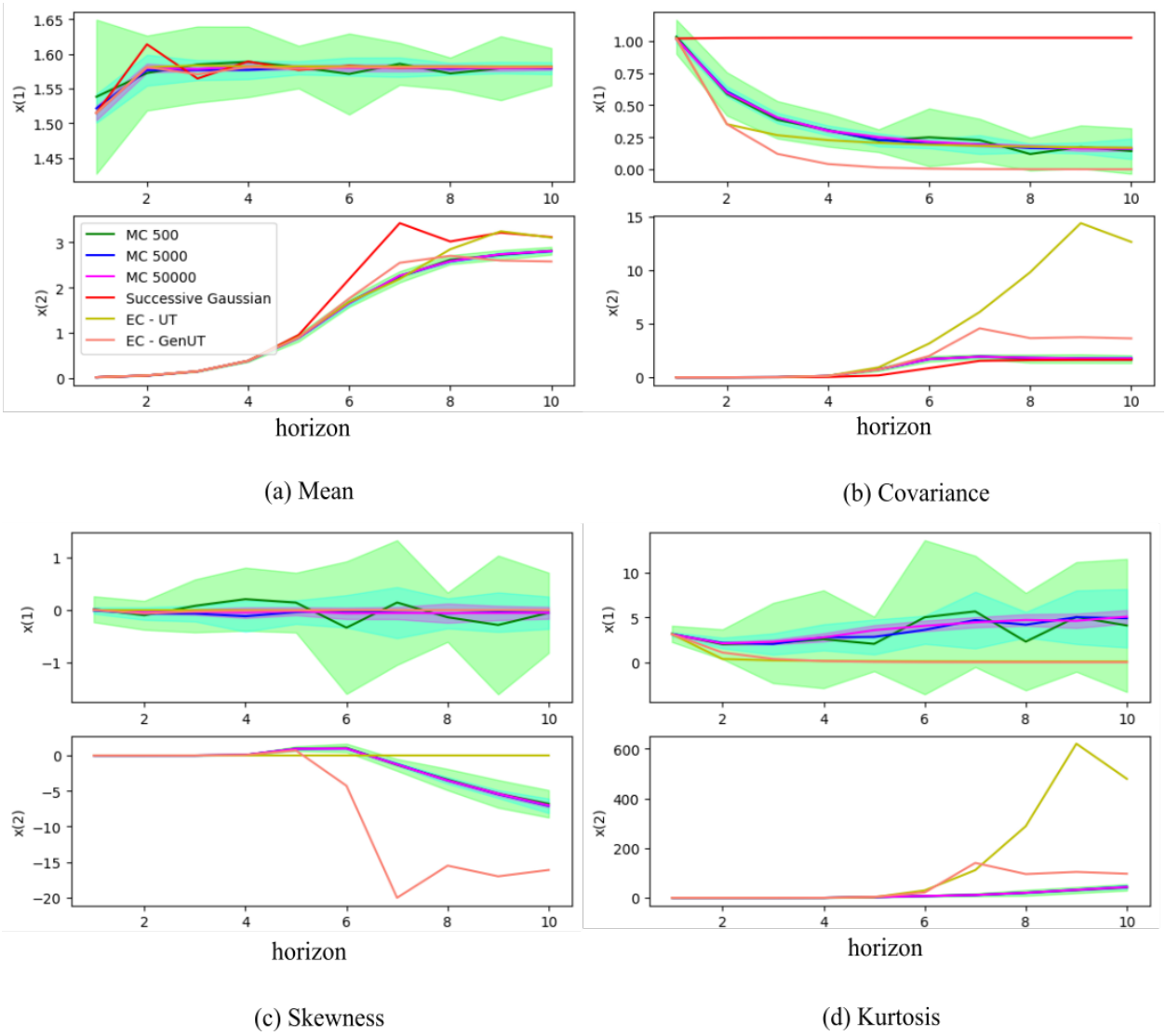


Figure 5.5: First four moments for different methods as a function of the horizon under (5.19). Colors follow the same notation as in Fig. 5.3.

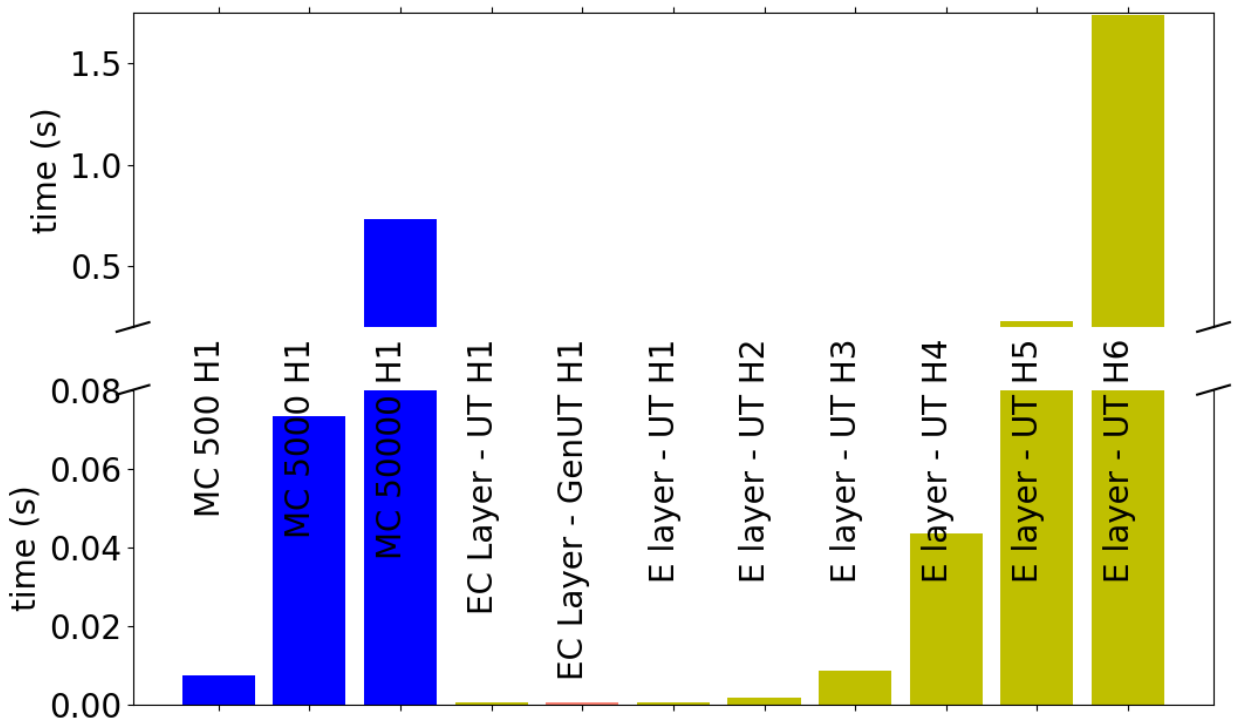


Figure 5.6: Time taken by each of methods in a naive Python implementation (without any optimizations): Monte Carlo (MC), Expansion-Compression (EC), and Expansion-only (E). All operations are performed on a single core of CPU. H value denotes the horizon. The time per step for MC and EC is fixed. The time taken per step by E Layer increases with horizon as the number of particles increases.

uncertain dynamics

$$f(x_t) = x_t + \begin{bmatrix} v_t \cos\psi_t \\ v_t \sin\psi_t \\ \omega_t \\ a_t \end{bmatrix} \Delta t, \tag{5.20}$$

$$x_{t+1} \sim N\left(\frac{3}{2} f(x_t), \frac{\text{diag}[(f(x_t) - x_t)^2]}{10}\right), \tag{5.21}$$

where p_{xt}, p_{yt} are the position coordinates along the axes of an inertial frame, ψ_t is the heading angle, v_t is the speed, $u_t = [\omega_t \; a_t]^T$ is the control input consisting of angular velocity ω_t and acceleration a_t, and $N(\mu, \Sigma)$ is the normal distribution with mean μ and covariance matrix Σ. The objective is to navigate the robot to its goal location while avoiding collision with a circular obstacle as shown in Fig.5.7. Denote the distance to the obstacle at time t as d_t, and the obstacle radius as d_{min}, and consider two cases: the first enforces collision avoidance, a scalar constraint, in expectation, that is $\mathbb{E}[{d_t}^2 - d_{min}^2] \geq 0$, and the second enforces collision avoidance with 95% confidence interval (CI) defined as $\mathbb{E}[{d_t}^2 - d_{min}^2] - 2\sqrt{\Sigma[{d_t}^2 - d_{min}^2]} \geq 0$. The objective function in (5.4) is chosen as $r(x_t, u_t) = 10 \left(\left\| \begin{bmatrix} p_{xt} \\ p_{yt} \end{bmatrix} - G \right\| \right)^2 + \|u_t\|^2$, where G is the goal location. We solve (5.4) for time step $\Delta t = 0.05s$ in (5.20) and horizon $H = 40$ using IPOPT solver. Fig. 5.7(left) and 5.7(right) show the resulting trajectory for both cases. To validate our method, we also perform Monte Carlo (MC) with 50,000 particles (to simulate the stochastic system dynamics), and propagate the state forward with the control input designed using expansion-compression layers (our method). We observe that our method provides good tracking of the mean of the state while under-approximating its covariance.

5.4.4 Application to Online Controller Tuning for Leader Follower

To showcase the online tuning of control policies, we use the proposed framework as a tuning mechanism for the control policy that results from a Quadratic Program (QP) subject to Control Barrier Function (CBF) conditions for safety [175, 15], termed thereafter the CBF-QP controller, since its parameters are difficult to generalize to new environments.

A continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ is a CBF for the set $\mathcal{D} = \{x \in \mathcal{X} \mid h(x) \geq 0\}$ under the deterministic dynamics $x_{t+1} = F(x_t, u_t),\ F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, $t \in \mathbb{Z}^+$,

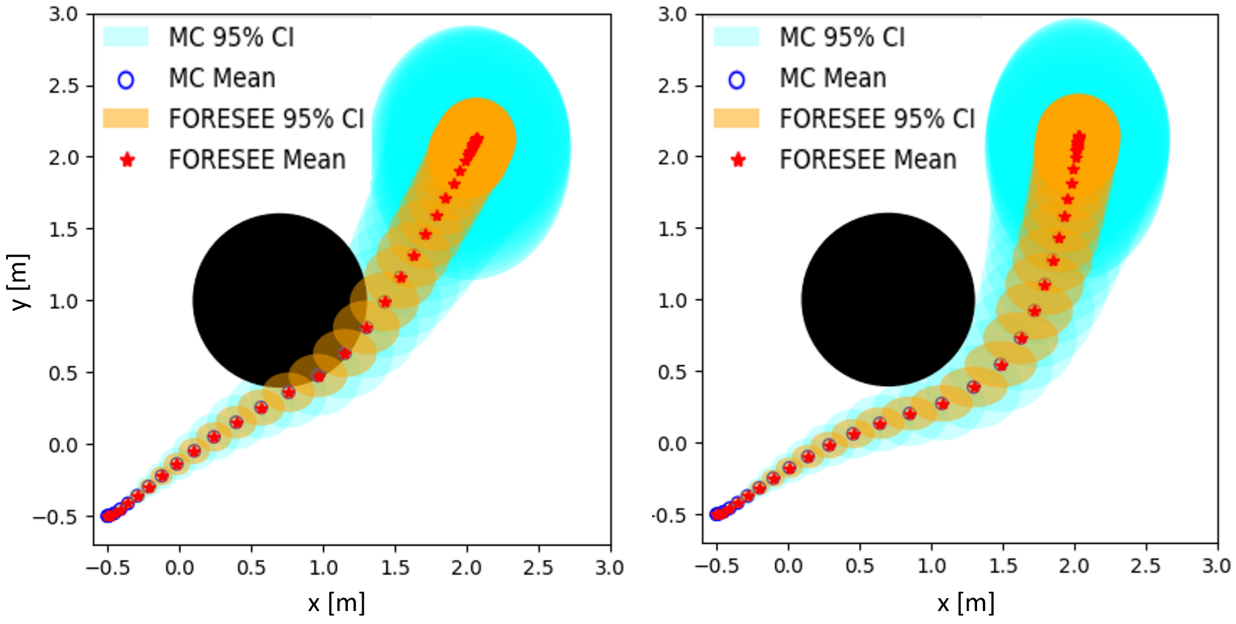


Figure 5.7: The predicted trajectories of dynamic unicycle modeled robot. The start location is at (-0.5,-0.5) and the goal is at (2,2). The orange and blue ovals represent the 95% confidence interval (w.r.t mean and covariance) of the sigma point and Monte Carlo states respectively. On the left is the case where constraint satisfaction in enforced in expectation. On the right is the case where constraint satisfaction is enforced with 95% confidence interval.

if there exists an extended class-$\mathcal{K}_\infty$ function ν such that $\forall t \in \mathbb{Z}^+$:

$$\sup_{u_t \in \mathcal{U}} [h(x_{t+1}) - h(x_t)] \geq -\nu(h(x_t)),\ \forall x_t \in \mathcal{X}. \tag{5.22}$$

For simplicity, we use $\nu(x) = \alpha x, \alpha \in \mathbb{R}^+$, where α is a parameter that can be tuned. Given K barrier functions $h_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in \mathcal{K}$, we can define corresponding CBF derivative conditions (5.22) for each of the barriers h_i, each of which will include a parameter α_i. Let also the desired state be encoded in terms of a Control Lyapunov Function (CLF) $V : \mathbb{R}^n \to \mathbb{R}$, and consider the policy $\pi_{QP}(x_t; \alpha_0, \alpha_1, ...\alpha_K)$ obtained by the CBF-CLF-QP:

$$\min_{u_t, \delta} \quad J(u_t) = (u_t - u_t^d)^T P (u_t - u_t^d) + Q\delta^2 \tag{5.23a}$$

$$\text{s.t.} \quad V(x_{t+1}) \leq (1 - \alpha_0) V(x_t) + \delta \tag{5.23b}$$

$$h_i(x_{t+1}) \geq (1 - \alpha_i) h_i(x_t)\ , \forall i \in \{1, .., K\}$$

$$Au_t \leq b, \tag{5.23c}$$

where $P \in \mathbb{R}^{m \times m}, Q \in \mathbb{R}^+$ are the weight factors, u_t^d is the nominal (or desired) control input at time t, $\delta \in \mathbb{R}$ is a slack variable used to relax the CLF condition, and A, b specify the control inputs in the set $\mathcal{U}$. We will refer to $\theta = \begin{bmatrix} \alpha_0 & \dots & \alpha_K \end{bmatrix}^T$ as the parameter vector to be optimized over a time horizon so that the cumulative reward is maximized while the system states are propagated over this time horizon with our computationally-efficient UT method.

To illustrate the efficacy of the proposed state prediction and online policy optimization method, we consider two mobile robots moving in a leader-follower fashion, where the leader is modeled as a single integrator with dynamics as follows: $\dot{p}_{x,l} = u_1,\ \dot{p}_{y,l} = u_2$, where $p_{x,l}, p_{y,l}$ are the position coordinates of the leader with respect to an inertial reference frame, and u_1, u_2 are the velocity components along the axes of reference frame, whereas the follower is modeled with unicycle dynamics: $\dot{p}_{x,f} = u\cos\phi,\ \dot{p}_{y,f} = u\sin\phi,\ \dot{\phi} = \omega$, where $p_{x,f}, p_{y,f}$ are the position coordinates of the follower with respect to the same inertial reference frame, ϕ is the angle between the x-axis of the body-fixed reference frame of the follower and the inertial frame, and u, ω are the linear and angular velocity of the follower expressed in its body-fixed reference frame.

The follower's objective is to maintain the leader visible in its limited camera Field-of-View (FoV), and preferably at the center of FoV. The follower's controller is given as a QP subject to three constraints as follows: $h_1 = s^2 - s_{min}^2 \geq 0$,$h_2 = s_{max}^2 - s^2 \geq 0$, $h_3 = b - \cos(\gamma) \geq 0$, where $s = \sqrt{(p_{x,l} - p_{x,f})^2 + (p_{y,l} - p_{y,f})^2}$ is the Euclidean distance between the follower and the leader, s_{min} is the minimum allowed distance for collision avoidance, s_{max} is the maximum

allowed distance for accurate detection using the onboard camera, b is the bearing vector from the follower to the leader, and γ is the FoV angle of the follower's camera. The CBF-CLF-QP (5.23) uses the CBF conditions of the aforementioned constraints, the Lyapunov function $V(s) = \left(s - \frac{s_{min}+s_{max}}{2}\right)^2$, and the input constraints $|u| \leq 2.5, |\omega| \leq 4$.

The leader is moving under the stochastic dynamics:

$$\dot{p}_l = N(\begin{bmatrix} 1.25u_x \\ 1.25u_y \end{bmatrix}, \begin{bmatrix} 0.25({u_x}^2 + u_y^2) & 0 \\ 0 & 0.25(u_x^2 + {u_y}^2) \end{bmatrix}),$$
$$u_x = 2, u_y = 3\cos 4\pi t. \tag{5.24}$$

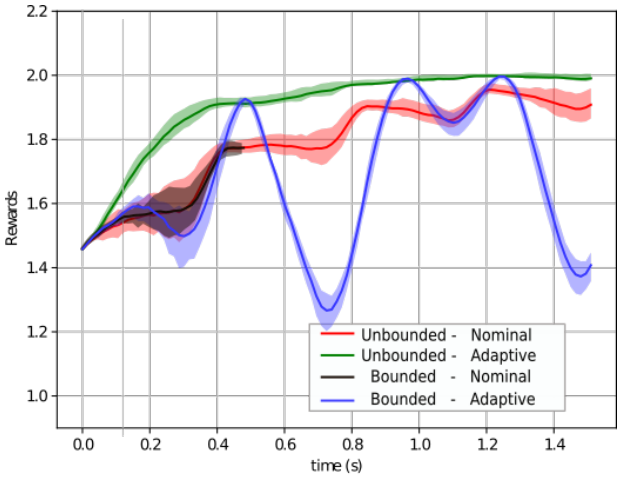
Since the leader's state is 2-dimensional, we use 5 sigma points in the Unscented Transform in Algorithm 2 in FORESEE to represent the leader's state. Fig. 5.8 shows the results for unbounded and bounded control inputs, each with fixed parameter values (no adaptation) and with adaptive parameter values wherein we use the proposed online parameter tuning framework (Algorithm 1). Since the leader's true dynamics is stochastic, the results are reported over 30 trials.

It is noteworthy that when the control input is unbounded, one can choose a set of fixed parameters that are able to satisfy constraints at all times (Fig. 5.8b), but with lower reward compared to the adaptive case (Fig. 5.8a); this means that the follower is not able to keep the leader centered in the FoV.

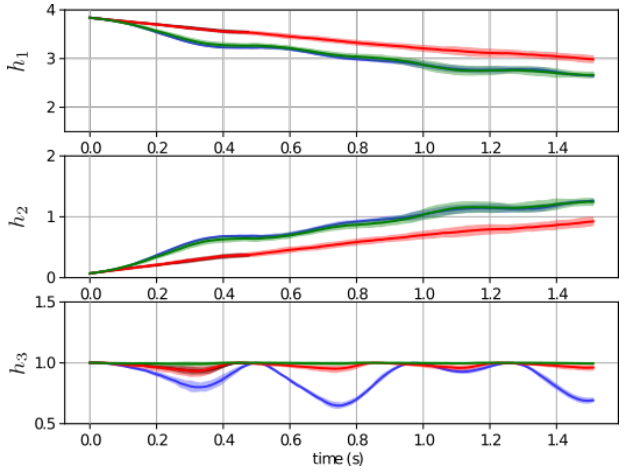
When control input bounds are enforced, it is noteworthy that in the fixed parameter case, the QP becomes infeasible after some time and the simulation fails, see the shorter black trajectories in Fig. 5.8. The update rule proposed in (5.15) succeeds in maintaining feasibility and adapts the parameters so that the constraints are satisfied (Fig. 5.8b). Interestingly, in the presence of input bounds, the response of the optimized policy reflects the fact that the follower does not have enough control bandwidth to focus on performance, i.e., keep the leader at the center of the FoV, and only ensures constraint satisfaction by turning minimally to keep the leader inside the FoV (i.e., angular velocity ω remains close to zero); this can be verified from the corresponding control inputs in Fig. 5.8c.

5.4.5 Benchmark Comparison

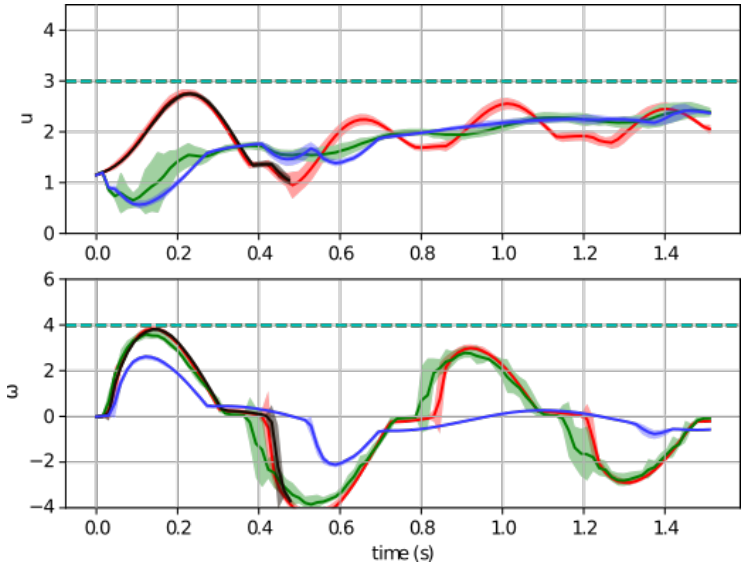
We compare our Algorithm 4 for online controller tuning to the GP-MPC approach in [106] when applied on a 2D quadrotor model subjected to state and input constraints. We implement our framework in safe-control-gym [264] that also provides implementation of [106].



(a) Variation of reward with time.



(b) Evolution of h_1 (minimum distance), h_2 (maximum distance) and h_3 (maximum angle w.r.t camera axis) with time t. Constraint maintenance corresponds to $h_i(t) > 0, i \in \{1, 2, 3\}$



(c) Evolution of follower's control inputs with time.

Figure 5.8: Responses over time for the system reward, barrier functions and the control input for the cases of bounded and unbounded control input and with nominal (fixed) and adaptive controller parameters.

The quadrotor dynamics is:

$$\ddot{p}_x = \frac{(T_1+T_2)\cos\theta}{m} \quad \ddot{p}_z = \frac{(T_1+T_2)\sin\theta}{m} - g \quad \ddot{\theta} = \frac{(T_2-T_1)d}{I_y},$$

where $p = \begin{bmatrix} p_x & p_z \end{bmatrix}^T$ are the horizontal and vertical positions of the quadrotor, θ is the pitch angle, T_1, T_2 are the thrusts of left and right motors, and the control input is denoted as $u = \begin{bmatrix} T_1 & T_2 \end{bmatrix}^T$. The quadrotor is subject to the state constraint $Ap \leq b$ with $A = [1 \ -1], b = 2$. We use the following nominal position controller:

$$a_d = -k_x(p - p_d) - k_v(\dot{p} - \dot{p}_d) - k_{rx} \frac{A}{\|A\|} \tanh\left(\frac{1}{b - Ap}\right), \tag{5.25}$$

where $T = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} (a_d + mg)$, $\theta_d = \tan^{-1}(T_x, T_z)$, $M = I_y(-k_R(\theta - \theta_d) - k_{Rv}\dot{\theta})$, $T_2 = 0.5(T + M/L)$, $T_1 = 0.5(T - M/L)$, and $a_d, p_d, \dot{p}_d$ are the desired accelerations, positions and velocities. k_x, k_v, k_{rx} are gains to be tuned for stabilization performance and to satisfy the safety constraint at all times, and k_R, k_{Rv} are constant gains to stabilize the quadrotor attitude control. Note that since the safe-control-gym benchmark runs the controller code at 10Hz, the resulting discretization errors are expected to degrade the performance of the continuous-time controller in (5.25). We perform all simulations on a Ubuntu 22.04 laptop running on an i9-13905H processor and no GPU acceleration is employed. We use the same trained Gaussian Process that GP-MPC uses to predict future states in our predictive framework. The prediction horizon for all the algorithms is 10. In Fig. 5.9(left), we show the results for GP-MPC and linear MPC controllers from safe-control-gym for a quadrotor stabilization task. The robot starts at $x = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ and has to stabilize at $x = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$. The simulation is run for 30 steps with a controller time step of 0.1 sec. The control input (thrust forces) are bounded, $0 \leq T_i \leq 0.2N, i \in \{1, 2\}$. The means and covariances of the system states for times $[t, t + 10]$ predicted by MPC at time t are shown and visualized by 95% confidence interval ellipses. The colormap denotes the simulation time, ranging from violet ($t = 0$) to red ($t = 3$).

In contrast to MPC, we deploy FORESEE with online constrained gradient descent to tune the gains of nominal controller (5.25). The gains are initiated as $k_x = 0.07, k_v = 0.02, k_{rx} = 0, k_R = 60, k_{Rv} = 10$ and mass and inertia are taken to be the true values from safe-control-gym. Fig. 5.9(middle) shows the performance of the nominal controller for fixed values of the gains. Under this choice of gains, the system violates safety constraints and exhibits poor stabilization performance. However, with our online prediction-based tuning, we can tune parameters for constraint satisfaction and better stabilization as seen in Fig. 5.9(right).

safe-control-gym uses IPOPT solver for solving the nonlinear MPC optimization. It also employs some approximations proposed in [106] to impose linear chance constraints in the MPC to improve optimization solve time. The first run takes 19.8 seconds for IPOPT to find a solution. The subsequent runs benefit from setting the initial guess based on the solution of the last solved MPC (a standard trick in receding horizon implementations) and take between 2.7 to 8.5 seconds to find a solution. Note that on average, despite using the efficient IPOPT solver written in C++, the time to solve a single MPC (even with a good initial guess) is still more than the real-world duration of 3 seconds that this quadrotor maneuver is supposed to take. FORESEE is written in Python using the JAX library and JAX's autograd features are used to compute the required gradients. Each update in (5.14) and (5.15) takes 0.02 seconds. The very first run takes 118 iterations of (5.15) to find a feasible solution but after that, we limit the number of iterations in receding horizon implementation to 10 to finish the simulation within 3 seconds. Writing a more efficient C++ code will be performed in future work.

5.4.6 Experiment Results

In this section, we demonstrate FORESEE's capability to auto-tune the controller of a real quadrotor platform in the presence of wind disturbances. The experiment setup consists of a self-assembled drone using a Pixhawk 6C flight controller board [1] using custom PX4 firmware [148] (based on PX4 release v1.14) that employs a geometric controller [127] translation and attitude dynamics stabilization. An axial fan, employed in [71], with a diameter of 36 in ($\approx$ 0.91m) is used to simulate wind disturbances. Inspired by [167], we model the effect due to wind disturbance to be dependent on the position, velocity, attitude (represented as a quaternion), and commanded translation acceleration (given by (5.27)) of the quadrotor. We keep the gains of the attitude controller the same and fixed throughout all the experimental trials and only attempt to tune the gains of the position controller.

The quadrotor dynamics, for position $p \in \mathbb{R}^3$, is written as:

$$m\ddot{p} = mg + F_T + F_{wind}, \tag{5.26}$$

where m is the mass of the quadrotor, g is the acceleration due to gravity, and $F_T \in \mathbb{R}^3$ is the total force exerted by the propellers. Finally, F_{wind} is the force exerted by wind and is designed as specified before. The geometric controller stabilizes the translation dynamics

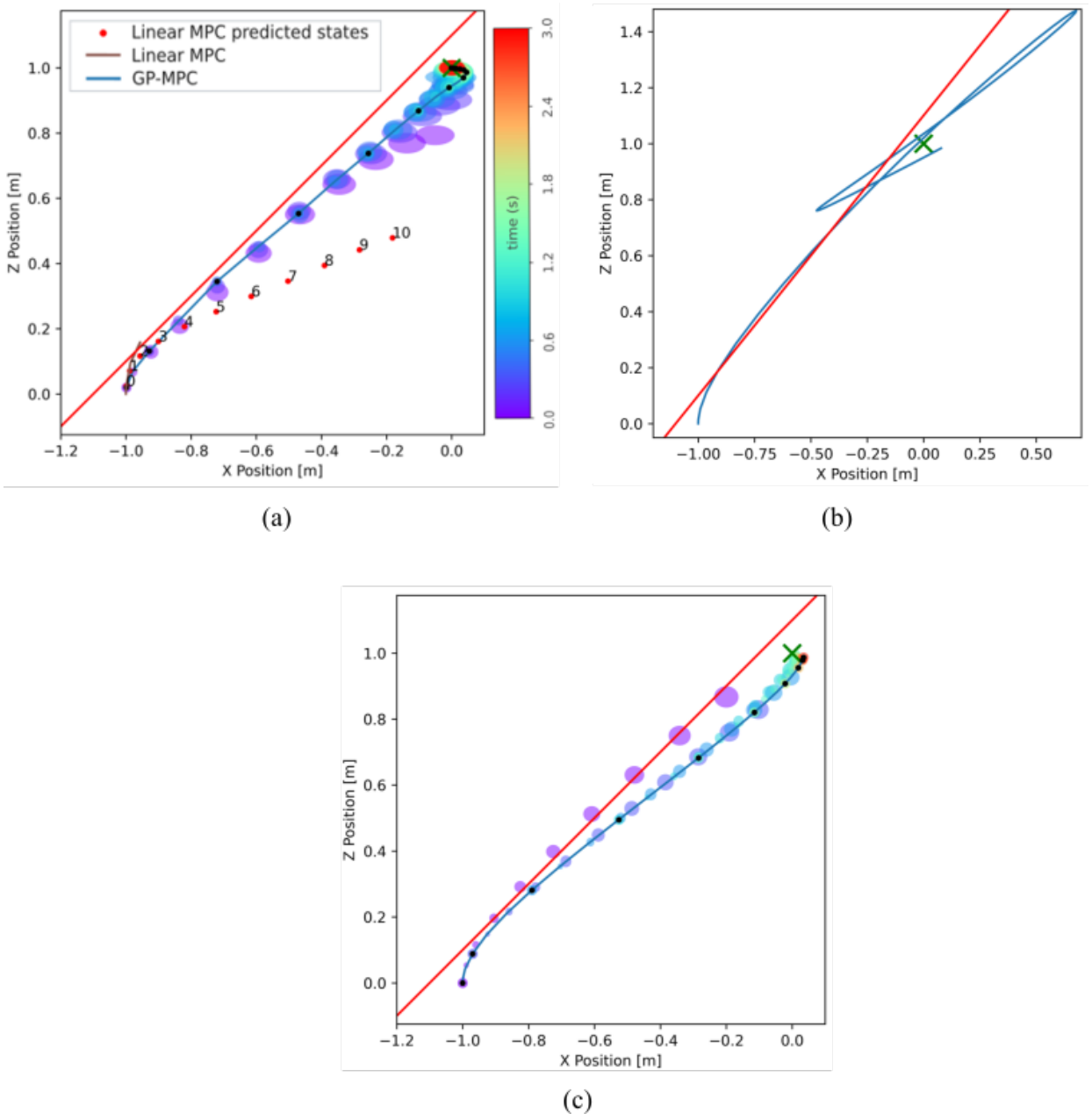


Figure 5.9: Quadrotor stabilization results for benchmarking in safe-control-gym. The ellipses in (a) and (c) show the 95% confidence interval Gaussian representing the states predicted by GP-MPC. The black dots are the (chosen) states from which predictions were made. The predicted ellipses originating from these dots are colored according to the colormap corresponding to the timestamp of the black dot.

and enforces collision avoidance with an obstacle using the following dynamics:

$$F_T = F_T^{max} \tanh(\beta(-k_x e_x - k_v e_v + m\ddot{p}_{ref} - mg + F_{obs})),$$
$$F_{obs} = k_{obs} \frac{p - p_o}{||p - p_o||} \tanh(\frac{1}{||p - p_o|| - r}), \tag{5.27}$$

where p_{ref} is the reference trajectory, $e_x = p - p_{ref}, e_v = \dot{p} - \dot{p}_{ref}$ are position and velocity errors, tanh is used to clip the designed input to the maximum allowed thrust F_T^{max}, and p_o, r are the position and radius of the obstacle.

We first learn the wind disturbances and use the learned function F_{wind} to perform offline as well as online optimization for the gains $k_x, k_v, k_{obs} > 0$. The procedure to collect data and learn F_{wind} by fitting a Gaussian Process (GP) is detailed in Appendix B.3.

All computations, offline as well as online, are performed on a computer with i9-12000K CPU and NVIDIA RTX 3080 GPU, and the commands are sent to the quadrotor through PX4's ROS2 communication bridge. Since GP's prediction time is proportional to size of training dataset, multiple predictions over a time horizon are not suitable for real-time implementation on a CPU. We therefore run the GPJax library on GPU to accelerate predictions enabling FORESEE to run at about 20 Hz. Implementation of FORESEE on a GPU onboard the quadrotor is also possible, however, we choose to run it on an offboard computer because of the ease of software setup owing to better library support.

5.4.6.1 Offline Optimization

In this section, we perform offline policy optimization for trajectory tracking ($k_{obs} = 0$) by using the learned GP model in (5.5) for a large horizon. We choose the stage-wise cost function as $r(x, u) = Q_1\|p - p_{ref}\| + Q_2\|\dot{p} - \dot{p}_{ref}\|$, where $Q_1, Q_2 > 0$ are scalar weights and $x = [p^T, \dot{p}^T]^T$ is the quadrotor state. We optimize policy (5.27) for the model (5.26) and $Q_1 = 1, Q_2 = 0.2$. We show results for two parameter values, specifically (1) default parameters $k_x = 7, k_v = 4$ that are tuned manually for flight in the absence of the fan, and (2) optimized parameters $k_x = 17.92, k_v = 6.34$ for a circular reference trajectory and $k_x = 17.9, k_v = 6.0$ for "figure 8" reference trajectory that are found by solving (5.4) for the learned GP model with horizon of 300 steps and time step of 0.05 s. As seen in Fig. 5.10, and as expected, the performance of default (un-optimized) parameters degrades significantly when the fan is turned on, whereas the optimized parameters exhibit improved performance.

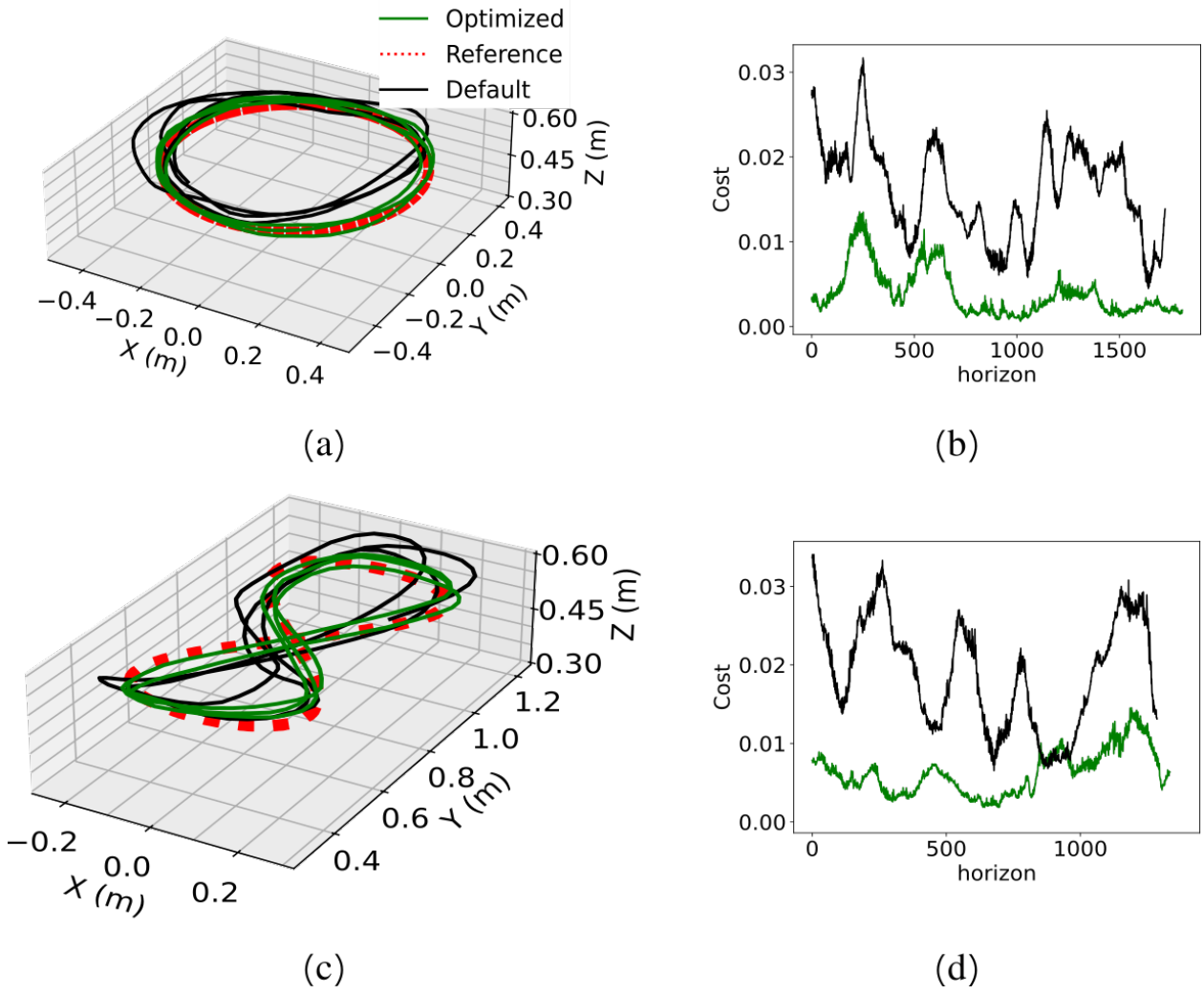


Figure 5.10: Path and cost plots for circular and figure-8 reference trajectories.

5.4.6.2 Online optimization

In this case, we consider a cylindrical obstacle of radius 0.4 m located at $p_{ox} = -0.4, p_{oy} = 0.1$ ($p_{oz} = p_z$ instantaneously for infinite cylinder) and tune k_{obs} along with k_x, k_v to improve tracking performance while ensuring collision avoidance. We choose initial parameters as $k_x = 7, k_v = 4, k_{obs} = 4$ and use FORESEE to tune the gains online. We consider the constraint $c(x_\tau) = \mathbb{E}[\|p_\tau - p_o\| - r] - 2\sqrt{\Sigma[\|p_\tau - p_o\| - r]} \geq 0$, and perform the following simplification of (5.14c) to reduce the number of constraints that need to be enforced in the LP (5.14). We combine the constraints across all time steps into a single constraint through the following equation $\bar{c} = \sum_{\tau=0}^{H} \min(c(x_\tau), 0) \geq 0$, where we choose horizon $H = 50$ and implement FORESEE in receding horizon fashion thereby continuously changing the parameters k_x, k_v, k_{obs} as the robot moves. Although min is a non-differentiable operator whenever $c(x_\tau) = 0$, due to discrete-time implementation and numerical precision, we never encounter this case in practice. If a constraint violation is predicted, we run 4 steps of gradient descent (GD) at every time step otherwise we run 1 step of proposed GD at each time step (at about 20 Hz). Note that the problem is nonlinear, we do not guarantee the timely convergence of parameters and therefore number of GD steps at each time and learning rates are subject to tuning. In practice, we choose a small horizon for faster implementation and running multiple steps of GD at every instant.

Fig. 5.11(a) shows the quadrotor paths. The default parameter $k_{obs} = 4$, being too conservative, prohibits the quadrotor from approaching the reference trajectory. With online adaptation, the gains are adapted to allow the quadrotor to approach the reference without violating the collision constraint. We allow quadrotor to loop multiple times and show the resulting path in Fig. 5.11(b). With each loop, the quadrotor gets closer to the reference while avoiding the obstacle. We also observe a violation of constraint in one of the loops. We show this result to reiterate that violations are subject to tuning of learning rate and horizon and finally model mismatch between predicted dynamics and reality. In Fig. 5.11(c) we show the trajectories that result from performing a computer simulation using the learned GP model of quadrotor. Comparing the two figures (b and c) we note that they are similar but not the same. We suspect that the trained GP model is close to actual dynamics but is still error-prone being a data-driven method. Videos for all the experiments can be found at https://github.com/hardikparwana/FORESEE and on the thesis website https://github.com/hardikparwana/PhD-Thesis.

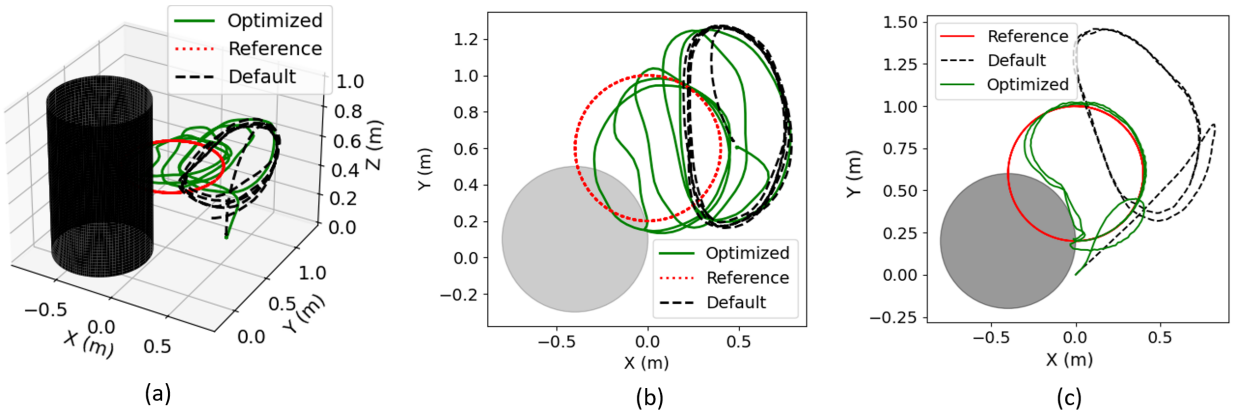


Figure 5.11: Online policy optimization for trajectory tracking and collision avoidance of a quadrotor. Default parameter video: https://www.youtube.com/watch?v=D59R0nZvPWE, Proposed method video: https://www.youtube.com/watch?v=vsBRkm59rAA. In the video, the hoop represents the cylindrical obstacle and the reference is not visualized.

5.5 Application to MPPI for Hybrid Systems

In this section, we present an application of proposed Expansion-Compression layers to Model Predictive Path Integral Control (MPPI), a popular sampling-based algorithm to solve unconstrained optimal control problems. We focus on a special case study - stochastic hybrid dynamical systems that present a unique challenge of predicting distributions of future states subject to a state-dependent dynamics switching function. A variant of Model Predictive Path Integral Control (MPPI) is proposed to plan kinodynamic paths for such systems. Monte Carlo may be inaccurate when few samples are chosen to predict future states under state-dependent disturbances. We employ the Expansion-Compression layers from 5.3.1 to capture stochasticity in the states as well as the state-dependent switching surfaces. This is in contrast to previous works that perform switching based only on the mean of predicted states. We focus our motion planning application on the navigation of a mobile robot in the presence of dynamically moving agents whose responses are based on sensor-constrained attention zones. We evaluate our framework on a simulated mobile robot and show faster convergence to a goal without collisions when the robot exploits the hybrid human dynamics versus when it does not.

This section, although advances the state-of-the-art in application for planning paths for stochastic hybrid dynamical systems, is not related to any other part of the thesis and may be skipped by the reader.

5.5.1 Motivation

Hybrid dynamical systems [141, 97], in which the dynamics function changes in response to an *event*, appear in several robotic applications. Examples include walking legged robots [121], contact-rich quadrotor navigation [120], and mobile robot navigation on different terrains [80]. In practice, due to disturbances or *a priori* unknown dynamics of the robot or the environment, these systems are often modeled as stochastic hybrid systems [12]. While there is extensive research on path planning and control for hybrid dynamical systems [47, 131], we focus here on model predictive approaches to address these challenges.

Our motivation for studying these systems arises from navigation challenges, and particularly two interrelated, but often independently addressed, challenges. First, the motion of agents, whether human or non-human, is challenging due to noisy observations of past states and the often unpredictable behavior of these agents (e.g., sudden changes in direction). Planners must account for this inherent uncertainty, which may be captured by modeling agent motion as a stochastic hybrid system. Second, agents may abruptly alter their behavior upon perceiving new entities in their environment, including robots, obstacles, or other agents. In this work, we model agents' dynamic modes as hybrid dynamical systems, where each mode represents varying levels of cooperation or responsiveness to a robot. Mode switching is triggered, for example, when the robot enters an agent's attention zone, such as a human's field of view or an aquatic or aerial agent's sensory range.

The integration of estimators into path planning systems is crucial for navigating environments that involve diverse agents. Previous works [108, 234] have proposed estimators that provide deterministic predictions, typically trained on datasets specific to constrained contexts like pedestrian traffic in mall corridors. More recent research has shifted towards developing adaptable behavioral models for estimation [221, 28, 151] or inferring deviations from expected motion based on observations [224]. These approaches consider factors such as social attention, which prioritizes nearby agents [234], or context-specific parameters like environmental constraints and sensory limitations [221]. In this work, we focus on modeling cooperative behaviors where agent interactions are influenced by sensory zones, such as a field of view or sensory range. These factors play a critical role in enhancing situational awareness and guiding responses to dynamic environmental changes [151].

Since exact analytical expressions for propagating probability distributions through nonlinear functions are generally not available, several approximate methods have been developed to solve stochastic MPC. These methods include Monte Carlo simulations [13], successive Gaussian approximation [75], and the Unscented Transform [174]. In the domain of hybrid systems, MPC problems are typically formulated as mixed-integer optimizations [80, 25], which can become computationally prohibitive, particularly when the system

dynamics are nonlinear.

Since solving the hybrid MPC exactly may not be feasible for real-time implementation, we adopt Model Predictive Path Integral (MPPI) Control [247], a sampling-based, gradient-free method for solving unconstrained MPC problems in real-time. MPPI typically converts hard constraints into soft constraints by incorporating them into the objective function. Variants of MPPI have been developed to address uncertainty in the initial state [153] and disturbances in system dynamics [248, 89]. However, [153] focuses on deterministic dynamics, and [248, 89] address only state-independent noise. To our knowledge, applying MPPI to systems with state-dependent disturbances remains a challenge due to the lack of analytical methods for propagating future states under such disturbances. In this work, we leverage recent advances in uncertainty propagation techniques [174] for stochastic MPC within the MPPI framework.

The problem of switching dynamics under uncertain states is similar to work on belief tree planning for motion planning under estimator uncertainty [46]. These works incorporate state estimation uncertainty in the planning module to probabilistically bound the safety violation metric as the robot navigates with an onboard localization module. The correction step of the Extended Kalman filter (EKF) that is used for state estimation is triggered only when an obstacle is within the sensing region of the robot. However, [46] implements the switch between these different modes based only on the mean of the state estimate, thereby making switching a deterministic phenomenon despite the state estimates being stochastic.

In this work, we propose to use a particle-based uncertainty propagation scheme based on Section 5.3.1 in MPPI. **Our contributions are twofold:** First, by utilizing weighted particles, we map the stochasticity of states to the stochasticity of mode switching, moving beyond deterministic switching based only on predicted state means. Second, we extend MPPI to systems with arbitrary state-dependent disturbances. Both of these contributions enable us to define a risk-aware framework for path planning with respect to static and dynamic obstacles for a hybrid dynamical system. Our experimental results demonstrate improved cost and safety performance by accounting for the hybrid nature of non-ego agent dynamics, compared to planning approaches that neglect cooperative behaviors or assume constant velocity models [237, 33].

5.5.2 Related Work

The work most similar to ours in terms of solution approach is [49], wherein different terrains induce different dynamics for a mobile robot. An unknown traction parameter in the dynamics is modeled with a terrain-dependent probability distribution, thereby making this

also an instance of stochastic hybrid dynamics. They propose a Monte Carlo (MC) scheme motivated by [242] to evaluate the cost of executing a control input sequence. From an application standpoint, [49] applied MPPI when the uncertainty in dynamics is dependent on static terrain features whereas uncertainty in our case stems from active interaction between dynamically moving ego and non-ego agents. However, the main difference is that the method presented in [49] uses Monte Carlo whereas we use UT to propagate state distributions across nonlinear functions. Whereas MC can require a prohibitively large number of samples [95], for similar performance in accuracy UT is shown to be more computationally efficient in contrast [82, 22, 174].

5.5.3 Notation and Problem Formulation

The set of real numbers is denoted as $\mathbb{R}$. For $x \in \mathbb{R}, y \in \mathbb{R}^n$, $|x|$ denotes the absolute value of x and $\|y\|$ denotes the L_2 norm of y. The time derivative of x is denoted by $\dot{x}$ and $\frac{\partial F}{\partial x}$ denotes the gradient of a differentiable function $F : \mathbb{R}^n \to \mathbb{R}$ with respect to $x \in \mathbb{R}^n$.

Consider a robot with state $x_r \in \mathbb{R}^{n_r}$, control input $u_r \in \mathbb{R}^{m_r}$, and dynamics

$$x_{r,t+1} = f_r(x_{r,t}, u_{r,t}). \tag{5.28}$$

Each agent p is modeled as a stochastic hybrid (i.e., multimodal) dynamical system with continuous state $x_p \in \mathbb{R}^{n_p}$ and N operating modes q_j, for $j \in \{1, 2, \ldots, N\}$. Each mode corresponds to different agent dynamics. For example, a robot may exhibit different behavior toward another agent when it enters its limited sensing region.

When the agent p is in mode q_j, then their motion is governed by stochastic dynamics $D_{q_j} : \mathbb{R}^{n_p} \times \mathbb{R}^{n_r} \to \mathbb{R}^{n_p}$, which depend both on the state of the agent and the robot, that is,

$$x_{p,t+1} \sim D_{q_j}(x_{p,t}, x_{r,t}). \tag{5.29}$$

A mode q_j is active in a region

$$\mathcal{A}_j = \{(x_p, x_r) \in \mathbb{R}^{n_p} \times \mathbb{R}^{n_r} \mid \wedge_{k \in \bar{K}}\, g_{j,k}(x_p, x_r) \leq 0\},$$

where $g_{j,k} : \mathbb{R}^{n_p} \times \mathbb{R}^{n_r} \to \mathbb{R}, k \in \bar{K} = \{1, \ldots, K\}$ are K activation functions for mode q_j. We assume that the regions $\{\mathcal{A}_j\}_{j \leq N}$ form a partition of the space $\mathbb{R}^{n_p} \times \mathbb{R}^{n_r}$.

Remark 17. To simplify the presentation and notation, we have introduced a hybrid dynamical model where the activation functions partition the agent-robot state space. However, the motion planning framework that we introduce in this chapter is more general and can

handle general hybrid dynamical systems [152].

In our framework, we allow for multiple dynamic agents. When necessary, we will use the subscript i, e.g., $x_{p,i}$, D_{i,q_j}, etc, to highlight the fact that multiple agents are present – potentially under different stochastic dynamics.

Example 5 (Simple motion model with attention radius)**.** Consider the following attention regions corresponding to two operating modes q_1 (non-attentive) and q_2 (attentive) of an external agent with corresponding regions

$$\mathcal{A}_1 = \{(x_p, x_r) \mid ||x_p - x_r|| > d_s\},$$
$$\mathcal{A}_2 = \{(x_p, x_r) \mid ||x_p - x_r|| \leq d_s\},$$

where d_s is the sensing area of the agent. The stochastic dynamics in each mode are:

$$D_{q_1}(x_{p,t}, x_{r,t}) = x_{p,t} + u_g(x_{p,t}) + w_{1,t}(x_{p,t}),$$
$$D_{q_2}(x_{p,t}, x_{r,t}) = x_{p,t} + u_g(x_{p,t}) + u_a(x_{p,t}, x_{r,t}) + w_{2,t}(x_{p,t}),$$

where $x_{p,t}$ and $x_{r,t}$ are the positions of the agent and the robot, respectively, at time t, u_g models the agents's attraction toward their goal, u_a models the agent's avoidance of the ego-robot, and $w_{j,t}$ are appropriate noise terms that may depend on the state of the agent in general.

In general, the agents dynamics can also be written as

$$x_{p,t+1} \sim \sum_{j=1}^{N} \delta_j D_{q_j}(x_{p,t}, x_{r,t}), \tag{5.30}$$

where δ_j are binary variables and $\delta_j = 1$ if mode q_j is active,

$$\delta_j = \begin{cases} 1 & \text{if } (x_{p,t}, x_{r,t}) \in \mathcal{A}_j \\ 0 & \text{otherwise .} \end{cases} \tag{5.31}$$

5.5.3.1 Safety Constraints

Our objective is to navigate a mobile robot in an environment with the presence of static obstacles, such as walls, and stochastic, dynamic agents. We model the requirement for collision avoidance between the robot x_r and each agent x_p and between the robot x_r and

each static obstacle x_o using distance functions:

$$h_p : \mathbb{R}^{n_r} \times \mathbb{R}^{n_p} \to \mathbb{R}^+ \tag{5.32a}$$

$$h_o : \mathbb{R}^{n_r} \times \mathbb{R}^{n_o} \to \mathbb{R}^+ \tag{5.32b}$$

We further assume that h_p and h_o are defined in such a way that given a safety threshold $\varepsilon > 0$, the sets $\{(x_r, x_p) \,|\, h_p(x_r, x_p) < \varepsilon\}$ and $\{(x_r, x_o) \,|\, h_o(x_r, x_o) < \varepsilon\}$ do not contain disconnected sets of measure zero. This means that the safety problem is well-defined.

5.5.3.2 Problem definition

The question we are answering in this section is whether a robot can improve its motion planning performance, e.g., time to reach its goal or energy consumption, if the robot utilizes a model of the agents' awareness of their environment. At the same time, we would like to ensure that safety is not compromised. In other words, we aim to improve the system performance while keeping the risk of collisions bounded by a desired threshold ε.

Formally, we aim to solve the following path planning problem:

Problem 3. Given the system comprised of (5.28) and (5.29), a safety threshold $\varepsilon > 0$, a risk threshold $\epsilon \in [0, 1]$, and a stage-wise cost $R : \mathbb{R}^{n_r} \times \mathbb{R}^{m_r} \to \mathbb{R}$, plan a control input trajectory $v = \{u_{r,1}, u_{r,2}, ..., u_{r,T}\}$ for a horizon T that solves the following optimization problem

$$v = \underset{\{u_{r,1},\ldots,u_{r,T}\}}{\arg\min} \quad \sum_{\tau=0}^{T} R(x_{r,\tau}, u_{r,\tau}) \tag{5.33a}$$

$$\text{s.t.} \quad x_{r,\tau+1} = f_r(x_{r,\tau}, u_{r,\tau}) \tag{5.33b}$$

$$x_{p,\tau+1} \sim \sum_{i=1}^{N} \delta_{j,\tau} D_{q_j}(x_{p,\tau}, x_{r,\tau}) \tag{5.33c}$$

$$\delta_{j,\tau} = 1 \text{ if } (x_{p,\tau}, x_{r,\tau}) \in \mathcal{A}_j \tag{5.33d}$$

$$P\left[\bigwedge_{k=\tau}^{T} h_p(x_{r,k}, x_{p,k}) \geq \varepsilon\right] \geq 1 - \epsilon \tag{5.33e}$$

$$P\left[\bigwedge_{k=\tau}^{T} h_o(x_{r,k}, x_{o,k}) \geq \varepsilon\right] \geq 1 - \epsilon. \tag{5.33f}$$

5.5.4 Preliminaries

Instead of solving Problem 3 exactly, we solve it approximately by imposing hard constraints as penalties in the objective function and using MPPI to find a locally optimal path. In this

section, we briefly describe the MPPI algorithm as well as the Expansion-Compression UT algorithms that help approximate Eq. (5.33c).

5.5.4.1 Model Predictive Path Integral Control

MPPI finds a locally optimal solution to an unconstrained optimal control problem by sampling trajectories around a nominal trajectory and assigning weights to the trajectories based on the cost incurred. The final trajectory results from a weighted summation of all the sampled trajectories.

Consider a nonlinear system with state $x_t \in \mathbb{R}^n$ and control input $u_t \in \mathbb{R}^m$ that follows the following discrete-time dynamics

$$x_{t+1} = F(x_t, u_t). \tag{5.34}$$

For a time horizon H, consider the state trajectory $\mathbf{x} = [x_t^T, ..., x_{t+H}^T]^T$, mean control input sequence $\mathbf{v} = [v_t^T, .., v_{t+H}^T]^T, v_\tau \in \mathbb{R}^m$ and injected Gaussian noise $\mathbf{g} = [g_t^T, .., g_{t+H}^T]^T$ where $g_\tau \sim N(0, \Sigma_g)$. Let the disturbed control input sequence be $\mathbf{u} = \mathbf{v} + \boldsymbol{g}$.

The MPPI algorithm solves the following problem

$$\min_{\mathbf{v}} \quad J(\mathbf{v}) = \mathbb{E}\left[\Phi(x_{t+H}) + \sum_{\tau=t}^{t+H-1}\left(Q(x_\tau) + \frac{\lambda}{2} v_\tau^T \Sigma_g^{-1} v_\tau\right)\right] \tag{5.35a}$$

$$\text{s.t.} \quad x_{\tau+1} = F(x_\tau, v_\tau + g_\tau) \tag{5.35b}$$

$$g \sim \mathcal{N}(0, \Sigma_g). \tag{5.35c}$$

Below we provide a summary of the MPPI algorithm and refer the reader to [247] for a more theoretical insight into its derivation and optimality properties. MPPI samples M trajectories where $M \gg 1$, and each trajectory is affected by a different disturbance sequence ϵ. We will add a superscript $()^m$ to denote quantities corresponding to m^{th} sampled trajectory. The cost S_m of each trajectory $m \in \{1, .., M\}$ is evaluated based on (5.35a) as follows

$$S_m = \Phi(x_{t+H}^m) \sum_{\tau=t}^{t+H-1} Q(x_\tau^m) + \gamma (v_\tau^m)^T \Sigma_g^{-1} (v_\tau^m + g_\tau^m),$$

where $\gamma \in [0, \lambda]$. The weight w_m for m^{th} trajectory is determined as

$$w_m = \exp\left(-\frac{1}{\lambda}(S_m - \beta)\right), \tag{5.36}$$

where $\beta = \min_{m\in\{1,..,M\}} S_m$. The optimal control sequence is then computed as

$$\mathbf{v}^+ = \sum_{m=1}^{M} w_m \mathbf{u}^m / \sum_{m=1}^{M} w_m. \tag{5.37}$$

In a receding horizon implementation, the control input is chosen as $u_t = \mathbf{v}_0^+$ and the mean sequence $\mathbf{u}$ for the next iteration is set to be $\mathbf{v} = \mathbf{v}^+$.

5.5.4.2 Expansion-Compression Unscented Transform (ECUT)

Algorithm 7 applies the expansion and compression operations from Section 5.3.1 to propagate state by one time step. Note that the expand_sigma_point function takes the dynamics function as an input and evaluates the switching function on each sigma point separately.

Algorithm 7 Expansion-Compression UT

Require: $(\mathcal{S}_\tau, w_\tau)$, F ▷ current state sigma points, Stochastic Dynamics function

1: $\mathcal{S}_{next}, w_{next} \leftarrow$ expand_sigma_point$(F, \mathcal{S}_\tau, w_\tau)$
2: $\mathbb{E}_s = \sum_{i=1}^{NN_F} w_{next,i}\mathcal{S}_{next,i}$
3: $\Sigma_s = \sum_{i=1}^{N} w_{next,i}(\mathcal{S}_{next,i} - \mathbb{E}_s)(\mathcal{S}_{next,i} - \mathbb{E}_s)^T$
4: $\mathcal{S}_{\tau+1}, w_{\tau+1} \leftarrow$ generate_UT_points$(\mathbb{E}_s, \Sigma_s)$

5.5.5 Methodology: ECUT-MPPI

We leverage MPPI to propose an approximate solution to Problem 3 that is amenable to real-time implementation. We make two algorithmic contributions to applying MPPI for stochastic hybrid systems. **First, we use the Expansion-Compression UT for state prediction under stochastic dynamics with state-dependent distributions. Second, unlike previous works that realize state-dependent switching dynamics based only on the mean on predicted states [46], we switch dynamics based on the sigma particles. This is essential because although the switching function in (5.31) is a deterministic function of the robot's and agents's position, the uncertainty in the human's position implies that switching at a given time is also probabilistic. Using sigma points helps capture this stochasticity in switching.**

We modify the objective of MPPI to be risk-aware as follows. The stage-wise cost is split into two objectives: $Q(x_{r,\tau}) = Q_c(x_{r,\tau}) + Q_h(x_{r,\tau}, x_{p,\tau}, x_o)$, where $Q_c : \mathbb{R}^{n_r} \mapsto \mathbb{R}$ is for

convergence to the desired robot state $x_{r,d}$, and $Q_h : \mathbb{R}^{n_r} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_o} \mapsto \mathbb{R}^+$ is safety, i.e.,

$$\begin{aligned} Q_c(x_\tau) &= \mathbb{E}\left[\gamma_1 ||x_r - x_{r,d}||^2\right] \\ Q_h(x_\tau) &= \frac{\gamma_2}{\mu_{h,p} - \alpha\sigma_{h,p}} + \frac{\gamma_3}{h_{obs}}, \end{aligned} \tag{5.38}$$

where $(\mu_{h,p}, \sigma_{h,p})$ are the mean and standard deviation of the smallest distance value (5.32a) of distances to all the agents p at time t, h_{obs} is similarly the smallest distance of distances to all the obstacles o, and $\gamma_1, \gamma_2, \gamma_3 > 0$ are user-chosen parameters. α is a risk tolerance parameter that can be related to ϵ in (5.33e), with larger α promoting lower risk tolerance. Essentially, $\mu_{h,p} - \alpha\sigma_{h,p}$ represents the lower bound of the central confidence interval of a gaussian and, for example, for $\epsilon = 0.05$, $\alpha = 1.96$. The MPPI optimization is thus designed as

$$\min_{\mathbf{v}} \quad J(\mathbf{v}) = Q_c(x_{t+H}) + \sum_{\tau=t}^{t+H-1} \left(Q_c(x_\tau) + Q_h(x_\tau) + \frac{\lambda}{2} v_\tau^T \Sigma_\epsilon^{-1} v_\tau \right) \tag{5.39a}$$

$$\text{s.t.} \quad x_{\tau+1} = F(x_\tau, v_\tau + \epsilon_\tau) \tag{5.39b}$$

$$\epsilon \sim \mathcal{N}(0, \Sigma_\epsilon). \tag{5.39c}$$

5.5.6 Algorithm

Algorithm 8 shows the proposed framework and is also illustrated in Fig. 5.12. Let $\mathcal{S}_t^m, w_t^m$ be the sigma points and weights of the m^{th} sampled trajectory at time t. Further, let $\delta_{j,i,t}^m$ be the binary switching variable corresponding to mode j of the sigma point $\mathcal{S}_{t,i}^m$ corresponding to m^{th} sample.

In lines 2 and 3, we sample the perturbations and compute the control input for each sample. The timestamped robot trajectory, for 3 samples, is illustrated in Fig. 5.12 with the green, violet, and orange curves. Line 4 loops over the M sampled control perturbed control input sequences. For each control input sequence, line 7 loops over the time horizon and predicts future states. At each time t, Line 8 computes the cost incurred at the current state, lines 9-17 determine the operation mode of the human, and line 18 computes the human sigma points at the next time step. When the robot is outside the attention zone, described by the field-of-view in our example, of the human, the human dynamics is unaffected by the robot, shown by the human states for the first five-time steps in Fig. 5.12. Upon perceiving the robot in its field-of-view, which happens for first time at $t = 6$ in Fig. 5.12, each sigma point

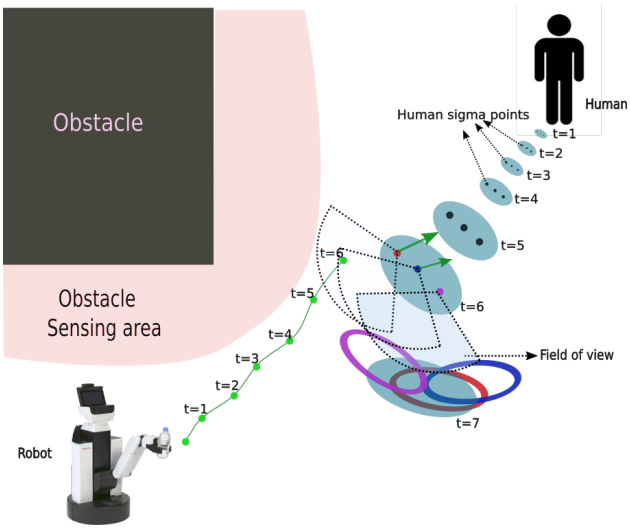


Figure 5.12: The robot's sampled trajectory (green) induces different responses from each of the human's (the non-ego agent in this example) sigma points. The ellipses represent the distribution of predicted human states and the dots inside them represent the sigma points. At $t = 6$, the human perceives the robot for the first time in its field-of-view for 2 out of 3 sigma points. The red and blue points get influenced by robot position at $t = 6$ while the pink point does not - resulting in three different future ellipses (of the same color respectively). The compression operation combines the three hollow ellipses into a single (solid) ellipse to represent human state distribution at $t = 7$. Note that distribution need not be an ellipse and depends on the variant of UT employed.

of the human determines its operation mode in lines 12 and 14 and continues propagating sigma points to future.

5.5.7 Simulation Results

In this section, we present two case studies to showcase our proposed framework. Code and videos available at https://github.com/hardikparwana/social-navigation. The videos are also available on thesis website https://github.com/hardikparwana/PhD-Thesis.

5.5.7.1 Multi-agent navigation

We evaluate the performance of our framework for short-term planning for a robot navigating amongst 10 non-ego agents and 2 circular obstacles. The scenario is shown in Fig 5.13(a). The robot is modeled as a single integrator with the following dynamics

$$\dot{r}_x = u_1, \dot{r}_y = u_2, \tag{5.40}$$

where $x_r = [r_x \ r_y]^T$ are the X, Y position of the robot and u_1, u_2 are the velocity control inputs. The states of the non-ego agents (other robots or humans for example) are given

Algorithm 8 ECUT-MPPI Controller

Require: (μ_0, Σ_0), $x_{r,0}$, H, F ▷ current human state moments, current robot state, time horizon, Stochastic Dynamics function

$\mathcal{S}_0, w_0 \leftarrow$ generate_UT_points(μ_0, Σ_0)
Sample $\boldsymbol{\epsilon}^m = [\epsilon_1^m, .., \epsilon_H^m] \sim \mathcal{N}(0, \Sigma_\epsilon), \forall m \in \{1, .., M\}$
$\mathbf{u}^m \leftarrow \mathbf{v} + \boldsymbol{\epsilon}^m$
for $m = 1$ to M **do** ▷ Loop over samples
    $\mathcal{S}_0^m, w_0^m \leftarrow \mathcal{S}_0, w_0,\ x_{r,0}^m \leftarrow x_{r,0}$
    $S^m = 0$ ▷ Initialize Cost
    for $t = 0$ to H **do** ▷ Loop over time horizon
        $S^m = S^m + Q(x_t^m) + \gamma (v_t^m)^T \Sigma_\epsilon^{-1} (u_t^m)$
        for all $j \in \{1, .., N\}$ **do** ▷ Operation Modes
            for all $\mathcal{S}_{t,i}^m \in \mathcal{S}_t^m$ **do** ▷ All Sigma points
                if $g(x_r^m, \mathcal{S}_{t,j}) \leq 0$ **then**
                ▷ Determine Switching variable
                    $\delta_{j,i,t}^m = 1$
                else
                    $\delta_{j,i,t}^m = 0$
                end if
            end for
        end for
        $\mathcal{S}_{t+1}^m, w_{t+1}^m \leftarrow \text{ECUT}(F(\delta_{i,j,t}^m), \mathcal{S}_t^m, w_t^m)$
        ▷ Get sigma points of next state Algorithm 7
        $x_{r,t+1}^m \rightarrow f_r(x_{r,t}^m, u_{r,t}^m)$
    end for
end for
$w^m \leftarrow$ (5.36)
$\mathbf{v}^+ \leftarrow$(5.37) **return** $\mathbf{v}^+$

by $x_p = [p_x\ p_y]^T$ where p_x, p_y are the position and the control inputs are given by velocities u_1, u_2. Every non-ego agent has a sensing radius that defines the switching criteria and thus the two operation modes: cooperative (when the robot is inside the sensing region) and uncooperative (otherwise). The non-ego agents use standard infinity going potential field (PF) for collision avoidance. Since PF is a deterministic model and actual non-ego motions in real life may differ from it, we augment it with the following disturbance model.

$$u = u_{nom} + PF(x_r, x_p, x_o) + \mathcal{N}\left(0, \alpha \tanh\left(\frac{\beta}{||[v_x\ v_y]||}\right)\right), \tag{5.41}$$

where $u_{nom} = [3, 0]^T$ is the nominal velocity of non-ego agents, $PF(x_r, x_p, x_o)$ is the input from PF, $\mathcal{N}(0, \Sigma)$ is a Gaussian distribution with zero mean and Σ covariance. The co-

variance is designed to be inversely proportional to the velocity of the non-ego agent, and $\alpha, \beta > 0$ are user-chosen parameters. We designed this model to mimic real-life scenarios where humans and other robots typically change their direction of motion at slower rather than faster speeds.

We compare our algorithm to a baseline based on Risk Aware MPPI [262]. We choose the risk metric proposed in (5.38) instead of CVaR that [262] exclusively uses since centralized confidence interval is easy to compute for both Monte Carlo and UT whereas CVaR estimation using UT is a relatively new formulation [104] and we refrain from using it in this work for simplicity. The method of risk evaluation is the same as [262] - for every m^{th} sampled control trajectory of the robot, we simulate K non-ego agent state trajectories where for each k_m^{th} trajectory, the non-ego agent motion is predicted by randomly sampling from its stochastic dynamics function resulting in a total of MK sampled trajectories. The risk for m^{th} robot control sample is thus evaluated using the corresponding K human trajectories. Note that treating each sigma-point as an independent sample, our method equivalently samples a total of MN trajectories where $N = 2n + 1$ for vanilla UT. In practice, $N << M$ for similar accuracy [82, 22] and therefore our objective here is to compare the achieved risk for similar computational speed.

Our MPPI code is written in JAX and the simulations are performed on a laptop running Ubuntu 22.04 with i9-13905H CPU. Our code performs in real-time on CPU with up to $M = 500$ samples and on GPU with up to $M = 20000$ samples. The prediction horizon is $H = 40$ with a time step of $\Delta t = 0.05s$. We also choose $\Sigma_\epsilon = 4.0$ in (5.35c), $\alpha = 4.0/\Delta t, \beta = 1.0$. We compare three cases: hybrid-dynamics aware, hybrid dynamics unaware, and mean-based prediction for $M = 500$ on CPU. In the aware case, the robot incorporates a non-ego agent's hybrid motion model in its prediction, whereas in the unaware case, it assumes that non-ego agents are always uncooperative. In the mean-based prediction, the robot performs switching for non-ego agent dynamics based only on their mean positions. We show that the aware case leads to lower costs while still maintaining safety requirements whereas the mean-based prediction leads to worse results compared to the dynamics aware case. We move the humans according to (5.41) and do a Monte Carlo evaluation of our framework by running it 50 times. Animations for the simulation can be found on our website. Figure 5.13(b) shows the cumulative cost as a function of time and Figs. 5.13(c) and 5.13(d) show the nearest distance to any non-ego agent and obstacle at a given time. Note that non-ego agents move independently in a decentralized fashion. The dynamics-aware MPPI that predicts non-ego agents' motion based on the knowledge of decentralized dynamics incurs lower costs and observes no constraint violation up to a 95% confidence interval of MC simulations. The mean-based prediction (green curves in Fig. 5.13) and dynamics-unaware

case (red curves) initially perform well but the performance degrades with time - the cost increases and constraints are violated (distance to humans is 0 and to obstacles boundary less than 0).

For RA-MPPI inspired method, we find that choosing $K = 20$ leads to a similar computation time per MPPI iteration as our method, and show its results in Fig. 5.14(a)-(c). We find that RA-MPPI insp[tired method fails to satisfy safety with the desired probabilistic bounds. For $K > 20$, the RA-MPPI inspired method takes longer than our method but to compare purely in terms of performance, we also show results for $K = 100$ (which runs 10 times slower than our algorithm) in Fig. 5.14(d)-(f). Note that $K = 200$ is much closer to our results (although still has violations) suggesting that the accuracy of UT-based prediction in our method is equivalent to predicting using large MC samples, which is expected.

We want to clarify that the implementation speed depends significantly on the programming language and hardware acceleration. Our claims on the number of particles required for similar performance between the inspired method and the proposed method are only representative of the behavior we observed in our implementation. Also, note that while the proposed method works better in our case study, Monte Carlo is more accurate than existing UT algorithms when a large number of particles are chosen. Monte Carlo is also better at representing multi-modal distributions than existing UT algorithms. Nevertheless, we reiterate that our method is suitable for real-time implementation where hardware limitations impose constraints on the number of particles. Moreover, our algorithms can always incorporate any improvements in UT algorithms. Finally, note that each non-ego agent is 2-dimensional, and therefore the total size of the state space is 22 and the computation time is expected to increase drastically despite software optimizations.

5.5.7.2 AWS Hospital Gazebo Experiments

We evaluate our framework on a gazebo simulator moving the robot in AWS Hospital environment that was developed in our previous work [172]. We run our JAX code on an NVIDIA RTX 4060 Laptop GPU with 10,000 samples.

The humans are modeled using a social force model [151] standard distance-based potential field (PF) to simulate more realistic motions than the standard potential field. Our algorithm though uses the model in (5.41) to predict humans' trajectories as SFM uses a 4-dimensional state per human which severely constrains the number of samples used by MPPI. We also limit trajectory prediction to the nearest 4 humans rather than all the humans. This results in our algorithm being run at 20 Hz.

The robot follows the unicycle model given by following equations $\dot{r}_x = u_1 \cos r_\theta, \dot{r}_y = u_1 \sin r_\theta, \dot{r}_\theta = u_2$ where r_x, r_y, r_θ are the position and heading angle of the robot and

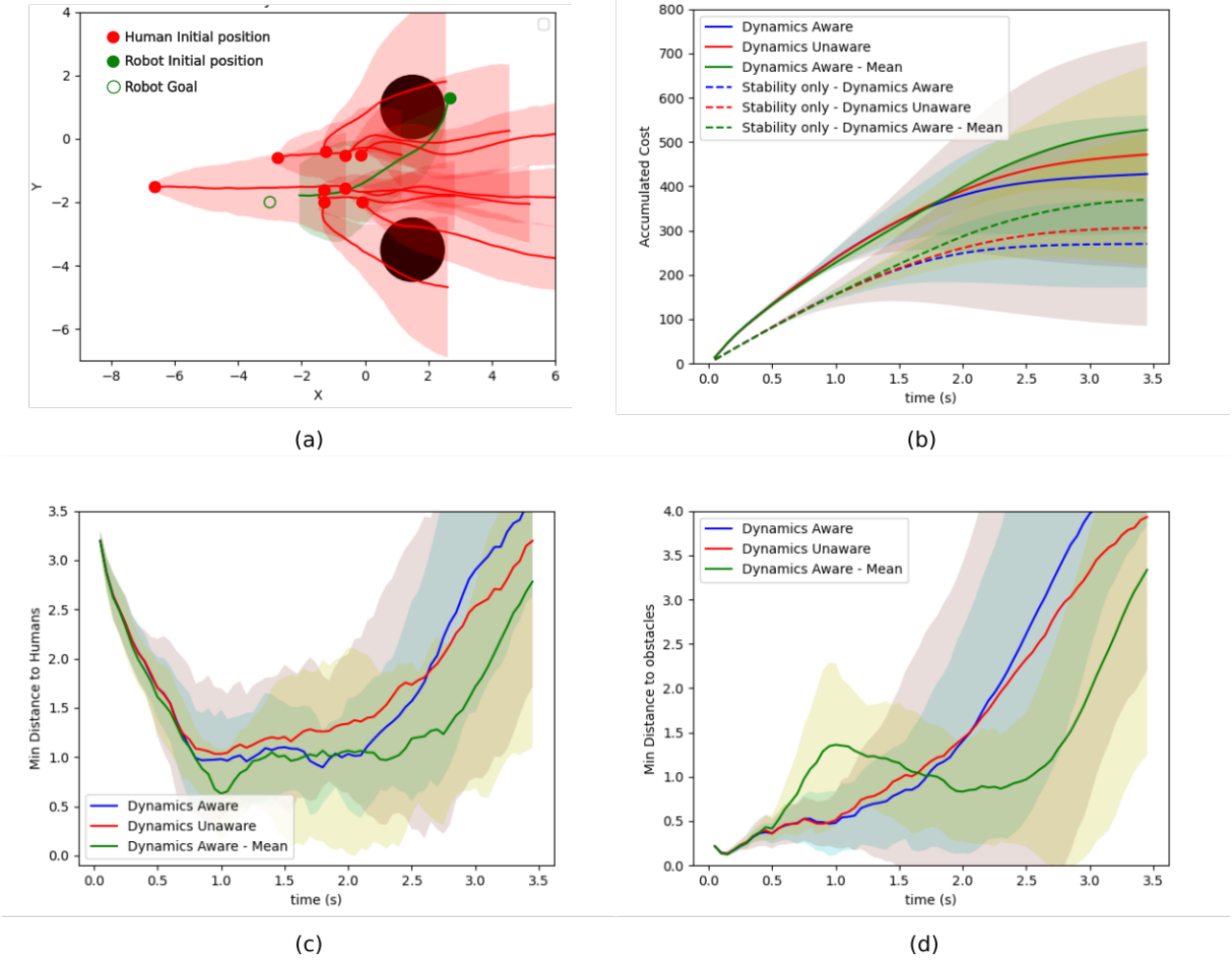


Figure 5.13: Simulation results for proposed method in Section 5.5.7.1. (a) The robot performs collision avoidance with humans (red) and obstacles (black). Robot trajectory is similarly shown in green. (b) Cost (5.39a) and stability cost (Q_c only) accumulated with time. (c), (d) Robot's minimum distance to any (c) human and (d) obstacle with time. The solid/dotted lines and shaded areas represent the mean and the 95% confidence interval in variation over 50 runs.

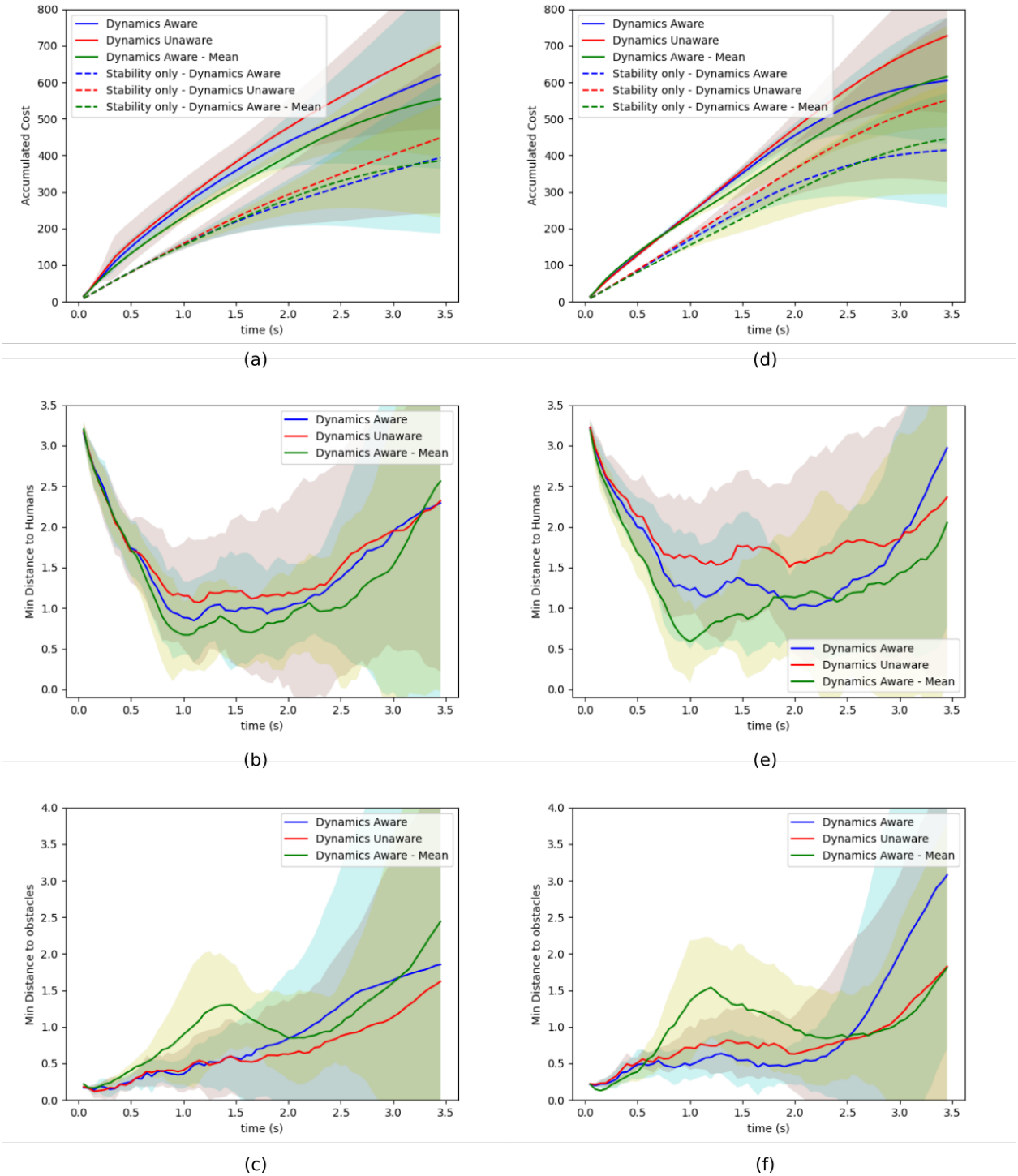


Figure 5.14: Simulation results for RA-MPPI inspired method in Section 5.5.7.1. (a),(b),(c) K =20 samples. Note from the increasing stability cost that the robot didn't converge to the goal location in the given time. (d), (e), (f) K =100 samples

u_1, u_2 are the linear and angular velocity control input. We use the ROS2 navigation stack's planner to plan a global path to the goal at the beginning and the proposed MPPI framework as a receding horizon controller to track the global path's waypoints. The video can be found on our paper GitHub website as well as thesis website https://github.com/hardikparwana/PhD-Thesis.

5.6 Conclusion

This chapter introduced a numerical scheme based on Unscented Transform to propagate state distributions across dynamics functions with state-dependent uncertainty. The numerical schme involved an expansion operation to increase We proposed a variant of MPPI using Unscented Transform for stochastic hybrid dynamical systems. Our future state prediction scheme can map the uncertainty of states to the uncertainty in switching. We are also able to implement our framework in real-time on both CPU and GPU subject to limits on the number of samples. Future work includes experimentation on real robots and evaluation of different types of hybrid dynamical systems.

CHAPTER 6

Graceful Degradation of Constraints

The preceding chapters of this chapter strived to ensure satisfaction of all the constraints. Chapter 3 added a new feasible space constraint while keeping the other constraint the same whereas Chapters 4 and 5 adapted the parameters of the controller.

This last chapter of the thesis delves into a different problem whose motivation comes from a scenario where any of the existing algorithms is unable to maintain all the constraints - **Which constraints should we remove and when?**

For this problem to be well defined, it is imperative to define a criteria for removing constraints. In this work, we assign additive priorities to the constraints and draw attention to some relevant work from optimization literature that has been unused in the context of dynamical systems. Specifically, optimization literature has proposed some solutions (with limited scope) to the problem of removing constraints given an infeasible optimization problem. However, they have not considered the correlation between solutions at different time steps when the coefficients of the optimization problem vary with the state of a dynamical system, which is typically the case when an optimization problems plays the role of a controller like MPC or CBF-QP. The aforementioned problem of removing constraints under additive priorities is known to be NP-Hard and this chapter attempts to model and exploit the correlation between solutions of controller at different time steps to generate a more informed prior to solving the NP-Hard problem.

This chapter addresses synthesizing receding-horizon controllers for nonlinear, control-affine dynamical systems under multiple incompatible hard and soft constraints. Handling incompatibility of constraints has mostly been addressed in literature by relaxing the soft constraints via slack variables. However, this may lead to trajectories that are far from the optimal solution and may compromise satisfaction of the hard constraints over time. In that regard, permanently dropping incompatible soft constraints may be beneficial for the satisfaction over time of the hard constraints (under the assumption that hard constraints are compatible with each other at the initial time). To this end, motivated by approximate

methods on the maximal feasible subset (maxFS) selection problem, we propose heuristics that depend on the Lagrange multipliers of the constraints. The main observation for using heuristics based on the Lagrange multipliers instead of slack variables (which is the standard approach in the related literature of finding maxFS) is that when the optimization is feasible, the Lagrange multiplier of a given constraint is non-zero, in contrast to the slack variable which is zero. This observation is particularly useful in the case of a dynamical nonlinear system where its control input is computed recursively as the optimization of a cost functional subject to the system dynamics and constraints, in the sense that the Lagrange multipliers of the constraints over a prediction horizon can indicate the constraints to be dropped so that the resulting constraints are compatible. The method is evaluated empirically in a case study with a robot navigating under multiple time and state constraints, and compared to a greedy method based on the Lagrange multiplier.

6.1 Related Work and Contributions

Autonomous robots are often employed to complete multiple (time-critical and safety-critical) tasks, such as to follow time-stamped waypoints, viewed as soft constraints, while avoiding obstacles, viewed as hard constraints. In the case when the tasks are incompatible, one may pose the question of how to relax, or permanently drop them. This problem is in general NP-hard, which is impossible to solve using any polynomial time algorithms assuming P $\neq$ NP. To address this question, we consider optimization problems with hard and soft constraints, and propose heuristics for removing soft constraints until all the hard constraints are met.

More specifically, in this chapter we consider the control synthesis for nonlinear control-affine systems under a receding-horizon optimization, subject to tasks that are captured by a set of hard and soft time-dependent state constraints. The objective is to maximize a performance score that depends both on safety (satisfaction of hard constraints) and task compatibility (selection of compatible soft constraints). We take a constrained optimization-inspired [146, 74] viewpoint on task selection. When an optimization problem is infeasible, i.e., it does not admit a solution, several methods exist to find the maximal subset of constraints that can be satisfied [66], a problem that is known to be NP-Hard. In general, finding this set has exponential complexity in the number of constraints and therefore heuristics are often proposed [195, 65]. A related problem is to find a subset of the given cardinality of a set that optimizes a performance metric. This problem is also known to be NP-hard [183, 70].

We propose a framework to decide which soft constraints to be dropped in polynomial time so that the optimization problem for the receding-horizon controller has a feasible

solution at every time step. We also show that selecting constraints that help maximize a performance metric for dynamical systems is NP-hard. We first demonstrate an application of Chinneck's Algorithm [64], a common method based on slack variables in the field of maximal feasible subset selection, to solving our problem. Next, we propose a novel heuristic based on Lagrange multipliers[117]. When the constraints are deemed incompatible under a given policy, we assign a Lagrange score to each of these constraints based on the variation in the corresponding Lagrange multipliers over the compatible time horizon. These scores are then used to decide the order in which constraints are dropped. To the best of the authors' knowledge, this is the first polynomial time algorithm using Lagrange multipliers for solving the NP-hard compatible constraint selection problems.

Compared to heuristics with slack variables, our heuristic utilizes information before the optimization problem becomes infeasible. On the contrary, slack variables will only have non-zero values, therefore meaningful information, at the exact time step when the optimization problem is infeasible. To evaluate the efficiency of the proposed heuristic, we compare it to solutions obtained by offline computation of approximate backward reachable sets, and the offline computation of control policies based on optimized Control Barrier Functions (CBFs).

With regards to the problem considered in our case study, i.e., deciding on tasks (timed waypoints) and generating safe (collision-free and time-respecting) trajectories to these tasks, the literature has considered approaches that are based on optimization and constraint relaxation, however none of the existing methods solves the problem in a systematic manner on how to drop the soft constraints, neither considers nonlinear, control-systems. For example, Model Predictive Control (MPC) and Control Barrier Functions (CBFs) methods can be used to enforce Signal Temporal Logic (STL) specifications for nonlinear system dynamics, [186, 135, 92, 90], however they can not choose or prioritize among constraints in order to prolong the feasibility of the optimization problem. With regards to some recent studies that address similar problems where a robot is assigned to travel through a series of regions: [101] allows a robot to execute tasks described by Linear Temporal Logic specifications; however, the approach assumes holonomic robots exhibiting straight-line motions, and uses a maximum velocity-based time-to-go estimate as a measure of the reachability of the target region. [160] and [239] perform reactive task allocation through relaxed CBF constraints in a CBF-QP controller by weighing the slack variables proportionally to a user-given priority of the constraint. However, all of the aforementioned approaches share the characteristics of a myopic controller, that is, they may enter regions from where the robot may not be able to recover and finish the high-priority tasks. They also do not provide an approach to choose between optional task constraints when they are incompatible with each other or with the other hard constraints imposed by the user. The issue of incompatible safety objectives

was recently addressed in [126] with a hierarchical relaxation of constraints in the order of absolute priorities defined by the user. However, their approach cannot be extended to cases where priorities have not been defined or are not assigned to individual constraints but to a collection of constraints.

In contrast to the aforementioned studies, our approach, while originally inspired from a time-and-safety-critical task (waypoint) allocation and following problem, offers a much more general framework for compatible constraint selection and hence recursively-feasible control synthesis. The rest of the chapter is structured as follows. Sections 6.3 and 6.4 present the problem formulation and proposed methods, respectively, and their evaluation is presented in Section 6.5.

6.2 Preliminaries

The set of real numbers is denoted as $\mathbb{R}$ and the non-negative real numbers as $\mathbb{R}^+$. Given $x \in \mathbb{R}^n$, $||x||$ denotes L_2 norm of x. Let $\mathbb{I}(y), y \in \mathbb{R}$ be step identity function such that $\mathbb{I}(y) = 1$ if $y \geq 0$ and $\mathbb{I}(y) = 0$ otherwise. The interior and boundary of a set $\mathcal{C}$ are denoted by $\text{Int}(\mathcal{C})$ and $\partial\mathcal{C}$. For $a \in \mathbb{R}^+$, a continuous function $\alpha : [0, a) \to [0, \infty)$ is a class-$\mathcal{K}$ function if it is strictly increasing and $\alpha(0) = 0$. Furthermore, if $a = \infty$ and $\lim_{r\to\infty} \alpha(r) = \infty$, then it is called class-$\mathcal{K}_\infty$.

Consider a discrete-time nonlinear dynamical system

$$x_{t+1} = f(x_t) + g(x_t)u_t, \tag{6.1}$$

where $x_t \in \mathcal{X} \subset \mathbb{R}^n$ and $u_t \in \mathcal{U} \subset \mathbb{R}^m$ represent the state and control input, and $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^m$ are locally Lipschitz continuous functions. The set of allowable states at time t is specified as an intersection of N sets $\mathcal{S}_i(t), i \in \{1, 2, .., N\}$, each of which is defined as the 0-superlevel set of a continuously differentiable function $c_i : \mathbb{R}^+ \times \mathcal{X} \to \mathbb{R}$ as:

$$\mathcal{S}_i(t) \triangleq \{x \in \mathcal{X} : c_i(t, x) \geq 0\}. \tag{6.2}$$

Define H_i: $\mathbb{R}^+ \times \mathcal{X} \to \mathbb{R}$ such that the super-level set of H_i at time t, denoted $\mathcal{S}_{H_i}(t) = \{x \in \mathcal{X} : H_i(t, x) \geq 0\}$, defines a constrained set that encodes n_H safety specifications or high-priority tasks that must be completed, where $i \in \mathcal{I}_H = \{1, \ldots, n_H\}$. Similarly, define h_j: $\mathbb{R}^+ \times \mathcal{X} \to \mathbb{R}$, where $j \in \mathcal{I}_h = \{1, \ldots, n_S\}$, and denote $\mathcal{S}_{h_j}(t) = \{x \in \mathcal{X} : h_j(t, x) \geq 0\}$ a constrained set that encodes n_S less prioritized tasks that can be removed if needed. Denote $N = n_H + n_S$.

6.3 Problem Formulation

Let $\pi : \mathbb{R}^+ \times \mathcal{X} \to \mathcal{U}$ be a Lipschitz continuous controller that is used by the autonomous agent to enforce safety and performance constraints. Let $\pi^{ref} : \mathbb{R}^+ \times \mathcal{X} \to \mathcal{U}$ be a reference input controller.

Now given the current state x_t at time t, let $x_{\tau|t}$ be the predicted future state at time $\tau + t$ obtained by forward propagation of x_t under dynamics (2.1) with input $u_{\tau|t} = \pi(\tau + t, x_{\tau|t})$, and reference input $u^{ref}_{\tau|t} = \pi^{ref}(\tau + t, x_{\tau|t})$. Note that $x_{0|t} = x_t$.

This chapter specifically considers the controller π defined by an optimization problem, termed as the Problem 0 (P_0), of the following form

$$\mathrm{P}_0 : \quad u_{\tau|t} = \pi(\tau + t, x_{\tau|t}) = \arg\max_u J(u, \pi_{ref}) \tag{6.3a}$$

$$\text{s.t.} \quad x_{\tau+1|t} = f(x_{\tau|t}) + g(x_{\tau|t})u \tag{6.3b}$$

$$H_i(t + \tau + 1,\ x_{\tau+1|t}) \geq 0, \forall i \in \mathcal{I}_H$$

$$h_j(t + \tau + 1,\ x_{\tau+1|t}) \geq 0, \forall j \in \mathcal{I}_h, \tag{6.3c}$$

where $J(u)$ is a control objective function. The optimization (6.3) may not admit a solution for a given time $\tau + t$ and state $x_{\tau|t}$ and in these cases, the controller is said to have failed.

Suppose that, for a time horizon T of interest, under the flow of dynamics (2.1), the original optimization problem (6.3) becomes infeasible at a prediction time step $t + \tau_{fail} \leq t + T$ and state $x_{\tau_{fail}|t}$. In such cases, to obtain a valid control input, one approach is to find a subset of the soft constraints with indices $\mathcal{I}^f_h \subset \mathcal{I}_h$ that give rise to a feasible problem in the whole time horizon when enforced.

More specifically, consider the following **Problem 1** (P_1), i.e., a reduced problem (called also thereafter a subproblem), where all the hard and a subset of the soft constraints are enforced:

$$\mathrm{P}_1 : \quad u_{\tau|t} = \quad \arg\max_u J(u, u_{ref}) \tag{6.4a}$$

$$\text{s.t.} \quad x_{\tau+1|t} = f(x_{\tau|t}) + g(x_{\tau|t})u \tag{6.4b}$$

$$H_i(t + \tau + 1,\ x_{\tau+1|t}) \geq 0, \forall i \in \mathcal{I}_H$$

$$h_j(t + \tau + 1,\ x_{\tau+1|t}) \geq 0, \forall j \in \mathcal{I}^f_h(t, x_t), \tag{6.4c}$$

where the set $\mathcal{I}^f_h(t, x_t)$ at time t, at state x_t, is such that the optimization problem (6.4) is feasible $\forall \tau \in \{0, 1, 2, \ldots, T\}$.

From now on, we use the terms "subproblem" and "reduced problem" interchangeably.

Definition 11. (Feasible subset) A set $\mathcal{I}_h^f(t, x_t)$ is called a feasible subset at time t and state x_t if the solution $u_{\tau|t}$ of (P_1) given out of (6.4) exists $\forall \tau \in \{0, 1, \ldots, T\}$.

A feasible subset $\mathcal{I}_h^f(t, x_t)$ can be non-unique, and therefore we search for the feasible subset that furthermore satisfies certain properties. Denote $\mathcal{P}(\mathcal{I}_h)$ the power set of $\mathcal{I}_h$, and denote $\mathcal{FP}(\mathcal{I}_h)(t, x_t) \subseteq \mathcal{P}(\mathcal{I}_h)$ the set that comprises all the elements of $\mathcal{P}(\mathcal{I}_h)$ that are feasible subsets of $\mathcal{I}_h$ at time t and state x_t.

Our first objective is to find the feasible subset of maximal cardinality, as defined in Objective 3:

Objective 3. If the set $\mathcal{X}_H(t) = \{x_t \in \mathcal{X} \mid H_i(t, x_t) \geq 0, \forall i \in \mathcal{I}_H\}$ is non-empty at time t, $\mathcal{X}_H(t) \neq \emptyset$, find the element of $\mathcal{FP}(\mathcal{I}_h)(t, x_t)$ of maximal cardinality, denoted $\mathcal{I}_h^{*f}(t, x_t)$, or in other words, the maximal feasible subset of (P_1), defined as:

$$\mathcal{I}_h^{*f}(t, x_t) = \arg\max \left|\mathcal{FP}(\mathcal{I}_h)(t, x_t)\right|. \tag{6.5}$$

More generally, for the case when a reward function of the form $R : \mathcal{P}(\mathcal{I}_h) \to \mathbb{R}$ is given, we consider the maximization of the given reward as:

$$\mathcal{I}_h^{*f}(t, x_t) = \arg\max R(\mathcal{FP}(\mathcal{I}_h)(t, x_t)). \tag{6.6}$$

Objective 3 is a NP-hard problem. To see why, consider a simple scenario in which (P_0) : $\pi(\tau + t, x_{\tau|t})$ is feasible at the prediction time $\tau = 0$, but becomes infeasible at time $\tau = 1$. The selection of feasible subsets of soft constraints when considering only $\pi(1 + t, x_{1|t})$ is equivalent to the maximum feasible subset problem (maxFS), which is a known NP-hard problem. Objective 3 considers not only $\pi(1+t, x_{1|t})$ but also its correlation with $\pi(0 + t, x_{0|t})$ when finding feasible subset of soft constraints, therefore it is NP-hard equivalently. Specifically, Objective 3 is a combinatorial problem with worst-case exponential complexity as the power set $\mathcal{P}(\mathcal{I}_h)$ has 2^{n_S} elements.

6.4 Methodology

In this section, we propose a heuristic approach to solve for Objective 3. Consider the point-wise optimal QP controller (6.3) that enforces $N = n_H + n_S$ (hard and soft) constraints. Let the trajectory at a future time step τ resulting from forward-propagating the closed-loop system (2.1) under the controller (6.3) at time t, be given by $(x_{\tau|t}, u_{\tau|t}, \tau \in \{0, .., \tau_{fail}\})$ where $\tau_{fail} \in \{0, .., \infty\}$ is such that a solution to (6.3) exists for all $\tau < \tau_{fail}$ and does not exist for $\tau = \tau_{fail}$.

6.4.1 Chinneck's algorithm

Several heuristics have been developed for finding the maxFS. In this section, we first briefly review one such iterative algorithm, Chinneck's algorithm, and propose a simple extension of it to our problem. Consider the following optimization problem:

$$\min \quad J(u) \tag{6.7a}$$

$$\text{s.t.} \quad c_i(u) \leq 0, \;\; i \in \mathcal{I}_c = \{1, .., N\}. \tag{6.7b}$$

where c_i's are constraints to the optimization problem and suppose that (6.7) is infeasible. The idea is that in each iteration of the algorithm, Chinneck's algorithm removes one constraint permanently until a feasible subset of the constraints is found. More specifically, let the set of constraint indices at iteration $iter \in \{0, 1, N-1\}$ be $\mathcal{I}^{iter}$ with $\mathcal{I}^0 = \mathcal{I}^c$. At an iterative step with index set $\mathcal{I}^{iter}$, each constraint $k \in \mathcal{I}^{iter}$ is assigned a score, hereby called Chinneck's score, by evaluating the following reduced relaxed problem $\text{RRP}(\mathcal{I}^{iter})_k$ that result by removing k^{th} constraint and adding slack variables to the remaining constraints in (6.7).

$$\text{RRP}(\mathcal{I}^{iter})_k : \min \quad J(u) + M \sum_{i \in \mathcal{I}^{iter} \setminus \{k\}} \delta_{c_i}^2 \tag{6.8a}$$

$$\text{s.t.} \quad c_i(u) \leq \delta_{c_i}, \; i \in \mathcal{I}^{iter} \setminus \{k\}. \tag{6.8b}$$

where δ_{c_i} are slack variables, while adding $M\delta_{c_i}^2$ with $M \gg 1$ promotes $\delta_{c_i} = 0$. The Chinneck's score of constraint k is now taken as the sum of all slack variables $\delta_{c_i}, i \in \mathcal{I}^{iter} \setminus \{k\}$ of $\text{RRP}(\mathcal{I}^{iter})_k$ and is an indication of how much the constraints, $\{c_i \geq \delta_{c_i} \; \forall i \in \mathcal{I}^{iter} \setminus \{k\}\}$, had to be violated for the $\text{RRP}(\mathcal{I}^{iter})_k$ to admit a feasible solution without relaxation:

$$r_{chinneck}(\text{RRP}(\mathcal{I}^{iter})_k) = \sum_{i \in \mathcal{I}^{iter} \setminus \{k\}} \delta_{c_i}. \tag{6.9}$$

The constraint k with the largest Chinneck's score $r_{chinneck}(\text{RRP}(\mathcal{I}^{iter})_k)$ is permanently removed at iteration $iter$, and the process is repeated until a feasible solution in which all slack variables are zero is found.

6.4.2 Application of Chinneck's Score to Problem P_0

In order to apply Chinneck's method to Problem P_0 given by (6.3), we have to account for the fact that we have a dynamical system, i.e., that the control inputs and the states are functions of time (trajectories), and not fixed (constant) with respect to time, and so we

perform the following modification: At each iterative step, we first remove one soft constraint $k \in \mathcal{I}_h^{iter} \subset \mathcal{I}_h$ temporarily, which results in a reduced problem (RP) as defined:

$$\text{RP}(\mathcal{I}_h^{iter})_k : u_{\tau|t} = \arg\max_u J(u) \tag{6.10}$$

$$\text{s.t.} \quad x_{\tau+1|t} = f(x_{\tau|t}) + g(x_{\tau|t})u \tag{6.11}$$

$$H_i(t+\tau+1,\ x_{\tau+1|t}) \geq 0, \forall i \in \mathcal{I}_H$$

$$h_j(t+\tau+1,\ x_{\tau+1|t}) \geq 0, \forall j \in \mathcal{I}_h^{iter} \setminus \{k\}, \tag{6.12}$$

and propagate the state x_t under dynamics (2.1) and (6.12) for $\tau = \{0, 1, .., \min(T, \tau_{fail} - 1)\}$ where τ_{fail} is the time at which (6.12) becomes infeasible. If $\tau_{fail} \leq T$, we evaluate a reduced relaxed problem (RRP) and compute the sum of slack variables of all the hard and the present soft constraints at the failing step τ_{fail}. More specifically, we consider each RRP $(\mathcal{I}_h^{iter})_k$ given as:

$$\text{RRP}(\mathcal{I}_h^{iter})_k : u_{\tau_{fail}|t} = \arg\max_u J(u) \tag{6.13}$$

$$\text{s.t.} \quad x_{\tau_{fail}+1|t} = f(x_{\tau_{fail}|t}) + g(x_{\tau_{fail}|t})u \tag{6.14}$$

$$H_i(t+\tau_{fail}+1,\ x_{\tau_{fail}+1|t}) + \delta_{H_i} \geq 0, \forall i \in \mathcal{I}_H$$

$$h_j(t+\tau_{fail}+1,\ x_{\tau_{fail}+1|t}) + \delta_{h_j} \geq 0, \forall j \in \mathcal{I}_h^{iter} \setminus \{k\},$$

and compute a Chinneck score as:

$$r_{chinneck}(\text{RRP}(\mathcal{I}^{iter})_k)) = \sum_{\forall i \in \mathcal{I}_H} \delta_{H_i} + \sum_{\forall j \in \mathcal{I}_h^{iter} \setminus k} \delta_{h_j} + \frac{1}{R(\mathcal{I}_h^{iter} \setminus \{k\})}. \tag{6.15}$$

Similarly to the original Chinneck's algorithm, we then permanently drop the constraint whose removal corresponds to a reduced relaxed problem with the largest slack score at each iterative step, and stop until the slack score becomes zero, or all the soft constraints have been removed. A pseudo-code can be found in Algorithm 9.

6.4.3 Proposed Algorithms based on Lagrange Score

Chinneck's algorithm utilizes slack variables in the heuristic to find the feasible subsets. However, the slack variables become non-zero only when the optimization problem is infeasible, i.e., at time step τ_{fail}. Therefore, the feasible time steps $\tau \in \{0, .., \tau_{fail} - 1\}$ do not contribute to the slack score. However, (2.1) being a dynamical system, we hypothesize that

the time steps preceding failure provide meaningful information about the evolution of the constraint functions over the system trajectories. In accordance with this hypothesis, we propose to use Lagrange multipliers resulting from the optimizations (6.4), rather than the slack variables.

Our key observation regarding the slack variable and the Lagrange multipliers corresponding to a single constraint is that the former is zero if the problem is feasible, while the latter is not; in that regard, the Lagrange multipliers gives more information about the strictness of the constraint (how much it constrains the feasible space), which is particularly useful when solving for dynamical trajectories over time horizons. It is worth noting that Lagrange multipliers have been known to offer insight into the satisfiability of the constraints and are often used to select candidate constraints that could be made feasible, for example, in the popular active set method [249] and interior point methods [181] for solving quadratic programs. Our novelty lies in utilizing Lagrange multipliers to evaluate the satisfaction of a constraint over a time horizon for an evolving system such as (2.1), and using it to find a feasible subset of constraints.

Let us first consider the problem (6.4) solved for time horizon $T' = \min(T, \tau_{fail})$ for $\mathcal{I}_h^{'} \subset \mathcal{I}_h$, where $\mathcal{I}_h'$ is any subset of $\mathcal{I}_h$. Let $\lambda_{h_j}(\tau)$ be the Lagrange multiplier of the inequality constraint $h_j(t+\tau+1, x_{\tau+1|t}) \geq 0$ resulting from solving (6.4) at time τ. We assign a *Lagrange value* l_{h_j} to each soft constraint h_j as follows

$$l_{h_j} = \sum_{\tau=0}^{\tau_{fail}-1} \lambda_{h_j}(\tau). \tag{6.16}$$

Since Lagrange multipliers are non-zero only when the respective constraints are active, we hypothesize that often the impact a constraint has on the feasibility of (6.4) over a time horizon is proportional to its Lagrange value, and therefore we remove the constraint with the highest l_{h_j}. We also note that using $\text{sign}(\lambda)$ instead of λ yields similar performance. In the ensuing however, we only present results with (6.16). In what follows, we will describe two heuristics utilizing the Lagrange values of the constraints to compute feasible subsets.

6.4.3.1 Proposed Method on Finding Feasible Subproblems based on Lagrange Multipliers

Similar to Chinneck's algorithm, we present an iterative procedure to remove constraints successively. Consider the relaxed subproblem (6.12) that results from removing constraint $k \in \mathcal{I}_h^{iter}$ at iteration $iter$. We forward propagate the state x_τ under (2.1) and (6.12) for a

Algorithm 9 Feasible Subproblem Algorithm

$\mathcal{I}_h^0 = \{1, .., n_S\} = \mathcal{I}_h$ ▷ Constraint Indices to consider
$\mathcal{I}_h^f = \emptyset$ ▷ Feasible Index Set Initialization
for $iter \in \{0, 1, \ldots, n_S - 1\}$ **do**
    $r_{min} = INF$
    if Chinneck **then**
        $r_k = r_{chinneck}(\text{RRP}(\mathcal{I}_h^{iter})_k)$
    else
        $r_k = r_{lag}(\text{RP}(\mathcal{I}_h^{iter})_k)$
    end if
    if $r_k < r_{min}$ **then**
        $r_{min} = r_k$, $k_{drop} = k$
    end if
    $\mathcal{I}_h^{iter+1} = \mathcal{I}_h^{iter} \setminus k_{drop}$
    if $r_{min} == 0$ **then**
        $\mathcal{I}^f = \mathcal{I}^{iter+1}$, break
    end if
end for
return $\mathcal{I}_h^f$

time horizon $T' = \min(T, \tau_{fail})$. The Lagrange score of constraint k is designed as follows

$$r_{lag}(\text{RP}(\mathcal{I}_h^{iter})_k) = \begin{cases} \sum_j^{\mathcal{I}_h^{iter}\setminus\{k\}} l_j + \frac{1}{R(\mathcal{I}_h^{iter}\setminus\{k\})} & \text{if } \tau_{fail} \leq T \\ \frac{1}{R(\mathcal{I}_h^{iter}\setminus\{k\})} & \text{otherwise.} \end{cases} \tag{6.17}$$

Now we remove the constraint with largest Lagrange score and the procedure continues until we find a feasible solution or remove all soft constraints as described in Algorithm 9.

6.4.3.2 A Greedy Approach based on Lagrange Multipliers

The approach in Algorithm 9 requires solving optimization problems for multiple subproblems in each iteration. This procedure can have a high computational overhead that might not be suitable for real-time implementation. In this section, we present a greedy approach Algorithm 10 to select a subset of soft constraints by dropping them in the order of Lagrange value. Specifically, consider the subproblem (6.4) solved for a constraint set $\mathcal{I}_h^{iter} \subset \mathcal{I}_h$ and τ_{fail} be the time (6.4) first becomes infeasible. Then we assign the following score to each

constraint k

$$r_{lag}(\mathrm{P}_1(\mathcal{I}_h^{iter})_k) = l_{h_k}, \tag{6.18}$$

and permanently drop the constraint with the largest score. We repeat this procedure until a feasible solution is found ($\tau_{fail} > T$) or all the soft constraints are dropped.

Algorithm 10 Greedy Algorithm

x_0 ▷ Current state of the system
$\mathcal{I}_h^0 = \{1, .., n_S\} = \mathcal{I}_h$ ▷ Constraint Indices to consider
$\mathcal{I}_h^f = \emptyset$ ▷ Feasible Index Set Initialization
for $iter \in \{0, n_S - 1\}$ **do**
    $k_{max} = \arg\max_k(r_{lag}(\mathrm{P}_1(\mathcal{I}_h^{iter})_k))$
    if $\mathrm{P}_1(\mathcal{I}_h^{iter})$ is feasible **then**
        $\mathcal{I}_h^f = \mathcal{I}_h^{iter}$, break
    end if
    $\mathcal{I}_h^{iter+1} = \mathcal{I}_h^{iter} \setminus k_{max}$
end for
return $\mathcal{I}_h^f$

6.4.3.3 Complexity Analysis

For the subproblem-based algorithm, the complexity of the number of optimizations to be solved in the worst-case, corresponding to the case when all soft constraints are removed, is $O(n_S^2 T)$ because in every iteration $iter \in \{0, .., n_S - 1\}$, $\tau \in \{0, .., T - 1\}$, it needs to solve $n_S - 1$ optimizations. The overall computational complexity also depends on the solve time for each optimization (6.3). For quadratic programs, for example, polynomial time solvers are available[206] (although QP, being NP-Hard in general, have exponential complexity). For such solvers, the complexity of the subproblem approach is $O(n_S^4 T)$. As for the greedy algorithm, since it drops one constraint at each step and checks whether the reduced problem is feasible or not, its worst-case complexity of the number of solved optimizations is $O(n_S T)$, and computational complexity for quadratic programs is $O(n_S^3 T)$.

6.5 Simulation Results

In this section, we illustrate the efficacy of our algorithm through simulations. Consider the scenario shown in Figs. 6.1, 6.2 and 6.3, in which a robot is tasked to track a series of time-stamped waypoints, i.e., the robot is required to arrive at the j-th waypoint before a

user-defined time T_j. The strict requirements on the robot are avoiding collision with the walls and reaching the final target position within user-defined time T_{final}. Let the robot position at time t be denoted p_t. The constraints to reach a waypoint p_j and the final target position p_{tf} at maximum allowable distance of d_{max} are formulated through the following Control Lyapunov functions (CLFs) [204]:

$$V_j(t, p_t) = d_{max}^2 - (p_j - p_t)^T(p_j - p_t) \tag{6.19}$$

$$V_{tf}(t, p_t) = d_{max}^2 - (p_{tf} - p_t)^T(p_t - p_t). \tag{6.20}$$

The constraints for the waypoints are encoded via the CLF conditions:

$$h_j(t, p_t) = \phi_j(t)\Big[V_j(t+1, p_{t+1}) - (1 + \alpha_{h_j})V_j(t, p_t)\Big]$$

$\forall j \in \mathcal{I}_h(t) = \{1, \ldots, n_{waypoints}\}$, where $\alpha_{h_j} > 0$ and $\phi_j(t)$ is the activation function defined as:

$$\phi_j(t) = \begin{cases} 1, & \text{if } T_{j-1} < t < T_j \\ 0, & \text{otherwise} \end{cases} \tag{6.21}$$

where T_j is a user-defined time specification so that the robot has to arrive at the j-th waypoint before T_j. When we remove waypoints, we shift the lower bound of the activation function of each included waypoint to the upper bound of the activation function of the included waypoint just before it accordingly. As a result, $[T_{j-1}, T_j]$ is the active time window of the j-th waypoint. Similarly, the hard constraint for arrival at the final target is defined as:

$$H_{tf}(t, p_t) = \phi_{tf}(t)\Big[V_{tf}(t+1, p_{t+1}) - (1 + \alpha_{H_{tf}})V_{tf}(t, p_t)\Big],$$

where $\alpha_{H_{tf}} > 0$, and ϕ_{tf} is defined with a final time T_{final} so that the robot is required to arrive at the final target position before T_{final}. Collision avoidance with static obstacles, which are approximated as a series of circular disks, is formulated using the following barrier function

$$b_{Hi}(t, p_t) = (p_t - x_{oi})^T(p_t - x_{oi}) - d_{min}^2, \tag{6.22}$$

and the Control Barrier Function (CBF)[15, 134] condition

$$H_i(t, p_t) = b_{Hi}(t+1, p_{t+1}) - (1 + \alpha_{Hi}) b_{Hi}(t, p_t), \tag{6.23}$$

$\forall i \in \mathcal{I}_H(t) = \{1, \ldots, n_{obstacles}\}$, where x_{oi} is the position of the centroid of the i-th circular static obstacle, and d_{min} is the radius of the obstacle.

The CLF-CBF-QP[15] controller that is employed is

$$\begin{aligned} \min_{u_{\tau|t} \in \mathbb{R}^m} \quad & J(u_{\tau|t}) = (u_{\tau|t} - u_{ref,\tau|t})^T (u_{\tau|t} - u_{ref,\tau|t}) \\ \text{s.t.} \quad & h_j(\tau|t, x_{\tau+1|t}) \geq 0, \ j \in \{1, .., n_{waypoints}\} \\ & H_i(\tau|t, x_{\tau+1|t}) \geq 0, \ i \in \{1, .., n_{obstacles}\} \\ & H_{tf}(\tau|t, x_{\tau+1|t}) \geq 0, \ u_{\tau|t} \leq 1.0, \end{aligned} \tag{6.24}$$

where $u_{ref,\tau|t}$ is the reference control signal at time $\tau + t$, designed to steer the robot towards the active waypoint.

In our implementation, the QP controller (6.24) is used to generate control inputs for the robot, which is modeled either under single-integrator dynamics (Test 1) or under the dynamic unicycle (Test 2) dynamics.

The environment is affected by known disturbances, whose magnitude is represented with light blue color in Figs. 6.1, 6.2, 6.3. The nonlinear disturbance may lead to a collision in the presence of input bounds if the robot navigates to one of the shaded areas. Thus, the robot may have to drop some of the soft constraints corresponding to blue waypoints to be able to reach the last red waypoint while remaining safe. In both test cases, the original problem, P_0, defined in (6.24), is rendered infeasible within the initial time interval $[0, T]$, where T is the controller's prediction time horizon. This infeasibility arises when mandating arrival at every waypoint with the presence of disturbances, necessitating the removal of some waypoints at $t = 0$. We evaluated all results on Intel® Core™ i7-9700 CPU @ 3.00 GHz with Python 3.8.10.

6.5.0.1 Test 1

We first start with each waypoint being rewarded equally, and we compare the number of waypoints arrived between different algorithms. The prediction time horizon in this scenario is equal to $T_{final} = 250$ (equivalent to 25s in real-time). We primarily seeks to underscore the advantages of selectively removing soft constraints over the common method of adding slack variables when facing infeasibility issues. Fig. 6.2(a) illustrates that while introduc-

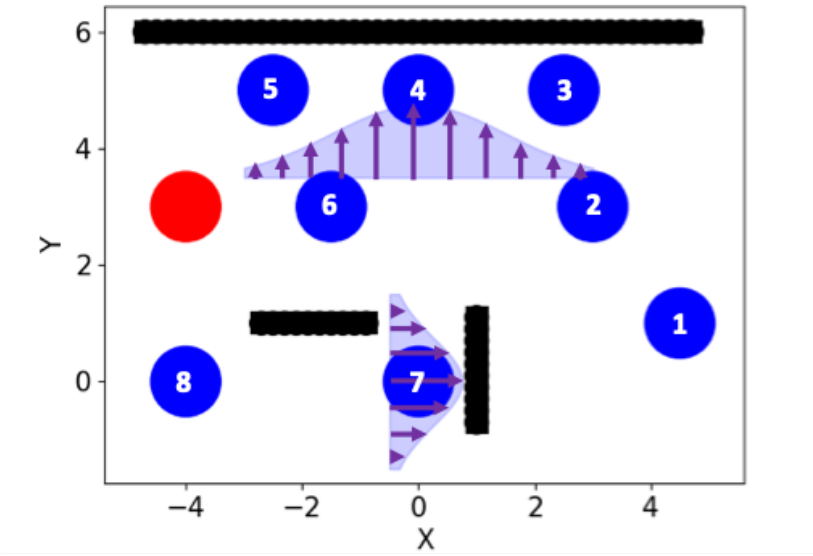
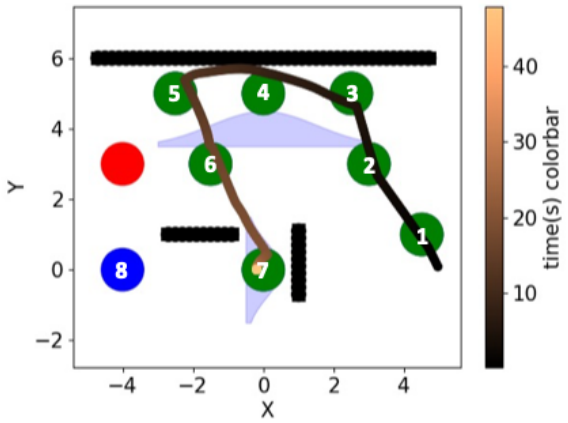


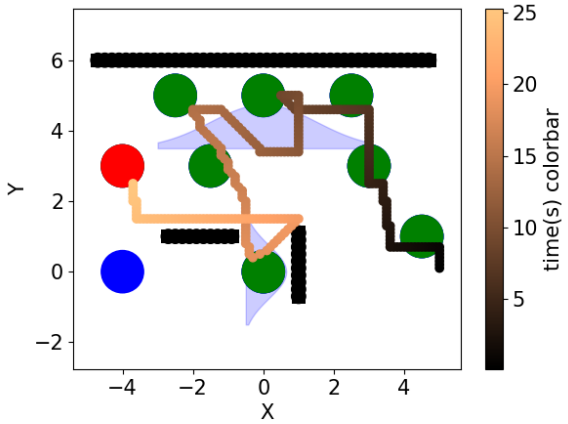
Figure 6.1: A sample testing environment, in which each waypoint is shown in blue and the shaded blue regions show disturbances affecting the robot. The robot is required to travel through the series of waypoints in a specified order (as shown in white numbers) and returns to the red final position before T_{final}.

ing slack variables allows the robot to reach waypoints, disturbances can prevent it from reaching the final target within the user-defined time, thus violating the hard constraints. In contrast to this case, our proposed method of removing selected soft constraints, as will be illustrated through Figs. 6.2(b)-(f) and 6.3, ensures that the robot meets all safety-related hard constraints and successfully reaches the target.

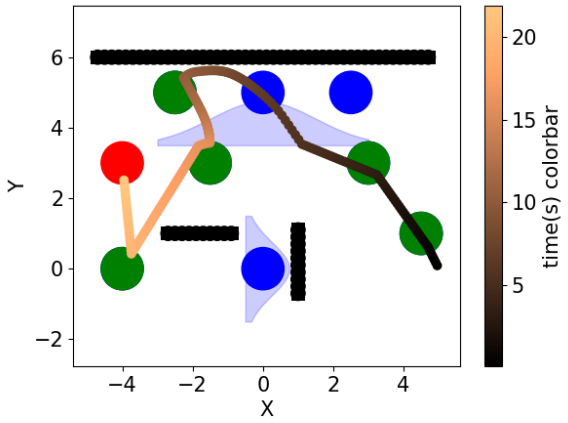
Next, we present results with 7 different testing environments under three different numbers of waypoints and under three different disturbance levels in TABLE 6.1. The "Offline" columns of the Table include the methods we implemented for comparison, namely: A "Reachability" approach, where the optimal solution is obtained by computing backward reachable sets as mentioned in Appendix C.1.2, and an "Exhaustive CBF search" approach, where the reward $R(\mathcal{P}(\mathcal{S}_h)_f)$ is designed to maximize the number of waypoints reached, and the Objective 3 is solved using an exhaustive search by evaluating all possible QP controllers of the form (6.24) with different combinations of soft constraints at every time. The results of our methods are shown in the "Heuristic" columns. Namely, we show the results of the "Greedy" heuristic that utilizes the Lagrange score r_{lag}, of Chinneck's method that utilizes Chinneck's heuristic score $r_{chinneck}$, and of the subproblem method that utilizes the Lagrange score r_{lag}, respectively. We apply all the heuristics at the initial time and find a subset of constraints that can be rendered feasible for all time in the future (specifically till T_{final}). Once such a subset is found, the robot implements the receding-horizon controller with this



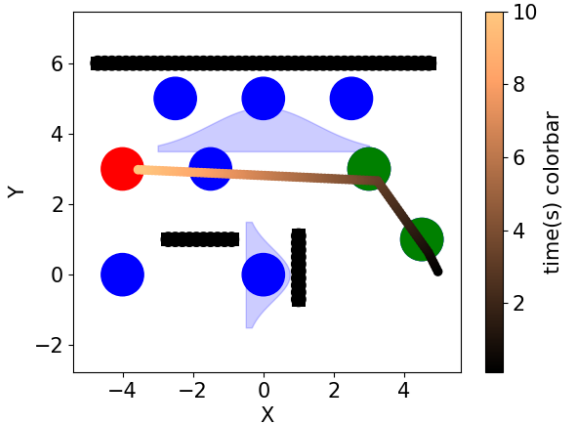
(a) Failure with slack variables



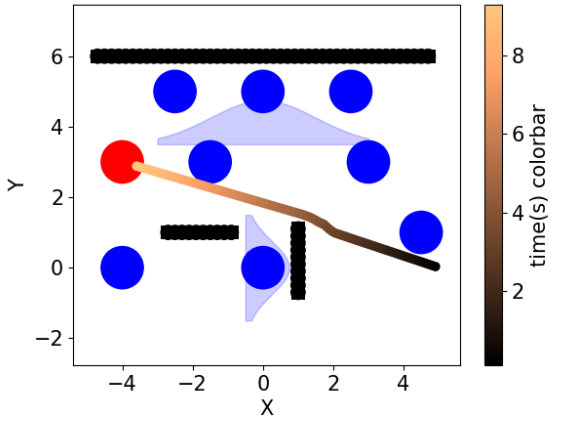
(b) Reachability



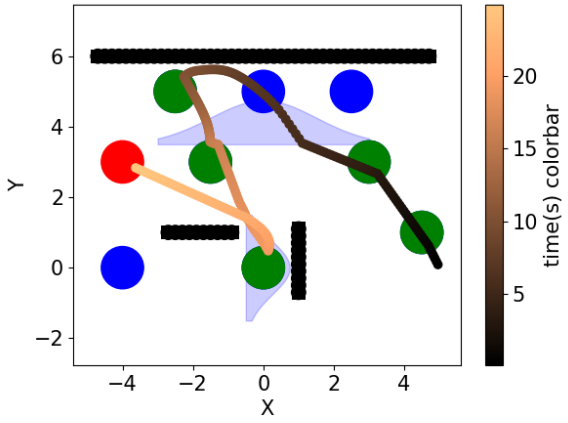
(c) Exhaustive CBF Search



(d) Algorithm 2 with r_{lag}



(e) Algorithm 1 with $r_{chinneck}$



(f) Algorithm 1 with r_{lag}

Figure 6.2: Performance of different algorithms for 9 waypoints. Note that the waypoints 4 in (c), 6 in (d), and 4 in (f), were dropped by the controller as they could not be reached within the prescribed time. The paths just happen to pass through them while the robot aims at the next waypoint.

subset of constraints and the trajectories with 8 waypoints and the Medium disturbance level from all cases are shown in Fig. 6.2.

From TABLE 6.1, the calculation time for "Reachability" is of the order 20000 seconds. The number of subproblems required in the exhaustive search for the best CBF controller grows exponentially in the number of constraints, thus its calculation increases dramatically with an increased number of waypoints. On the other hand, the subproblem algorithm with both heuristics and the greedy algorithm only needs to solve a polynomial or linear number of subproblems with respect to the number of all soft constraints. We observe that the subproblem algorithm with either $r_{chinneck}$ or r_{lag} heuristic can arrive at more waypoints compared to the greedy algorithm, but at the cost of solving more subproblems, thus taking a longer computational time. We also observe the expected improved performance in terms of more waypoints arrived when replacing the heuristic from $r_{chinneck}$ to r_{lag}. Furthermore, the subproblems that the subproblem algorithm needed to solve grow linearly with the number of soft constraints, which is more efficient than the exhaustive search method, which needs to solve an exponentially increasing number of subproblems with an increase in number of soft constraints.

		OPTIMAL				HEURISTICS					
No. Way-points	Disturb Level	Reachability		Exhaustive CBF Search		Algorithm 2 based on r_{lag}		Algorithm 1 based on $r_{chinneck}$		Algorithm 1 based on r_{lag}	
		No. Way-points Reached	Comp. Time	No. Way-points Reached	Comp. Time	No. Way-points Reached	Comp. Time	No. Way-points Reached	Comp. Time	No. Way-points Reached	Comp. Time
6	Low and Medium	6	5.55h	6	1.74s	Every CBF methods reaches 6 waypoints with the same computational time because the original problem is feasible with CBF based controller					
	High			4	20.86s	1	2.58s	2	8.87s	2	8.56s
8	Low	8	5.55h	7	560.6s	7	4.64s	7	23.24s	7	22.69s
	Medium	7		5	236.3s	2	4.14s	0	19.64s	5	15.04s
	High	5		4	156.2s	2	3.63s	1	17.47s	2	19.79s
14	Low	14	5.55h	12	3.61h	12	6.95s	5	234.32s	9	164.01s
	Medium	13		9	2.61h	1	7.01s	2	55.52s	3	48.54s
	High	12		8	2.55h	2	6.34s	0	52.18s	2	47.19s

Table 6.1: Number of waypoints reached with different algorithms under the single integrator dynamics.

	OPTIMAL		HEURISTICS			
Reward	Exhaustive CBF Search		Algorithm 1 based on $r_{chinneck}$		Algorithm 1 based on r_{lag}	
	Reward	Time	Reward	Time	Reward	Time
Case 1	17	7.22h	0	123.08s	12	118.49s
Case 2	18	6.32h	0	124.31s	3	119.02s
Case 3	12	6.33h	0	123.56s	5	119.25s

Table 6.2: Reward achieved and computation time for different algorithms. rewards for each waypoint are as follows: Case 1: [1,1,1,1,1,2,2,2,2,2,4,4,4,4], Case 2: [4,4,4,4,4,2,2,2,2,2,1,1,1,1], Case 3: [1,2,3,4,1,2,3,4,1,2,3,2,1,2].

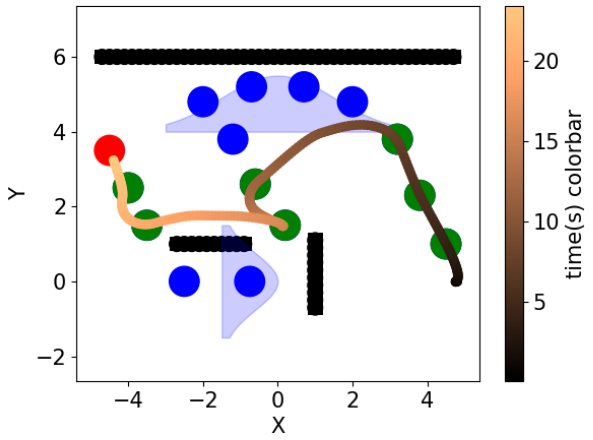
6.5.0.2 Test 2

For our second testing scenario, we considered an environment with 14 waypoints and "High" disturbance level, as defined in Test 1. The time horizon for our controller in this scenario is also equal to $T_{final} = 250$. In contrast to Test 1, where each waypoint is rewarded equally, waypoints in Test 2 are assigned different rewards as described in TABLE 6.2.

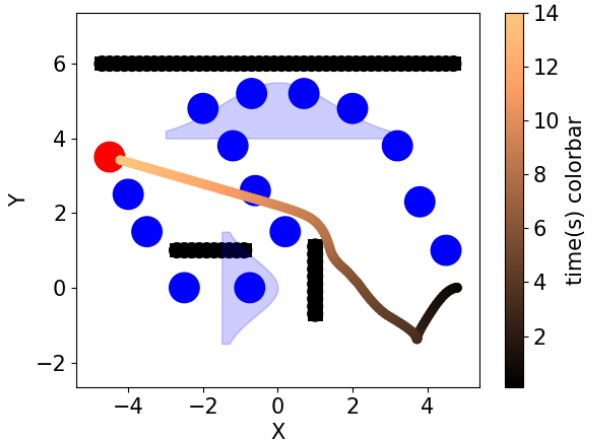
TABLE 6.2 shows results from simulations with 3 different sets of rewards for each waypoint. We evaluate the proposed algorithms with both the r_{lag} and $r_{chinneck}$ heuristic in solving **Objective 3**. Fig.6.3 shows the performance of the proposed algorithms using the rewards from Case 3 defined in TABLE 6.2.

Based on results shown in TABLE 6.2 and Fig.6.3, our proposed heuristic with the r_{lag} score performs better compared to Chinneck's algorithm with $r_{chinneck}$ score in terms of being able to reach more waypoints than Chinneck's method, at a lower computational effort. Both methods are also much faster than an exhaustive search over all possible combinations of constraints.

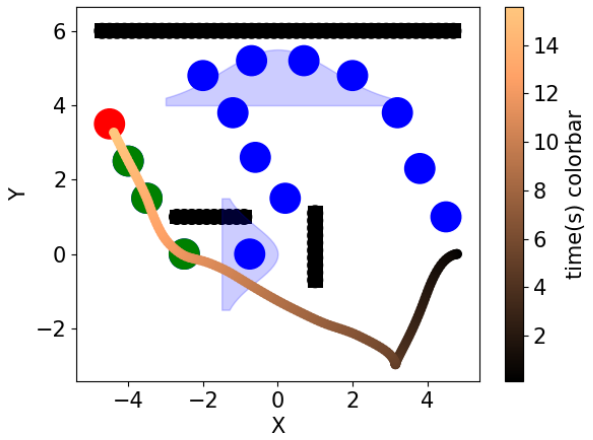
We observe that Lagrange variables are well suited to reason about the feasibility of optimization problems for a dynamical system. We believe that a possible reason for the good performance of our heuristic is that the optimization is more likely to become infeasible if it has more active constraints in preceding time steps. Since Lagrange multipliers are non-zero only for active constraints, they are a measure of "total activity". Further analysis is left for future work.



(a) Exhaustive CBF Search (Optimal)



(b) Algorithm 1 with $r_{chinneck}$



(c) Algorithm 1 with r_{lag}

Figure 6.3: Trajectories realized under different algorithms for 14 waypoints.

6.6 Conclusion

We developed heuristics for finding feasible sets of constraints by dropping soft constraints based on the Lagrange multipliers of each constraint over a time horizon. Evaluation of our algorithm is then done for a test case of robot waypoint following in the presence of obstacles and disturbances. Future work will involve the formal analysis of the convergence guarantees of the proposed heuristic. Finally, to improve the optimality of our solutions, we also plan to induce randomness into our decision-making by using approaches such as genetic algorithms whose fitness score could be made a function of our heuristic.

CHAPTER 7

Conclusion

7.1 Conclusions

Controllers exhibit degraded performance and safety violations when deployed in scenarios that were difficult to characterize in the manual or offline tuning phase. This dissertation advances the capabilities of robot controllers to adapt to complex, uncertain, and untested environments. Towards this, several contributions were made by formalizing the notion of compatibility of multiple constraints as well a performance index and then proposing algorithms that are amenable to online implementation.

Chapter 1 presented an overview of constrained control and planning literature that inspired our work. Chapter 2 presented a comprehensive review of tools and concepts that this dissertation employs - optimization, probability distributions and their representation, set invariance theory, and control barrier functions. Chapter 2 also presented a thorough review of CBF-based methods and their limitations. Importantly, the computational speed and memory limitations hamper the application of existing methods for finding valid CBFs even when they are applicable theoretically. Consequently, robots are deployed with invalid or candidate CBFs in their controllers. This necessitates the need for online adaptation to customize the response for specific scenarios.

Before delving into the design of adaptation laws, Chapter 3 first introduces the notion of compatibility of constraints. Several examples serve to empirically show that CBF-QP controllers become infeasible and the time to infeasibility can be sensitive to other modules of the navigation stack such as the nominal controller. Changing the nominal controller would result in drastic changes in how and when CBF-QP controller becomes infeasible. The volume of the feasible control space at a given state was then introduced as a metric to quantify the distance to infeasibility. The compatibility of multiple CBF constraints would then require the volume to be non-zero at all times. Therefore, Chapter 3 then proceeded with introducing a new CBF, called feasible space CBF (FS-CBF) whose purpose is to prevent

the volume from going to zero. Finding valid FS-CBF though remains a challenge and falls under the same curse of being limited by the scope of offline analysis. The importance of this chapter lies in using the feasible space volume itself as a candidate (invalid) FS-CBF which was shown empirically to improve the feasibility of CBF-QP controllers and reduce sensitivity to changes in nominal controller.

Towards the objective of forming CBF-based controllers that can adapt, Chapter 4 introduces the notion of Rate-Tunable CBFs. RT-CBFs allow the parameters of class-$\mathcal{K}$ function to vary with time thereby changing the aggressiveness of the controller. Specifically, we show that the existence of RT-CBF is necessary for safety by proving that every possible safe trajectory of the system accepts a corresponding parameter trajectory such that the CBF condition is satisfied. Two approaches were then proposed to design the parameter dynamics. The first algorithm is point-wise optimal (depends on the current state only) and steers the parameters toward their desired values while ensuring the solution to CBF-QP exists under some conditions. The second algorithm is a model-predictive algorithm that uses the sensitivity (gradients) of future states to the controller parameters to improve the performance. Specifically, it computes the gradients of a high-level cost function and the constraint function at each time step in an SQP-inspired online gradient-descent framework. The second algorithm although applied on CBF-QP, applies to any parametric controller.

Chapter 5 stems from the motivation to extend the model-predictive framework from Chapter 4 to systems with stochastic dynamics. The main contribution lies in introducing a numerical scheme for predicting future state distributions to relay to the chance constraints formulation. Existing approaches are mostly based on Monte Carlo or Gaussian approximation. The proposed scheme is instead based on Unscented Transform (UT). While some works employing UT do exist, they do not consider state-dependent uncertainties while propagating state distributions across a stochastic dynamics function. Chapter 5 introduced expansion-compression layers based on moment-matching as a scalable method to propagate distribution while keeping the number of sigma points fixed. Utilizing some of the existing UT algorithms in the expansion layer, the overall prediction scheme is shown to be composed of differentiable functions. Finally, applications are presented to gradient-based model predictive tuning framework as well as to MPPI controller.

Chapter 6, departing from the parameter adaptation rhetoric in previous chapters aimed at satisfying all the constraints, presented an approach to remove constraints when necessary. When the tasks are incompatible even in the ideal scenario or when the controller at hand is incapable of completing all the tasks, analogous to state and time-dependent constraints in this work, Chapter 6 shows that removing constraints enables the controller to satisfy higher priority tasks at the expense of lower-priority tasks. Already established as an NP-Hard

problem called maxFS, Chapter 6 revisits the problem in the context of dynamical systems where the optimization problem plays the role of controller. It proposes a new Lagrange multiplier-based heuristic to use past feasible optimizations to make an informed prior for current infeasible optimization. Finally, improved performance over a baseline adapted from a standard slack-based approach to solving the maxFS problem.

7.2 Future Work

While this work is expansive, there are still many avenues for improvement. Presented below is a high-level description of the most important research topics suitable for immediate extension of this dissertation.

7.2.1 Adaptation of CBFs

The work presented in this dissertation lacks strong convergence guarantees of the adaptation laws. The locally optimal method proposed in Section 4.3 still suffers from the so-called myopicness property and is unable to guarantee that the conditions under which it admits a solution will continue to hold. Similarly, the model predictive approach in Section 4.4 is based on a variation of gradient descent and is therefore prone to getting stuck in local minima and possibly a failure to converge to the safe solution on time. Strict guarantees for online optimization are difficult to achieve for nonlinear closed-loop systems without making strong assumptions. The author believes that a combination of offline and online approaches is best suited to have performance guarantees. Some existing research avenues of performing offline computations, although not directly applicable, have the potential to be paired with online approaches. Backup safe sets with corresponding backup controllers and controlled invariant sets for recursive feasibility of MPC controllers are two popular methods for ensuring safety in theory. Such sets, however, at least when static in time, may not exist in scenarios such as non-cooperative multi-robot systems and or underactuated dynamics with spatiotemporal disturbances. One solution would be to assume rules of engagement specific to scenarios. For example for multi-robot systems consisting of noncooperative but not adversarial robots, some amount of centralized planning and deconfliction analogous to the walking on the right side rule could help derive time-varying backup safe sets. The proposed online adaptation methods could then be modified for adapting controller parameters while ensuring that the robot never leaves the time-varying backup or controlled invariant set.

7.2.2 Predictions of state distributions

Chapter 5 relies on existing UT algorithms to perform the expansion operation. For the sake of efficient online implementation, two variations namely (vanilla) UT and GenUT are chosen for control applications owing to their admittance of analytical formulas. However, UT variants capable of preserving more of the higher-order moments involve solving a system of polynomial equations that does not admit analytical formulas in general. One extension would be to use clusters of UT points (based on simple UT algorithms like vanilla UT and GenUT) and expand each of them independently analogous to a mixture of the Gaussian model. Such an extension is also needed when the distributions are multi-modal (leading naturally to larger variation in higher-order moments), as UT and GenUT cannot model such distributions. This would also entail devising algorithms to compute the number of modes at a given time and assigning a set of UT points and weights to each mode.

Next, note that the ideal scenario portrayed in Theorem 6 in Section 5.3.1 would not be realized in practice. Therefore, another venue for improvement is in quantifying the error in the expansion and compression layer and propagating the errors along with sigma points.

Finally, state prediction under stochastic dynamics suffers from *uncertainty explosion*. The term means that the uncertainty in the state increases across each prediction step, eventually approaching infinity, thereby rendering long-horizon planning too conservative and useless. Covariance steering[164, 261] aims to design control inputs that control the spread of the state distribution besides just focusing on instability and safety objectives. It has been applied to MPPI and stochastic MPC controllers with linearized dynamics. The UT methods can help extend covariance steering to generic nonlinear systems. Additionally, due to the capability of preserving higher-order moments, UT or particle-based methods additionally open up the venue for tracking and controlling not just the covariance but also the higher-order moments of the distribution.

Finally, depending on the source of uncertainty, it may be possible to avoid uncertainty explosion by accounting for the dynamics of uncertainty. Consider, for example, a social navigation scenario in which an estimation module predicts human motion by learning from or interpolating observed human positions. As the robot navigates, the estimator receives new observations and improves the prediction. Another example is the Kalman filter that uses observations w.r.t a static obstacle to improve the robot's state estimate. Therefore a prediction scheme accounting for estimator dynamics would exhibit a slower uncertainty explosion. This idea was also exploited in [46] wherein the authors incorporated Kalman Filter dynamics in sampling-based LQR tree planner to distinguish between paths based on the robot localization accuracy in the future.

7.2.3 Constraint Selection

Chapter 6 addressed the problem of removing a minimum number of constraints rendering an infeasible optimization problem feasible given the (additive) priority of constraints. The existing methods based on the slack variable as well as the proposed method based on Lagrange multiplier are heuristics without any proven optimality or suboptimality guarantees. Given that these problems are NP-Hard, it is impractical to expect that these polynomial time algorithms will always converge close to the optimal solution. And even if they did, a real-time. Moreover, in practice, these algorithms can only be run for a small number of iterations dependent on the computational speed. Therefore, there is a need for quantifying the optimality gap of these methods as a function of iteration number. These error bounds can then be incorporated to avoid making decision controllers work under the worst-case scenario.

A common framework to derive provable error bounds for solving NP-Hard problems in submodularity[122]. When the objective function or the constraint function in an optimization problem is submodular, greedy algorithms have been shown to have a bounded optimality gap. An interesting direction would be to derive new heuristics that are submodular. Finally, the proposed heuristics were also tested only on CBF-QP controllers. Future work will involve evaluation on different types of controllers and for different applications.

APPENDIX A

RT-CBF proofs

A.1 Proof of analytical solution of QP for Theorem 3

Here we provide a derivation of the CBF-QP controller (4.5). The procedure is similar to the one in [15]. Consider the following CBF-QP:

$$\min_{u} \quad \|u - u_r(x)\|^2 \tag{A.1a}$$

$$\text{s.t.} \quad L_f h(x) + L_g h(x) u \geq -\alpha(h(x)). \tag{A.1b}$$

The above can be written in generic form as

$$\min_{u} \quad \frac{1}{2} u^T Q u + G(x)^T u \tag{A.2a}$$

$$\text{s.t.} \quad A(x)u \geq b(x), \tag{A.2b}$$

where $Q \in S_m, G \in \mathbb{R}^{m\times 1}, A \in \mathbb{R}^{1\times m}, b \in \mathbb{R}$ and $A = L_g h, b = -\alpha(h) - L_f h, Q = 2I, G = -2u_r$. The solution of the above QP can be found analytically and we outline the procedure here. Substituting $u = (\sqrt{Q})^{-1}u' + \bar{u}$ in (A.2), where $\bar{u} = -Q^{-1}G$ and $\bar{u}$ is the point that minimizes the objective (A.2a), we can reduce (A.2) to the following minimum-norm optimization problem:

$$\min_{u'} \quad \|u'\|^2 \tag{A.3a}$$

$$\text{s.t.} \quad A\sqrt{Q}^{-1}u' \geq b + AQ^{-1}G. \tag{A.3b}$$

The solution to (A.3) is given as:

$$u' = \begin{cases} \frac{\sqrt{Q}^{-1}A^T}{\|A\sqrt{Q}^{-1}\|^2}(b + AQ^{-1}G) & \text{if } b + AQ^{-1}G > 0 \\ 0 & \text{if } b + AQ^{-1}G \leq 0. \end{cases} \tag{A.4}$$

And therefore the solution u of (A.2) is given as:

$$u = \begin{cases} Q^{-1}\left[A^T \frac{b+AQ^{-1}G}{||A\sqrt{Q^{-1}}||^2} - G\right] & \text{if } b + AQ^{-1}G > 0 \\ -Q^{-1}G & \text{if } b + AQ^{-1}G \leq 0. \end{cases} \tag{A.5}$$

Substituting for $A = L_g h, b = -\alpha(h) - L_f h, Q = 2I, G = -2u_r$, we get that the solution to (A.1) is:

$$u_{QP} = \begin{cases} u_r - \frac{L_g h^T}{||L_g h||^2}(L_f h + L_g h u_r + \alpha(h)) & \text{if } \psi^1 < 0 \\ u_r & \text{if } \psi^1 \geq 0, \end{cases} \tag{A.6}$$

where $\psi^1 = L_f h + L_g h u_r + \alpha(h)$.

A.2 A Synthesis Case Study: Designing RT-CBFs For Multi-Robot Systems

In this section, as an application case study, we utilize RT-CBFs and propose class-$\mathcal{K}$ function parameter dynamics for decentralized multi-robot control based on our work in [173]. We show through simulations that our online parameter adaptation scheme leads to a less conservative response by quantifying the ease of satisfaction of CBF constraints.

We consider a multi-robot system with N agents. The state of each agent $l \in \{1, .., N\}$ is denoted as $x_l \in \mathbb{R}^n$ and follows the dynamics

$$\dot{x}_l = f_l(x_l) + g_l(x_l)u_l, \tag{A.7}$$

where $f_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}, g_l : \mathbb{R}^{n_l} \to \mathbb{R}^{m_l}$ are locally Lipschitz continuous functions and $u_l \in \mathcal{U}_l \subset \mathbb{R}^{m_l}$ is the control input of each agent l. Note that in general the agents can be of heterogeneous dynamics, i.e., $f_p \neq f_q$, $g_p \neq g_q$ for $p \neq q$.

The set of neighbors of agent l is denoted as $\mathcal{N}_l = \{1, .., N\} \setminus \{l\}$. We consider a decentralized scenario where each agent decides its own control input under the following assumption:

Assumption 4. Each agent l has perfect observations of the state x_o and its derivative $\dot{x}_o$ of every other agent $o \in \mathcal{N}_l$.

We consider pairwise safety constraints (i.e., constraints affecting a pair of two agents i, j) denoted $h_{ij} : \mathbb{R}^{n_i} \times \mathbb{R}^{n_j} \to \mathbb{R}$, so that their zero-superlevel sets denoted $\mathcal{S}_{ij} = \{x_i, x_j \mid h_{ij}(x_i, x_j) \geq 0\}$ define sets of safe states such that $\partial\mathcal{S}_{ij} = \{x_i, x_j \mid h_{ij}(x_i, x_j) = 0\}$

and $\text{Int}(\mathcal{S}_{ij}) = \{x_i, x_j \mid h_{ij}(x_i, x_j) > 0\}$, and their time derivatives along the system dynamics (A.7) read:

$$\dot{h}_{ij}(x_i, x_j) = \frac{\partial h_{ij}}{\partial x_i}\ \dot{x}_i + \frac{\partial h_{ij}}{\partial x_j}\ \dot{x}_j. \tag{A.8}$$

We note that for $(x_i, x_j) \in \partial\mathcal{S}_{ij}$, i.e., when $h_{ij}(x_i, x_j) = 0$, the sign of (A.8) should be non-negative for the value of the function h_{ij} to not decrease further along the trajectories (A.7), i.e., for the trajectories to remain within $\mathcal{S}_{ij}$. Based on this, we say that at a given time and states (t, x_i, x_j), an agent j has one of the following behaviors towards agent i:

- Cooperative: When the dynamics of agent j is such that $\frac{\partial h_{ij}}{\partial x_j}\dot{x}_j \geq 0$
- Noncooperative: When the dynamics of agent j is such that $\frac{\partial h_{ij}}{\partial x_j}\dot{x}_j < 0$.

We clarify that the definitions above do not aim to classify intentions (e.g., malicious vs non-malicious) of the agent j w.r.t. agent i. For example, in the presence of multiple agents, a possibly non-malicious agent j may still be perceived as noncooperative by agent i if agent j is moving towards agent i under the influence of another agent k that is moving towards j. Our objective is not to classify whether agent j is malicious or not, but rather to tune the controller of the agent i so that it responds safely and accordingly to the different types of neighbor agents, and the type of motion exhibited by each one of them. For example, a fast (respectively, slow) adversarial agent j will ideally warrant more (respectively, less) conservative response from the agent i. In the sequel we often refer to the agent i as an ego agent, and a neighbor agent j as a neighbor agent.

A dynamic environment with moving agents implies that the world is inherently time-varying from the perspective of an agent i. We propose an online adaptation scheme to shape the response of the ego agent i so that it becomes less or more conservative w.r.t. its neighbor agents j based on observations of their states and their derivatives. To this end we develop a *trust metric* based on instantaneous observations that is used to modify $\theta^k_{\alpha_i j}$.

A.2.0.1 Design of Trust Metric

Consider the following CBF constraint

$$\begin{aligned} \dot{\psi}^{k-1}_{ij} &\geq -\alpha^k_{ij}(\psi^{k-1}_{ij}), \\ \implies \frac{\partial \psi^{k-1}_{ij}}{\partial x_i}\dot{x}_i + \frac{\partial \psi^{k-1}_{ij}}{\partial x_j}\dot{x}_j + \frac{\psi^{k-1}_{ij}}{\partial \Theta_{ij}}\dot{\Theta}_{ij} &\geq -\alpha^{k-1}_{ij}(\psi^k_{ij}), \end{aligned} \tag{A.9}$$

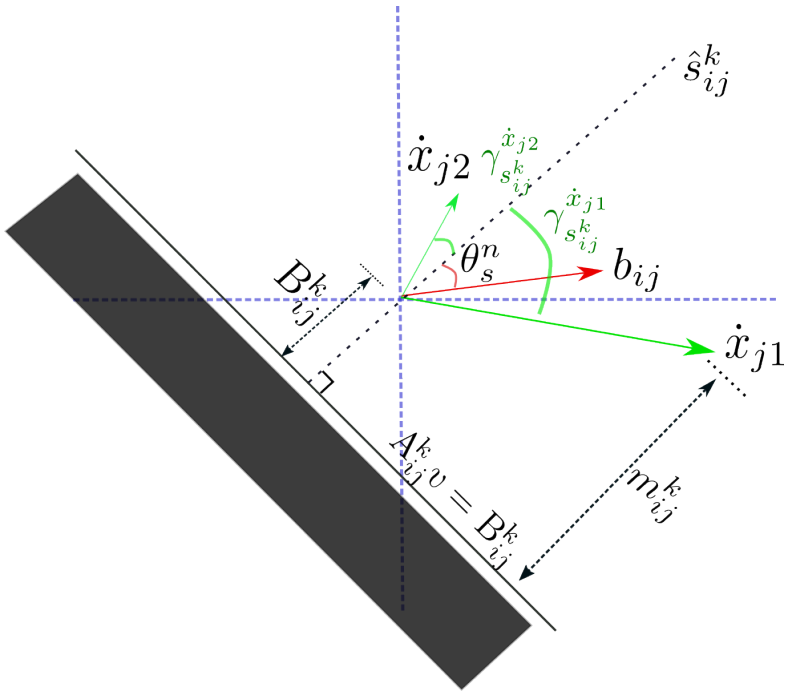


Figure A.1: Illustration of the halfspaces induced by the constraint function. $Av \geq b$ represents the halfspace described by (A.11) with $A = \frac{\partial h_{ij}}{\partial x_j}$, $v = \dot{x}_j$, and $b = -\alpha h - \max\left\{\frac{\partial h_{ij}}{\partial x_i}\dot{x}_i\right\}$. The two x_j are possible instances of the actual motion of agent j. $\hat{s}$ is the normal to the hyperplane $Av = b$. $\hat{n}_j$ is the nominal direction of motion for agent j.

where note that although $\dot{\Theta}_{ij}$ appears in (A.9), the $r_{ij}-k+1$-order derivative to be designed will not contribute in the equation (see (4.3) for example). The margin by which the CBF condition (A.9) is satisfied, i.e., the value of $\dot{\psi}_{ij}^{k-1} + \alpha_{ij}^{k-1}(\psi_{ij}^{k-1})$ represents of the ease of satisfaction of constraint w.r.t agent i. The best case action by agent i thus corresponds to the largest possible margin for a given motion $\dot{x}_j$ of agent j.

Constraint margin based trust score: The motions of agent j that lead to a feasible solution to (A.9) are considered as *safe* by i. Consider the following decomposition

$$\dot{\psi}_{ij}^{k-1} = \frac{\partial \psi_{ij}^{k-1}}{\partial x_i}\dot{x}_i + \frac{\partial \psi_{ij}^{k-1}}{\partial x_j}\dot{x}_j + \frac{\partial \psi_{ij}^{k-1}}{\partial \alpha_{ij}^k}\dot{\alpha}_{ij}^k \tag{A.10}$$

Here, we have ignored the higher order derivatives of $\alpha_{ij}^m, m < k-1$. The safe motions of j

are thus given by

$$\begin{aligned}\frac{\partial \psi_{ij}^{k-1}}{\partial x_j}\dot{x}_j &\geq -\alpha_{ij}^k(\psi_{ij}^{k-1}) - \frac{\partial \psi_{ij}^{k-1}}{\partial \alpha_{ij}^k}\dot{\alpha}_{ij}^k - \frac{\partial \psi_{ij}^{k-1}}{\partial x_i}\dot{x}_i \\ &\geq -\alpha_{ij}^k(\psi_{ij}^{k-1}) - \frac{\partial \psi_{ij}^{k-1}}{\partial \alpha_{ij}^k}\dot{\alpha}_{ij}^k \\ &\quad - \max_{u_i \in \mathcal{U}} \left\{ \frac{\partial \psi_{ij}^{k-1}}{\partial x_i}\dot{x}_i \right\}, \end{aligned} \tag{A.11}$$

where the last term in (A.11) can be obtained from the following Linear Program (LP):

$$\max_{u_i} \quad \frac{\partial \psi_{ij}^k}{\partial x_i}\dot{x}_i \tag{A.12a}$$

$$\text{s.t.} \quad \dot{\psi}_{il}^k \geq -\alpha_{il}\psi_{il}^k, \quad \forall l \in N_i \setminus j. \tag{A.12b}$$

The inequality (A.11) gives a lower bound on motions of agent j, a violation of which will lead to the infeasibility of QP controller of agent i. The design of the trust factor is thus based on a comparison of the contribution of agent j with the aforementioned lower bound.

The evaluation of (A.11) as equality represents a halfspace whose separating hyperplane has the normal direction $\hat{s}_{ij}^k = \frac{\partial \psi_{ij}^k}{\partial x_j}$, and has been visualized in Fig. A.1. Let the half-space in (A.11) be represented in the form $A_{ij}^k v \geq B_{ij}^k$ with $v = \dot{x}_j$ and A_{ij}^k, B_{ij}^k defined accordingly. Let m_{ij}^k be the constraint margin by which (A.11) is satisfied

$$m_{ij}^k = A_{ij}^k v + B_{ij}^k, \tag{A.13}$$

where $m_{ij}^k < 0$ is an incompatible vector, corresponding to scenarios that should never happen. The larger the value of m_{ij}^k, the greater the ease of satisfaction for agent i. Hence, a larger m_{ij}^k allows room for relaxing the constraint. Therefore, the constraint margin-based trust score is designed as

$$\rho_{m_{ij}^k} = f_d(m_{ij}^k), k \in \{1, .., r\}, \tag{A.14}$$

where $f_d : \mathbb{R}^+ \to [0, 1]$ is a monotonically-increasing, Lipschitz continuous function, such that $f_d(0) = 0$. A possible choice would be $f_d(d) = \tanh(\beta d)$, with β being a scaling parameter.

Belief based trust score: We now describe a method to incorporate i's prior belief of the motion of agent j. Note that infinitely many values of $\dot{x}_j$ may satisfy the CBF derivative

condition (A.11) with the same constraint margin (all $\dot{x}_j$ lying on the yellow hyperplane in Fig. A.1 will have the same m_{ij}^k). This gives us an additional degree of freedom for deciding which motions are more trustworthy. Consider the scenario shown in Fig.A.2 where an agent j_1 is moving directly towards i with speed 2 and another agent moving at an angle ϕ w.r.t the vector $x_i - x_j$ but with speed $2/\cos 45°$. Both j_1 and j_2 would thus lead to the same constraint margin but we would like to trust j_2 more if we had a belief that it is supposed to be moving in that direction or because it is not moving straight towards i. We denote this nominal belief direction of agent j w.r.t agent i as b_{ij}. If $\dot{x}_j$ performs worse than the nominal direction, then it is a cause of concern as it is not consistent with the $i's$ belief for agent j. Note that b_{ij} need not be the true direction of motion of agent j for our algorithm to work since if b_{ij} is not known, we can consider the worst-case scenario where $b_{ij} = x_i - x_j$, i.e., that j is an adversary and intends to collide with agent i; over time, our algorithm will learn that j is not moving along b_{ij} and therefore increases $i's$ trust of j. Knowledge of nominal direction plays an important role in shaping belief as it helps to distinguish between uncooperative and adversarial agents.

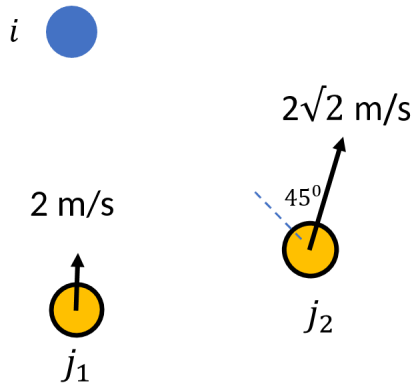


Figure A.2: Agents j_1 and j_2 contribute the same constraint margin for satisfying the CBF derivative condition (A.9).

Suppose the angle between the vectors b_{ij} and $\hat{s}_{ij}^k$ is denoted by $\gamma_{s_{ij}}^{b_{ij}}$, and the angle between $\dot{x}_j$ and $\hat{s}_{ij}$ is denoted by $\gamma_{s_{ij}^k}^{\dot{x}_j}$. The belief-based trust is designed as

$$\rho_{\gamma_{ij}} = f_\gamma(\gamma_{s_{ij}^k}^{b_{ij}} / \gamma_{s_{ij}^k}^{\dot{x}_j}), \tag{A.15}$$

where $f_\gamma : \mathbb{R}^+ \to [0, 1]$ is again a monotonically-increasing Lipschitz continuous function with $f_\gamma(0) = 0$. Note that even if $\gamma_{s_{ij}^k}^{b_{ij}} = \gamma_{s_{ij}^k}^{b_{ij}}$, i.e., j is perfectly following its nominal direction, we do not design f_γ to be 1 as the robot might be uncooperative. We give f_γ higher values when $\gamma_{s_{ij}^k}^{b_{ij}} < \gamma_{s_{ij}^k}^{b_{ij}}$, as with a_{j_2} in Fig. A.1, j seems to be compromising its

nominal movement direction for improved safety, thus leading to a higher score. Finally, when $b_{ij_{s_{ij}}} < \gamma_{s_{ij}}^{b_{ij}}$, as with a_{j_1} in Fig. A.1, then j's current motion is performing worse for inter-robot safety than its nominal motion and therefore has lower trust than the latter case.

Final Trust Score: The trust metric is now designed based on $\rho_{m_{ij}^k}$ and $\rho_{\gamma_{ij}^k}$. Let $\bar{\rho}_{m_{ij}^k} \in (0, 1)$ be the desired minimum robustness/margin in satisfying the CBF condition (A.11). Then, the trust metric $\rho_{ij} \in [-1, 1]$ is designed as follows:

$$\rho_{ij}^k = \begin{cases} (\rho_{m_{ij}^k} - \bar{\rho}_{m_{ij}^k})\rho_{\gamma_{ij}^k}, & \text{if } \rho_{m_{ij}^k} \geq \bar{\rho}_m, \\ (\rho_{m_{ij}^k} - \bar{\rho}_m)(1 - \rho_{\gamma_{ij}^k}), & \text{if } \rho_{m_{ij}^k} < \bar{\rho}_m. \end{cases} \tag{A.16}$$

Here, $\rho_{ij}^k = 1$ and -1 represent the absolute belief in another agent being cooperative and noncooperative, respectively. If $\rho_{m_{ij}^k} > \bar{\rho}_m$, then we would like to have $\rho_{ij}^k > 0$, and its magnitude is scaled by $\rho_{\gamma i j^k}$ with smaller values of $\rho_{\theta_{ij}^k}$ conveying low trust. Whereas, if $\rho_{m_{ij}^k} < \bar{\rho}_m$, then we would like the trust factor to be negative. A smaller $\rho_{\gamma_{ij}^k}$, in this case, implies more distrust and should make the magnitude larger, hence the term $1 - \rho_{\gamma_{ij}^k}$. The trust ρ is now used to adapt α with following equation

$$\dot{\alpha}_{ij}^k = f_\alpha(\rho_{ij}^k), \tag{A.17}$$

where $f_\alpha : [-1, 1] \to \mathbb{R}$ is a monotonically increasing function. A positive value of $f_{\alpha_{ij}^k}$ relaxes the CBF condition by increasing α_{ij}^k, and a negative value decreases α_{ij}^k. Note that (A.11) needs $\dot{\alpha}_{ij}^{k-1}$. Therefore, we first design $\dot{\alpha}_{ij}^1$ and then use it in the design of $\dot{\alpha}_{ij}^2$ and repeat this procedure for higher order derivatives. This is similar to the cascaded control structures.

A.2.0.2 Projection to Feasible Parameter Space

To ensure the feasibility of the QP, the heuristically designed parameter update rules are passed as the desired derivatives in Algorithm 1 of Section 4.3. However, since this approach requires projecting to one time step ahead future state, it requires knowledge of the closed-loop dynamics of the multi-agent system which is unknown to individual agents. We therefore implement a robust version of our Algorithm 1 based on the following assumption

Assumption 5. The closed-loop of every agent has a bounded change every sampling period, that is, $||\dot{x}_i(t) - \dot{x}_j(t - \Delta t)|| \leq \Delta F$. This bound is known to the other agent.

Assumption 5 would imply, for example, that if agent i observes the other agent to be moving at $2m/s$ at time $t - \Delta t$, then the actual velocity of j at time t would be $2 \pm \Delta v$, where

Δv is a known constant. This assumption leads to less conservative results than assuming a worst-case bound on dynamics and does not require continuity of controllers. This bound is used in Lines 5-11 of Algorithm 1 to update the parameters of class-$\mathcal{K}$ function.

APPENDIX B

FORESEE

B.1 Approximating chance constraints with robust constraints

The analysis is similar to ones presented in [26, 158]. In our case, the chance constraint (5.14c) is of the form

$$P\Big[\underbrace{\begin{bmatrix} a^T & b \end{bmatrix}}_{\alpha^T} \begin{bmatrix} d \\ 1 \end{bmatrix} \geq 0\Big] \geq 1 - \epsilon, \tag{B.1}$$

where $a^T = \nabla_\theta c_i(x_{t+\tau})$, $b = c_i(x_{t+\tau})$, $\alpha = \begin{bmatrix} a^T \\ b \end{bmatrix} \in \mathbb{R}^{\kappa+1}$. We assume that α can be written as a linear parametrization $\alpha = \alpha_0 + \sum_{l=1}^{L} \alpha_l \zeta_l$, where ζ_l are perturbation variables. Now, in the case that ζ_l are known to be independent, zero mean, random variables varying in segments $[-\sigma_l, \sigma_l]$, then (B.1) can be safely approximated by the conic quadratic inequality

$$w_0 - \sqrt{2 \ln \frac{1}{\epsilon}} \sqrt{\sum_{l=1}^{L} {\sigma_l}^2 {w_l}^2} \geq 0. \tag{B.2}$$

where by definition $\alpha^T \begin{bmatrix} d \\ 1 \end{bmatrix}$ is rewritten for compactness as:

$$\left(\alpha_0 + \sum_{l=1}^{L} \zeta_l \alpha_l\right)^T \begin{bmatrix} d \\ 1 \end{bmatrix} \equiv w_0[d] + \sum_{l=1}^{L} \zeta_l w_l[d], \tag{B.3}$$

where $w_l[d]$, $l \in \{0, \ldots, L\}$, are affine functions of d. Hence, obtaining w_l provides a safe, approximate constraint (safe in the sense that the solution of the approximate problem is

guaranteed to be a solution of the original problem as well).
To this end, we note that in our case we have:

$$\alpha = \begin{bmatrix} \nabla_\theta(c_i(x_{t+\tau})) \\ c_i(x_{t+\tau}) \end{bmatrix} = \begin{bmatrix} \nabla_{\theta_1}(c_i(x_{t+\tau})) \\ \vdots \\ \nabla_{\theta_\kappa}(c_i(x_{t+\tau})) \\ c_i(x_{t+\tau}) \end{bmatrix} \tag{B.4}$$

and that we can consider the Taylor series first-order approximation of each element of the vector α around the mean value $\mathbb{E}[x_{t+\tau}]$ of $x_{t+\tau}$, denoted for compactness $\hat{x}_{t+\tau}$. For example: $c_i(x_{t+\tau}) = c_i(\hat{x}_{t+\tau}) + \frac{\partial c_i(x_{t+\tau})}{\partial x_{t+\tau}}\big|_{\hat{x}_{t+\tau}} \tilde{x}_{t+\tau}$, where $\tilde{x}_{t+\tau} \triangleq x_{t+\tau} - \hat{x}_{t+\tau}$. For compactness, we drop the subscript $t+\tau$ and denote $\nabla_{\theta_\kappa}(c_i(\cdot)) = g_\kappa(\cdot)$, so that

$$\alpha = \begin{bmatrix} g_1(x) \\ \vdots \\ g_\kappa(x) \\ c_i(x) \end{bmatrix} = \begin{bmatrix} g_1(\hat{x}) + \frac{\partial g_1(x)}{\partial x}\big|_{\hat{x}} \tilde{x} \\ \vdots \\ g_\kappa(\hat{x}) + \frac{\partial g_\kappa(x)}{\partial x}\big|_{\hat{x}} \tilde{x} \\ c_i(\hat{x}) + \frac{\partial c_i(x)}{\partial x}\big|_{\hat{x}} \tilde{x} \end{bmatrix}, \tag{B.5}$$

which further yields

$$\begin{aligned} \alpha^T \begin{bmatrix} d \\ 1 \end{bmatrix} &= c_i(\hat{x}) + \underbrace{\frac{\partial c_i(x)}{\partial x}\big|_{\hat{x}}}_{\left(\nabla_x c_i(x)\big|_{\hat{x}}\right)^T} \tilde{x} + \\ &+ \left(\underbrace{g_1(\hat{x})}_{\nabla_{\theta_1} c_i(\hat{x})} + \underbrace{\frac{\partial g_1(x)}{\partial x}\big|_{\hat{x}}}_{(\nabla_x(\nabla_{\theta_1} c_i(x))\big|_{\hat{x}})^T} \tilde{x} \right) d_1 + \cdots + \\ &+ \left(\underbrace{g_\kappa(\hat{x})}_{\nabla_{\theta_\kappa} c_i(\hat{x})} + \underbrace{\frac{\partial g_\kappa(x)}{\partial x}\big|_{\hat{x}}}_{(\nabla_x(\nabla_{\theta_\kappa} c_i(x))\big|_{\hat{x}})^T} \tilde{x} \right) d_\kappa = \\ &= c_i(\hat{x}) + (\nabla_x c_i(x)\big|_{\hat{x}})^T \tilde{x} + \\ &+ d^T \nabla_\theta(c_i(\hat{x})) + d^T \left((\nabla_x(\nabla_\theta c_i(x))\big|_{\hat{x}})^T \tilde{x} \right), \end{aligned} \tag{B.6}$$

where $w_0 = c_i(\hat{x}) + d^T \nabla_\theta(c_i(\hat{x}))$, and the remaining terms correspond to the right-hand side of (B.3) for $L = 1$, with $\tilde{x}$ being a zero-mean vector (assuming an unbiased estimate $\hat{x}$ of x),

which is confined within intervals $[-\sigma, \sigma]$ known by the quality of the estimation (e.g., using the covariance of the UT-based estimate in the earlier section). Hence, not surprisingly, what (B.6) and (B.2) reveal is that the constraint (5.14c) can be approximated by a deterministic constraint that captures (and aims to counteract) the sources that contribute to possible violation of the constraint, namely the estimation error (via σ) and the behavior of the function around the current value via $\nabla_x(c_i(\cdot)), \nabla_x(\nabla_\theta(c_i(\cdot)))$. In practice it might be difficult to know bounds a priori for $\nabla_x(c_i(\cdot)), \nabla_x(\nabla_\theta(c_i(\cdot)))$ at each point $x \in \mathbb{R}^n$. However, local or global estimates of the Lipschitz constants of the corresponding functions might be available, and in that case they can provide a good estimate of the bounds. In summary, the chance constraint (5.14c) can be approximated as:

$$c_i(\mathbb{E}[x_{t+\tau}]) + d^T \nabla_\theta(c_i(\mathbb{E}[x_{t+\tau}]) + \sqrt{2 \ln \frac{1}{\epsilon}} \Omega \geq 0, \tag{B.7}$$

where $\Omega = \omega(\Sigma[x_{t+\tau}], \nabla_x(c_i(\cdot)), \nabla_x(\nabla_\theta(c_i(\cdot))))$, $\omega : \mathbb{R}^n \to \mathbb{R}$ is a function that captures the statistics of the state, and the behavior of the constraint functions around the estimates of the state and the parameter.

B.2 Approximating the chance constraint

We note that it is computationally inefficient for the purpose of online implementation to evaluate the gradients $\nabla_x(c_i(\cdot)), \nabla_x(\nabla_\theta(c_i(\cdot))))$ involved in (B.7). Therefore, in the implementation of case studies 3 and 4, which involve the tuning of the controller parameters, we consider the constraint:

$$CF_i(x_{t+\tau}, \epsilon) + d^T \nabla_\theta(CF_i(x_{t+\tau}, \epsilon)) \geq 0, \tag{B.8}$$

where the confidence function $CF : \mathbb{R}^n \times [0, 1] \to \mathbb{R}$ is such that $CF_i(x, \epsilon) \geq 0 \implies P[c_i(x) \geq 0] \leq 1 - \epsilon$. Eq. (B.8) could be an exact robust formulation of the chance constraint (5.5b), or a conservative approximation. For example, supposing $c(x)$ is Gaussian, one choice of CF is as follows

$$CF(x, \epsilon) = \mathbb{E}[c_i(x)] - \beta(\epsilon)\sqrt{\Sigma[c_i(x)]}, \tag{B.9}$$

where $\beta : [0, 1] \to \mathbb{R}^+$ relates the probability ϵ to a factor of standard deviation that defines the confidence interval of Gaussian in (B.9). This factor is looked up in a table, e.g., for $\epsilon = 0.95$, $\beta(\epsilon) = 1.96$. Eq. (B.9) is based on the confidence interval of a Gaussian situated

symmetrically about the mean, which is more conservative than a confidence interval that includes the tail and exactly solves (5.5b). Our results are based on (B.9) due to the lack of analytical formulas for confidence intervals accounting for higher order moments of a distribution.

B.3 Data Collection and Wind Disturbance Learning

In this section, we detail our procedure to learn force F_{wind} exerted by wind by collecting quadrotor trajectory data in an offline phase. We move the quadrotor autonomously in several circular and "figure-8" shaped trajectories using the geometric controller with gains $k'_x = 14, k'_v = 7$. k'_x, k'_v were obtained by manually tuning the controller for precise trajectory following in the absence of fan (wind flow).

The circular and figure-8 trajectories follow the following references: $p_{ref}^{circle}(t) = \begin{bmatrix} x_0 + r\cos(\omega(t-t_0)) \\ y_0 + r\sin(\omega(t-t_0)) \end{bmatrix}$, $p_{ref}^{fig8}(t) = \begin{bmatrix} x_0 + r\sin(\omega(t-t_0)) \\ y_0 + r\sin(\omega(t-t_0))\cos(\omega(t-t_0)) \end{bmatrix}$, where x_0, y_0 is the center, and r, ω are the radius and angular velocity used in design of the references. The parameters of circular and figure-8 trajectories that were used for data collection are given in Table 1.

Shape	x_0 [m]	y_0 [m]	r [m]	ω [rad/s]
Circle	0.0	0.0	0.4	1.0
Circle	0.2	0.0	0.4	1.0
Circle	0.6	0.0	0.4	1.0
Circle	0.6	0.0	0.6	1.0
Circle	0.6	0.0	0.6	1.5
Figure of 8	0.0	0.0	0.4	1.0
Figure of 8	0.2	0.0	0.6	1.0
Figure of 8	0.2	0.0	0.6	1.5
Figure of 8	0.4	0.0	0.6	1.5
Figure of 8	0.6	0.0	0.6	1.0

Table B.1: Trajectory parameters for data collection

We record the actual and reference position p, p_{ref}, velocity $\dot{p}, \dot{p}_{ref}$ and acceleration $\ddot{p}, \ddot{p}_{ref}$ data of the quadrotor (outputted by EKF running onboard the Pixhawk board).

B.3.1 Raw data filtering

The acceleration data was extremely noisy and required some data pre-processing. Given that the drone was directed to fly at an angular frequency of about 1 rad/s, the primary

frequency component of the sinusoidal trajectory signal was approximately 0.32 Hz. Consequently, the Nyquist sampling frequency is determined to be 0.64 Hz. The Robot Operating System (ROS) messages containing the coordinates of the drone are published at a frequency of 75 Hz, thereby satisfying the Nyquist Sampling Theorem and ensuring that no aliasing occurs in the signal. We subsequently performed Fourier Transformation and analyzed the frequency component of recorded data, we gained some key insights into the signal in frequency domain. The highest peak of the Fourier domain plot lies in the range of 0 and 1 while high frequency noise is present, which is expected. We then devised an algorithm that finds the peak of the frequency domain signal from all the frequency components below 1, which would be the true frequency of the signal, and compute the cutoff frequency accordingly. To avoid distortion of the signal, we selected Butterworth filter which features a flat passband and smooth roll-off. We also selected a suitable passband to ensure minimal attenuation in the passband and avoid exorbitant group delay.

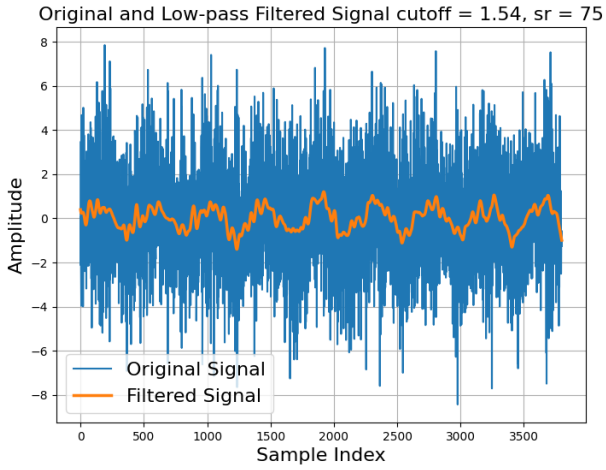


Figure B.1: The original and filtered signal

B.3.2 Gaussian Process Fit

After recording and filtering the data, we compute F_{wind} using (5.26) and (5.27) as follows $F_{wind}(p, \dot{p}) = m\ddot{p} - mg - F_T$.

Next, we use GPJax [179] library to fit a GP that takes as input the position and velocity of the quadrotor and outputs the acceleration due to wind disturbance. We employ the following kernel: kernel = Periodic * Rational Quadratic + RBF + White. In total we have about 4000 data points. Since the GP prediction on test data has complexity proportional to the size of training dataset, we train the GP with all the 4000 points for optimizing kernel

hyperparameters but only employ 500 randomly selected training points during prediction phase. For online optimization in FORESEE, we further speed up GP prediction by using Sparse GP implementation from GPJax.

APPENDIX C

Degradation of Constraints

C.1 Degradation of Constraints

C.1.1 Dynamics Definition

C.1.1.1 Single Integrator Dynamics

Let state and control inputs: $x = \begin{bmatrix} p_x & p_y \end{bmatrix}^T$, $u = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$, the dynamics is: $x_{t+1} = x_t + u_t\, dt$, with dt being the discrete time step.

C.1.1.2 Dynamic Unicycle Dynamics

For the Dynamic Unicycle dynamics used, we define state and control inputs: $x = \begin{bmatrix} x & y & v & \theta \end{bmatrix}^T$, $u = \begin{bmatrix} a & \omega \end{bmatrix}^T$, and their dynamics as $x_{t+1} = x_t + \begin{bmatrix} v_t \cos(\theta_t) & v_t \sin(\theta_t) & a_t & \omega_t \end{bmatrix}^T dt$, where dt is the discrete time step.

C.1.2 Reachability based Solution

In the waypoint tracking scenarios discussed in section 6.5, a reachability-based solution is employed. Let the backward reachable set $\mathcal{BRS}(p_{tf})$ of final target waypoint p_{tf} under dynamics (6.3b) is given by

$$\begin{aligned} \mathcal{BR}_{p_{tf}}(t) = \{x \mid\ & x_{0|t} = x,\ \exists u_{\tau|t}, \\ & \text{s.t. } H_i(t+\tau, x_{\tau|t}) \geq 0, \forall \tau \in \{0, 1, .., T\}\} \end{aligned} \tag{C.1}$$

Also define the unsafe backward reachable set $\mathcal{UBR}_i(t)$ under dynamics (6.3b) of each unsafe region as:

$$\begin{aligned}\mathcal{UBR}_i(t) = \{x \mid x_{0|t} = x, \forall u_{\tau|t},\ \exists \tau_{unsafe},\\ \text{s.t. } H_i(x_{\tau_{unsafe}|t}, \tau_{unsafe}|t) < 0\}\end{aligned}$$

This computation is amenable to offline computation only, even for simple systems [23]. At time t and state x_t, the robot chooses to go the next waypoint j' if there exists a path $p(t)$ going to the waypoint j' with $p(0) = x_t$ such that

$$p(t) \in \mathcal{BRS}(p_{tf}) \wedge p(t) \notin \mathcal{UBR}_{Hi}(t)\ \forall t > 0. \tag{C.2}$$

To implement the above decision-making process, we first discretize the state and control input space into a grid and make a graph connecting grid cells such that each edge represents a feasible transition between cells in one-time step. Subsequently, Dijkstra's algorithm is used for each waypoint combination to ascertain the existence of a safe path, enabling the timely reaching of each waypoint and the target position, with path length verifying the time requirements for reaching each waypoint.

BIBLIOGRAPHY

[1] https://docs.px4.io/main/en/flight_controller/pixhawk6c.html.

[2] https://github.com/aws-robotics/aws-robomaker-hospital-world.

[3] https://github.com/robotics-upo/gazebo_sfm_plugin.

[4] https://pypi.org/project/socialforce/.

[5] https://volesti.readthedocs.io/en/latest/tutorials/general.html.

[6] Mohammad Aali and Jun Liu. Multiple control barrier functions: An application to reactive obstacle avoidance for a multi-steering tractor-trailer system. *arXiv preprint arXiv:2209.05156*, 2022.

[7] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

[8] Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walla M Moursi. Differentiating through a cone program. *J. Appl. Numer. Optim*, 1(2):107–115, 2019.

[9] Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and Systems*, volume 13, pages 1–10. Cambridge, MA, USA, 2017.

[10] Devansh R Agrawal and Dimitra Panagou. Safe control synthesis via input constrained control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6113–6118. IEEE, 2021.

[11] Devansh R Agrawal, Hardik Parwana, Ryan K Cosner, Ugo Rosolia, Aaron D Ames, and Dimitra Panagou. A constructive method for designing safe multirate controllers for differentially-flat systems. *IEEE Control Systems Letters*, 6:2138–2143, 2021.

[12] Mohamad S Alwan and Xinzhi Liu. Recent results on stochastic hybrid dynamical systems. *Journal of Control and Decision*, 3(1):68–103, 2016.

[13] Fabio Amadio, Alberto Dalla Libera, Riccardo Antonello, Daniel Nikovski, Ruggero Carli, and Diego Romeres. Model-based policy search using monte carlo gradient

estimation with real systems application. *IEEE Transactions on Robotics*, 38(6):3879–3898, 2022.

[14] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *European control conference*, pages 3420–3431, 2019.

[15] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. on Automatic Control*, 62(8):3861–3876, 2016.

[16] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in neural information processing systems*, 33:14927–14937, 2020.

[17] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, 31, 2018.

[18] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[19] Olov Andersson, Fredrik Heintz, and Patrick Doherty. Model-based reinforcement learning in continuous environments using real-time constrained optimization. In *AAAI Conf. on Artificial Intelligence*, volume 29, 2015.

[20] Olov Andersson, Mariusz Wzorek, Piotr Rudol, and Patrick Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4597–4604. IEEE, 2016.

[21] Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends® in Machine Learning*, 16(4):494–591, 2023.

[22] Leopoldo Angrisani, Massimo D'Apuzzo, and R Schiano Lo Moriello. The unscented transform: a powerful tool for measurement uncertainty evaluation. In *Proceedings of the 2005 IEEE International Workshop onAdvanced Methods for Uncertainty Estimation in Measurement, 2005.*, pages 27–32. IEEE, 2005.

[23] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[24] Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018.

[25] Alberto Bemporad and Stefano Di Cairano. Model-predictive control of discrete hybrid stochastic automata. *IEEE Transactions on Automatic Control*, 56(6):1307–1321, 2010.

[26] A. Ben-Tal, L. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

[27] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478, 2012.

[28] Aniket Bera, Tanmay Randhavane, Rohan Prinja, Kyra Kapsaskis, Austin Wang, Kurt Gray, and Dinesh Manocha. The emotionally intelligent robot: Improving social navigation in crowded environments. *arXiv preprint arXiv:1903.03217*, 2019.

[29] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

[30] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[31] Mitchell Black, Georgios Fainekos, Bardh Hoxha, Danil Prokhorov, and Dimitra Panagou. Safety under uncertainty: Tight bounds with risk-aware control barrier functions. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12686–12692. IEEE, 2023.

[32] Mitchell Black, Mrdjan Jankovic, Abhishek Sharma, and Dimitra Panagou. Future-focused control barrier functions for autonomous vehicle control. *arXiv preprint arXiv:2204.00127*, 2022.

[33] Mitchell Black, Mrdjan Jankovic, Abhishek Sharma, and Dimitra Panagou. Future-focused control barrier functions for autonomous vehicle control. In *2023 American Control Conference (ACC)*, pages 3324–3331. IEEE, 2023.

[34] Mitchell Black and Dimitra Panagou. Adaptation for validation of a consolidated control barrier function based control synthesis. *arXiv preprint arXiv:2209.08170*, 2022.

[35] Mitchell Black and Dimitra Panagou. Adaptation for validation of consolidated control barrier functions. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 751–757. IEEE, 2023.

[36] Franco Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[37] Franco Blanchini, Stefano Miani, et al. *Set-theoretic methods in control*, volume 78. Springer, 2008.

[38] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[39] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015.

[40] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[41] Eric Bradford, Lars Imsland, Dongda Zhang, and Ehecatl Antonio del Rio Chanona. Stochastic data-driven model predictive control using gaussian processes. *Computers & Chemical Engineering*, 139:106844, 2020.

[42] Joseph Breeden, Kunal Garg, and Dimitra Panagou. Control barrier functions in sampled-data systems. *IEEE Control Systems Letters*, 6:367–372, 2021.

[43] Joseph Breeden and Dimitra Panagou. Predictive control barrier functions for online safety critical control. *arXiv preprint arXiv:2204.00208*, 2022.

[44] Joseph Breeden and Dimitra Panagou. Compositions of multiple control barrier functions under input constraints. In *2023 American Control Conference (ACC)*, pages 3688–3695. IEEE, 2023.

[45] Bruno Brito, Achin Agarwal, and Javier Alonso-Mora. Learning interaction-aware guidance policies for motion planning in dense traffic scenarios. *arXiv preprint arXiv:2107.04538*, 2021.

[46] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE international conference on robotics and automation*, pages 723–730. IEEE, 2011.

[47] Martin Buss, Markus Glocker, Michael Hardt, Oskar Von Stryk, Roland Bulirsch, and Günther Schmidt. Nonlinear hybrid dynamical systems: modeling, optimal control, and applications. In *Modelling, Analysis, and Design of Hybrid Systems*, pages 311–335. Springer, 2002.

[48] Kuanqi Cai, Weinan Chen, Daniel Dugas, Roland Siegwart, and Jen Jen Chung. Sampling-based path planning in highly dynamic and crowded pedestrian flow. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[49] Xiaoyi Cai, Michael Everett, Lakshay Sharma, Philip R Osteen, and Jonathan P How. Probabilistic traversability model for risk-aware motion planning in off-road environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11297–11304. IEEE, 2023.

[50] Roberto Calandra, Nakul Gopalan, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian gait optimization for bipedal locomotion. In *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers 8*, pages 274–290. Springer, 2014.

[51] Marco C Campi, Simone Garatti, and Maria Prandini. Scenario optimization for mpc. *Handbook of Model Predictive Control*, pages 445–463, 2019.

[52] Ovidiu Cârja, Mihai Necula, and Ioan I Vrabie. *Viability, invariance and applications*. Elsevier, 2007.

[53] Jun Chai and Ricardo G Sanfelice. Forward invariance of sets for hybrid dynamical systems (part i). *IEEE Transactions on Automatic Control*, 64(6):2426–2441, 2018.

[54] Jun Chai and Ricardo G Sanfelice. Forward invariance of sets for hybrid dynamical systems (part ii). *IEEE Transactions on Automatic Control*, 66(1):89–104, 2020.

[55] Rohan Chandra, Vrushabh Zinage, Efstathios Bakolas, Peter Stone, and Joydeep Biswas. Deadlock-free, safe, and decentralized multi-robot navigation in social minigames via discrete-time control barrier functions. 2024.

[56] Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-box data-efficient policy search for robotics. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 51–58, 2017.

[57] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[58] Heping Chen, Seth Bowels, Biao Zhang, and Thomas Fuhlbrigge. Controller parameter optimization for complex industrial system with uncertainties. *Measurement and Control*, 52(7-8):888–895, 2019.

[59] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[60] Zhaozhong Chen, Christoffer Heckman, Simon Julier, and Nisar Ahmed. Weak in the nees?: Auto-tuning kalman filters with bayesian optimization. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 1072–1079. IEEE, 2018.

[61] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3387–3395, 2019.

[62] Sheng Cheng, Minkyung Kim, Lin Song, Chengyu Yang, Yiquan Jin, Shenlong Wang, and Naira Hovakimyan. Difftune: Auto-tuning through auto-differentiation. *IEEE Transactions on Robotics*, 2024.

[63] Sheng Cheng, Lin Song, Minkyung Kim, Shenlong Wang, and Naira Hovakimyan. Difftune+: Hyperparameter-free auto-tuning using auto-differentiation. In *Learning for Dynamics and Control Conference*, pages 170–183. PMLR, 2023.

[64] John W Chinneck. An effective polynomial-time heuristic for the minimum-cardinality iis set-covering problem. *Annals of Mathematics and Artificial Intelligence*, 17:127–144, 1996.

[65] John W Chinneck. *Feasibility and Infeasibility in Optimization:: Algorithms and Computational Methods*, volume 118. Springer Science & Business Media, 2007.

[66] John W Chinneck. The maximum feasible subset problem (maxfs) and applications. *INFOR: Information Systems and Operational Research*, 57(4):496–516, 2019.

[67] Andrew Clark. Control barrier functions for stochastic systems. *Automatica*, 130:109688, 2021.

[68] Andrew Clark. Verification and synthesis of control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6105–6112. IEEE, 2021.

[69] Max H Cohen and Calin Belta. Approximate optimal control for safety-critical systems with control barrier functions. In *2020 59th IEEE Conference on Decision and Control*, pages 2062–2067. IEEE, 2020.

[70] Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13:57–98, 1997.

[71] Behdad Davoudi, Karthikeyan Duraisamy, Ella M Atkins, Peter Gaskell, and Maximilian Krogius. Physics-based modeling for autonomous operation of unmanned aerial systems in extreme gusts. In *AIAA AVIATION Forum*, page 3785, 2023.

[72] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023.

[73] Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

[74] Kalyanmoy Deb and Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies: Introductory tutorials in optimization and decision support techniques*, pages 403–449. Springer, 2013.

[75] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[76] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2013.

[77] D Den Hertog, Cornelis Roos, and Tamás Terlaky. On the classical logarithmic barrier function method for a class of smooth convex programming problems. *Journal of Optimization Theory and Applications*, 73(1):1–25, 1992.

[78] Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[79] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[80] Leroy D'Souza and Yash Vardhan Pant. Stochastic hybrid model predictive control using gaussian processes for systems with piecewise residual dynamics. In *2023 American Control Conference (ACC)*, pages 2670–2677. IEEE, 2023.

[81] Sebastian East, Marco Gallieri, Jonathan Masci, Jan Koutnik, and Mark Cannon. Infinite-horizon differentiable model predictive control. *arXiv preprint arXiv:2001.02244*, 2020.

[82] Donald Ebeigbe, Tyrus Berry, Michael M Norton, Andrew J Whalen, Dan Simon, Timothy Sauer, and Steven J Schiff. A generalized unscented transformation for probability distributions. *ArXiv*, 2021.

[83] Ioannis Z Emiris and Vissarion Fisikopoulos. Practical polytope volume approximation. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–21, 2018.

[84] Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. Safe contextual bayesian optimization for sustainable room temperature pid control tuning. *arXiv preprint arXiv:1906.12086*, 2019.

[85] Maziar Fooladi Mahani, Longsheng Jiang, and Yue Wang. A bayesian trust inference model for human-multi-robot teams. *International Journal of Social Robotics*, pages 1–15, 2020.

[86] Anthony Francis, Claudia Pérez-d'Arpino, Chengshu Li, Fei Xia, Alexandre Alahi, Rachid Alami, Aniket Bera, Abhijat Biswas, Joydeep Biswas, Rohan Chandra, et al. Principles and guidelines for evaluating social robot navigation algorithms. *arXiv preprint arXiv:2306.16740*, 2023.

[87] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[88] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, volume 4, page 25, 2016.

[89] Manan S Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2):1423–1430, 2021.

[90] Kunal Garg, Ehsan Arabi, and Dimitra Panagou. Fixed-time control under spatiotemporal and input constraints: A quadratic programming based approach. *Automatica*, 141:110314, 2022.

[91] Kunal Garg and Dimitra Panagou. Control Lyapunov and control barrier functions based quadratic program for spatio-temporal specifications. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1422–1429. IEEE, 2019.

[92] Kunal Garg and Dimitra Panagou. Control-Lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications. In *IEEE 58th Conference on Decision and Control*, pages 1422–1429, 2019.

[93] Kunal Garg, Ricardo G Sanfelice, and Alvaro A Cardenas. Control barrier function-based attack-recovery with provable guarantees. In *IEEE Conference on Decision and Control*, pages 4808–4813, 2022.

[94] Ali Gharib, David Stenger, Robert Ritschel, and Rick Voßwinkel. Multi-objective optimization of a path-following mpc for vehicle guidance: A bayesian optimization approach. In *2021 European Control Conference (ECC)*, pages 2197–2204. IEEE, 2021.

[95] Mohammad Ghavamzadeh and Yaakov Engel. Bayesian policy gradient algorithms. *Advances in neural information processing systems*, 19, 2006.

[96] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE control systems letters*, 1(2):310–315, 2017.

[97] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE control systems magazine*, 29(2):28–93, 2009.

[98] Ruben Grandia, Andrew J Taylor, Aaron D Ames, and Marco 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)*, pages 8352–8358. IEEE, 2021.

[99] Sébastien Gros and Mario Zanon. Towards safe reinforcement learning using nmpc and policy gradients: Part ii-deterministic case. *arXiv preprint arXiv:1906.04034*, 2019.

[100] Luis Guerrero-Bonilla and Vijay Kumar. Realization of r-robust formations in the plane using control barrier functions. *IEEE Control Systems Letters*, 4(2):343–348, 2019.

[101] David Gundana and Hadas Kress-Gazit. Event-based signal temporal logic tasks: Execution and feedback in complex environments. *IEEE Robotics and Automation Letters*, 7(4):10001–10008, 2022.

[102] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[103] Astghik Hakobyan, Gyeong Chan Kim, and Insoon Yang. Risk-aware motion planning and control using cvar-constrained optimization. *IEEE Robotics and Automation letters*, 4(4):3924–3931, 2019.

[104] Astghik Hakobyan and Insoon Yang. Distributionally robust optimization with unscented transform for learning-based motion control in dynamic environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3225–3232. IEEE, 2023.

[105] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[106] Lukas Hewing, Juraj Kabzan, and Melanie N Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2019.

[107] Chuan Hu and Junmin Wang. Trust-based and individualizable adaptive cruise control using control barrier function approach with prescribed performance. *IEEE Trans. on Intelligent Transportation Systems*, 2021.

[108] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019.

[109] Pushpak Jagtap, George J Pappas, and Majid Zamani. Control barrier functions for unknown nonlinear systems using Gaussian processes. pages 3699–3704. IEEE, 2020.

[110] Wanxin Jin, Todd D Murphey, Dana Kulić, Neta Ezer, and Shaoshuai Mou. Learning from sparse demonstrations. *IEEE Transactions on Robotics*, 39(1):645–664, 2022.

[111] Wanxin Jin, Zhaoran Wang, Zhuoran Yang, and Shaoshuai Mou. Pontryagin differentiable programming: An end-to-end learning and control framework. *Advances in Neural Information Processing Systems*, 33:7979–7992, 2020.

[112] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.

[113] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. Spie, 1997.

[114] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[115] Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International conference on artificial intelligence and statistics*, pages 1701–1710. PMLR, 2018.

[116] Hassan K Khalil. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

[117] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[118] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

[119] Rohit Konda, Aaron D Ames, and Samuel Coogan. Characterizing safety: Minimal control barrier functions from scalar comparison systems. *IEEE Control Systems Letters*, 5(2):523–528, 2020.

[120] Nathan J Kong, George Council, and Aaron M Johnson. ilqr for piecewise-smooth hybrid dynamical systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 5374–5381. IEEE, 2021.

[121] Nathan J Kong, Chuanzheng Li, George Council, and Aaron M Johnson. Hybrid ilqr model predictive control for contact implicit stabilization on legged robots. *IEEE Transactions on Robotics*, 2023.

[122] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3(71-104):3, 2014.

[123] Athindran Ramesh Kumar and Peter J Ramadge. Diffloop: Tuning pid controllers by differentiating through the feedback loop. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2021.

[124] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[125] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and computational robotics*, pages 303–307, 2001.

[126] Jaemin Lee, Jeeseop Kim, and Aaron D Ames. Hierarchical relaxation of safety-critical controllers: Mitigating contradictory safety conditions with application to quadruped robots. *preprint arXiv:2305.03929*, 2023.

[127] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *49th IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[128] Unsik Lee and Mehran Mesbahi. Constrained consensus via logarithmic barrier functions. In *2011 50th IEEE conference on decision and control and European control conference*, pages 3608–3613. IEEE, 2011.

[129] Yin Tat Lee and Santosh S Vempala. Convergence rate of riemannian hamiltonian monte carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1115–1121, 2018.

[130] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. *Advances in Neural Information Processing Systems*, 27, 2014.

[131] He Li and Patrick M Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020.

[132] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.

[133] Yun Li, Kiam Heong Ang, and Gregory CY Chong. Patents, software, and hardware for pid control: an overview and analysis of the current art. *IEEE Control Systems Magazine*, 26(1):42–54, 2006.

[134] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE control systems letters*, 3(1):96–101, 2018.

[135] Lars Lindemann and Dimos V Dimarogonas. Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks. *IEEE control systems letters*, 3(3):757–762, 2019.

[136] Lars Lindemann and Dimos V Dimarogonas. Barrier function based collaborative control of multiple robots under signal temporal logic tasks. *IEEE Transactions on Control of Network Systems*, 7(4):1916–1928, 2020.

[137] Lars Lindemann, Haimin Hu, Alexander Robey, Hanwen Zhang, Dimos Dimarogonas, Stephen Tu, and Nikolai Matni. Learning hybrid control barrier functions from data. In *Conference on Robot Learning*, pages 1351–1370. PMLR, 2021.

[138] Simin Liu, Changliu Liu, and John Dolan. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pages 1970–1980. PMLR, 2023.

[139] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[140] Antonio Loquercio, Alessandro Saviolo, and Davide Scaramuzza. Autotune: Controller tuning for high-speed flight. *IEEE Robotics and Automation Letters*, 7(2):4432–4439, 2022.

[141] John Lygeros, Karl Henrik Johansson, Slobodan N Simic, Jun Zhang, and S Shankar Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on automatic control*, 48(1):2–17, 2003.

[142] Hengbo Ma, Bike Zhang, Masayoshi Tomizuka, and Koushil Sreenath. Learning differentiable safety-critical control using control barrier functions for generalization to novel environments. In *2022 European Control Conference*, pages 1301–1308, 2022.

[143] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725. IEEE, 2020.

[144] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic lqr tuning based on gaussian process global optimization. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 270–277. IEEE, 2016.

[145] Kostas Margellos, Paul Goulart, and John Lygeros. On the road between robust optimization and the scenario approach for chance constrained optimization problems. *IEEE Transactions on Automatic Control*, 59(8):2258–2263, 2014.

[146] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26:369–395, 2004.

[147] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

[148] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *IEEE Int. Conf. on Robotics and Automation*, pages 6235–6240, 2015.

[149] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[150] Ali Mesbah. Stochastic model predictive control with active uncertainty learning: A survey on dual control. *Annual Reviews in Control*, 45:107–117, 2018.

[151] Kota Minami, Kotaro Hayashi, and Jun Miura. Development of the pedestrian awareness model for mobile robots. In *32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1295–1301. IEEE, 2023.

[152] Sayan Mitra. *Verifying Cyber-Physical Systems: A Path to Safe Autonomy*. The MIT Press, 2021.

[153] Ihab S Mohamed, Junhong Xu, Gaurav Sukhatme, and Lantao Liu. Towards efficient mppi trajectory generation with unscented guidance: U-mppi control strategy. *arXiv preprint arXiv:2306.12369*, 2023.

[154] Tamas G Molnar and Aaron D Ames. Composing control barrier functions for complex safety specifications. *IEEE Control Systems Letters*, 2023.

[155] Manfred Morari, Carlos E Garcia, and David M Prett. Model predictive control: theory and practice. *IFAC Proceedings Volumes*, 21(4):1–12, 1988.

[156] Aquib Mustafa and Hamidreza Modares. Attack analysis and resilient control design for discrete-time distributed multi-agent systems. *IEEE Robotics and Automation Letters*, 5(2):369–376, 2019.

[157] Aquib Mustafa and Dimitra Panagou. Adversary detection and resilient control for multi-agent systems. *IEEE Transactions on Control of Network Systems*, 2022.

[158] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007.

[159] Quan Nguyen and Koushil Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *2016 American Control Conference (ACC)*, pages 322–328. IEEE, 2016.

[160] Gennaro Notomista, Siddharth Mayya, Seth Hutchinson, and Magnus Egerstedt. An optimal task allocation strategy for heterogeneous multi-robot systems. In *18th European Control Conference*, pages 2071–2076, 2019.

[161] Lukáš Novák and Drahomír Novák. On taylor series expansion for statistical moments of functions of correlated random variables. *Symmetry*, 12(8):1379, 2020.

[162] Aidan O'dwyer. *Handbook of PI and PID controller tuning rules*. World Scientific, 2009.

[163] Anthony O'Hagan. Monte Carlo is fundamentally unsound. *The Statistician*, pages 247–249, 1987.

[164] Kazuhide Okamoto and Panagiotis Tsiotras. Optimal stochastic vehicle path planning using covariance steering. *IEEE Robotics and Automation Letters*, 4(3):2276–2281, 2019.

[165] Pio Ong and Jorge Cortés. Event-triggered control design with performance barrier. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 951–956. IEEE, 2018.

[166] Naoya Ozaki, Stefano Campagnola, Ryu Funase, and Chit Hong Yam. Stochastic differential dynamic programming with unscented transform for low-thrust trajectory design. *Journal of Guidance, Control, and Dynamics*, 41(2):377–387, 2018.

[167] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66), 2022.

[168] Yunpeng Pan and Evangelos Theodorou. Probabilistic differential dynamic programming. *Advances in Neural Information Processing Systems*, 27, 2014.

[169] Dimitra Panagou, Dušan M Stipanovič, and Petros G Voulgaris. Multi-objective control for multi-agent systems using Lyapunov-like barrier functions. In *52nd IEEE Conference on Decision and Control*, pages 1478–1483. IEEE, 2013.

[170] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Distributed coordination control for multi-robot networks using lyapunov-like barrier functions. *IEEE Transactions on Automatic Control*, 61(3):617–632, 2015.

[171] Min Gyu Park, Jae Hyun Jeon, and Min Cheol Lee. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 3, pages 1530–1535. IEEE, 2001.

[172] Hardik Parwana, Mitchell Black, Bardh Hoxha, Hideki Okamoto, Georgios Fainekos, Danil Prokhorov, and Dimitra Panagou. Feasible space monitoring for multiple control barrier functions with application to large scale indoor navigation. *arXiv preprint arXiv:2312.07803*, 2023.

[173] Hardik Parwana, Aquib Mustafa, and Dimitra Panagou. Trust-based rate-tunable control barrier functions for non-cooperative multi-agent systems. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2222–2229. IEEE, 2022.

[174] Hardik Parwana and Dimitra Panagou. Foresee: Prediction with expansion-compression unscented transform for online policy optimization. *arXiv preprint arXiv:2209.12644*, 2022.

[175] Hardik Parwana and Dimitra Panagou. Recursive feasibility guided optimal parameter adaptation of differential convex optimization policies for safety-critical systems. In *IEEE International Conference on Robotics and Automation*, pages 6807–6813, 2022.

[176] Fabio Pasqualetti, Antonio Bicchi, and Francesco Bullo. Consensus computation in unreliable networks: A system theoretic approach. *IEEE Transactions on Automatic Control*, 57(1):90–104, 2011.

[177] Pakpoom Patompak, Sungmoon Jeong, Nak Young Chong, and Itthisek Nilkhamhang. Mobile robot navigation for human-robot social interaction. In *2016 16th international conference on control, automation and systems (ICCAS)*, pages 1298–1303. IEEE, 2016.

[178] Harry A Pierson and Michael S Gashler. Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16):821–835, 2017.

[179] Thomas Pinder and Daniel Dodd. Gpjax: A gaussian process framework in jax. *Journal of Open Source Software*, 7(75):4455, 2022.

[180] Mohammad Pirani, Ehsan Nekouei, Seyed Mehran Dibaji, Henrik Sandberg, and Karl Henrik Johansson. Design of attack-resilient consensus dynamics: a game-theoretic approach. In *2019 18th European Control Conference (ECC)*, pages 2227–2232. IEEE, 2019.

[181] Florian A Potra and Stephen J Wright. Interior-point methods. *Journal of computational and applied mathematics*, 124(1-2):281–302, 2000.

[182] Stephen Prajna, Ali Jadbabaie, and George J Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[183] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by Pareto optimization. *Advances in neural information processing systems*, 28, 2015.

[184] Zengyi Qin, Kaiqing Zhang, Yuxiao Chen, Jingkai Chen, and Chuchu Fan. Learning safe multi-agent control with decentralized neural barrier certificates. *arXiv preprint arXiv:2101.05436*, 2021.

[185] Rahul Rahaman and Alexandre H. Thiery. Uncertainty quantification and deep ensembles. *Advances in neural information processing systems*, 34:20063–20075, 2021.

[186] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*, pages 81–87. IEEE, 2014.

[187] James B Rawlings, Edward S Meadows, and Kenneth R Muske. Nonlinear model predictive control: A tutorial and survey. *IFAC Proceedings Volumes*, 27(2):185–197, 1994.

[188] Maria Isabel Ribeiro. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46):3736–3741, 2004.

[189] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.

[190] Stephen Roberts. Safepilco: A software tool for safe and data-efficient policy synthesis. In *Quantitative Evaluation of Systems: 17th Int. Conf., QEST 2020, Vienna, Austria*, volume 12289, page 18. Springer, 2020.

[191] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

[192] R Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471, 2002.

[193] Ugo Rosolia, Andrew Singletary, and Aaron D Ames. Unified multirate control: From low-level actuation to high-level planning. *IEEE Transactions on Automatic Control*, 67(12):6627–6640, 2022.

[194] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1:127–190, 1999.

[195] Payman Sadegh. A maximum feasible subset algorithm with application to radiation therapy. In *Proceedings of the 1999 American Control Conference*, pages 405–408, 1999.

[196] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[197] Kelsey Saulnier, David Saldana, Amanda Prorok, George J Pappas, and Vijay Kumar. Resilient flocking for mobile robot teams. *IEEE Robotics and Automation letters*, 2(2):1039–1046, 2017.

[198] Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[199] Joohwan Seo, Joonho Lee, Eunkyu Baek, Roberto Horowitz, and Jongeun Choi. Safety-critical control with nonaffine control inputs via a relaxed control barrier function for an autonomous vehicle. *IEEE Robotics and Automation Letters*, 7(2):1944–1951, 2022.

[200] Steven Silva, Nervo Verdezoto, Dennys Paillacho, Samuel Millan-Norman, and Juan David Hernández. Online social robot navigation in indoor, large and crowded environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9749–9756. IEEE, 2023.

[201] Sumeet Singh, Jean-Jacques Slotine, and Vikas Sindhwani. Optimizing trajectories with closed-loop dynamic sqp. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5249–5254. IEEE, 2022.

[202] Andrew Singletary, Karl Klingebiel, Joseph Bourne, Andrew Browning, Phil Tokumaru, and Aaron Ames. Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8129–8136. IEEE, 2021.

[203] Oswin So, Zachary Serlin, Makai Mann, Jake Gonzales, Kwesi Rutledge, Nicholas Roy, and Chuchu Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539. IEEE, 2024.

[204] Eduardo D Sontag. A 'universal'construction of artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13(2):117–123, 1989.

[205] Samuel Stanton, Ke Alexander Wang, and Andrew Gordon Wilson. Model-based policy gradients with entropy exploration through sampling. *ICML Generative Modeling and Model-Based Reasoning for Robotics and AI Workshop*.

[206] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[207] David Stenger and Dirk Abel. Benchmark of bayesian optimization and metaheuristics for control engineering tuning problems with crash constraints. *arXiv preprint arXiv:2211.02571*, 2022.

[208] David Stenger, Muzaffer Ay, and Dirk Abel. Robust parametrization of a model predictive controller for a cnc machining center using bayesian optimization. *IFAC-PapersOnLine*, 53(2):10388–10394, 2020.

[209] David Stenger, Maximilian Nitsch, and Dirk Abel. Joint constrained bayesian optimization of planning, guidance, control, and state estimation of an autonomous underwater vehicle. In *2022 European Control Conference (ECC)*, pages 1982–1987. IEEE, 2022.

[210] Dušan M Stipanović, Claire J Tomlin, and George Leitmann. Monotone approximations of minimum and maximum functions and multi-objective problems. *Applied Mathematics & Optimization*, 66(3):455–473, 2012.

[211] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International conference on machine learning*, pages 997–1005. PMLR, 2015.

[212] Chuangchuang Sun, Dong-Ki Kim, and Jonathan P How. FISAR: Forward invariant safe reinforcement learning with a deep neural network-based optimizer. In *IEEE Int. Conf. on Robotics and Automation*, pages 10617–10624, 2021.

[213] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Trans. on Automatic Control*, 56(7):1495–1508, July 2011.

[214] Xiao Tan, Wenceslao Shaw Cortez, and Dimos V Dimarogonas. High-order barrier functions: Robustness, safety, and performance-critical control. *IEEE Trans. on Automatic Control*, 67(6):3021–3028, 2021.

[215] Xiao Tan and Dimos V Dimarogonas. Compatibility checking of multiple control barrier functions for input constrained systems. *arXiv preprint arXiv:2209.02284*, 2022.

[216] Ran Tao, Sheng Cheng, Xiaofeng Wang, Shenlong Wang, and Naira Hovakimyan. Difftune-mpc: Closed-loop learning for model predictive control. *IEEE Robotics and Automation Letters*, 2024.

[217] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[218] Andrew J Taylor, Pio Ong, Jorge Cortés, and Aaron D Ames. Safety-critical event triggered control via input-to-state safe barrier functions. *IEEE Control Systems Letters*, 5(3):749–754, 2020.

[219] Andrew J Taylor, Pio Ong, Tamas G Molnar, and Aaron D Ames. Safe backstepping with control barrier functions. *arXiv preprint arXiv:2204.00653*, 2022.

[220] Keng Peng Tee, Shuzhi Sam Ge, and Eng Hock Tay. Barrier lyapunov functions for the control of output-constrained nonlinear systems. *Automatica*, 45(4):918–927, 2009.

[221] Neerja Thakkar, Karttikeya Mangalam, Andrea Bajcsy, and Jitendra Malik. Adaptive human trajectory prediction via latent corridors. *arXiv preprint arXiv:2312.06653*, 2023.

[222] Evangelos Theodorou, Yuval Tassa, and Emo Todorov. Stochastic differential dynamic programming. In *American Control Conference*, pages 1125–1132, 2010.

[223] Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. Safety-critical control and planning for obstacle avoidance between polytopes with control barrier functions. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 286–292. IEEE, 2022.

[224] Ran Tian, Liting Sun, Andrea Bajcsy, Masayoshi Tomizuka, and Anca D Dragan. Safety assurances for human-robot interaction via confidence-aware game-theoretic human models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11229–11235. IEEE, 2022.

[225] André L Tits. Feasible sequential quadratic programming. *Encyclopedia of Optimization*, 2009.

[226] Sander Tonkens and Sylvia Herbert. Refining control barrier functions through hamilton-jacobi reachability. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13355–13362. IEEE, 2022.

[227] Sebastian Trimpe, Alexander Millane, Simon Doessegger, and Raffaello D'Andrea. A self-tuning lqr approach demonstrated on an inverted pendulum. *IFAC Proceedings Volumes*, 47(3):11281–11287, 2014.

[228] Ryan Turner and Carl Edward Rasmussen. Model based learning of sigma points in unscented Kalman filtering. *Neurocomputing*, 80:47–53, 2012.

[229] James Usevitch, Kunal Garg, and Dimitra Panagou. Strong invariance using control barrier functions: A clarke tangent cone approach. In *2020 59th IEEE Conference on Decision and Control*, pages 2044–2049. IEEE, 2020.

[230] James Usevitch and Dimitra Panagou. Resilient leader-follower consensus to arbitrary reference values in time-varying graphs. *IEEE Transactions on Automatic Control*, 65(4):1755–1762, 2019.

[231] James Usevitch and Dimitra Panagou. Adversarial resilience for sampled-data systems using control barrier function methods. In *2021 American Control Conference (ACC)*, pages 758–763. IEEE, 2021.

[232] Aris Valtazanos and Subramanian Ramamoorthy. Intent inference and strategic escape in multi-robot games with physical limitations and uncertainty. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3679–3685. IEEE, 2011.

[233] Benjamin Van Niekerk, Andreas Damianou, and Benjamin Rosman. Online constrained model-based reinforcement learning. *arXiv:2004.03499*, 2020.

[234] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pages 4601–4607. IEEE, 2018.

[235] Ngo Anh Vien and Gerhard Neumann. Differentiable robust lqr layers. *arXiv preprint arXiv:2106.05535*, 2021.

[236] Julia Vinogradska, Bastian Bischoff, Jan Achterhold, Torsten Koller, and Jan Peters. Numerical quadrature for probabilistic policy search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 42(1):164–175, 2018.

[237] Veronica Vulcano, Spyridon G Tarantos, Paolo Ferrari, and Giuseppe Oriolo. Safe robot navigation in a crowd combining nmpc and control barrier functions. In *61st IEEE Conference on Decision and Control (CDC)*, pages 3321–3328. IEEE, 2022.

[238] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[239] Chenggang Wang, Shanying Zhu, Bochen Li, Lei Song, and Xinping Guan. Time-varying constraint-driven optimal task execution for multiple autonomous underwater vehicles. *IEEE Robotics and Automation Letters*, 2022.

[240] Jincheng Wang, Lei Xia, Lei Peng, Huiyun Li, and Yunduan Cui. Efficient uncertainty propagation in model-based reinforcement learning unmanned surface vehicle using unscented kalman filter. *Drones*, 7(4):228, 2023.

[241] Xiao Wang. Ensuring safety of learning-based motion planners using control barrier functions. *IEEE Robotics and Automation Letters*, 7(2):4773–4780, 2022.

[242] Ziyi Wang, Oswin So, Keuntaek Lee, and Evangelos A Theodorou. Adaptive risk sensitive model predictive control with stochastic search. In *Learning for Dynamics and Control*, pages 510–522. PMLR, 2021.

[243] Charles W Warren. Global path planning using artificial potential fields. In *1989 IEEE International Conference on Robotics and Automation*, pages 316–317. IEEE Computer Society, 1989.

[244] Charles W Warren. Multiple robot path coordination using artificial potential fields. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 500–505. IEEE, 1990.

[245] Dustin J Webb and Jur Van Den Berg. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE international conference on robotics and automation*, pages 5054–5061. IEEE, 2013.

[246] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[247] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[248] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M Rehg, and Evangelos A Theodorou. Robust sampling based model predictive control with sparse objective information. In *Robotics: Science and Systems*, volume 14, page 2018, 2018.

[249] Elizabeth Wong. *Active-set methods for quadratic programming*. University of California, San Diego, 2011.

[250] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251. IEEE, 2020.

[251] Guofan Wu and Koushil Sreenath. Safety-critical control of a planar quadrotor. In *2016 American control conference (ACC)*, pages 2252–2258. IEEE, 2016.

[252] Zhe Wu, Fahad Albalawi, Zhihao Zhang, Junfeng Zhang, Helen Durand, and Panagiotis D Christofides. Control Lyapunov-barrier function-based model predictive control of nonlinear systems. *Automatica*, 109:108508, 2019.

[253] Wei Xiao and Calin Belta. Control barrier functions for systems with high relative degree. In *2019 IEEE 58th conference on decision and control (CDC)*, pages 474–479. IEEE, 2019.

[254] Wei Xiao and Calin Belta. High-order control barrier functions. *IEEE Transactions on Automatic Control*, 67(7):3655–3662, 2021.

[255] Wei Xiao, Calin Belta, and Christos G Cassandras. Adaptive control barrier functions. *IEEE Trans. on Automatic Control*, 67(5):2267–2281, 2021.

[256] Wei Xiao, Calin A Belta, and Christos G Cassandras. Feasibility-guided learning for constrained optimal control problems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1896–1901. IEEE, 2020.

[257] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 2023.

[258] Zhaoming Xie, C Karen Liu, and Kris Hauser. Differential dynamic programming with nonlinear constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 695–702. IEEE, 2017.

[259] Shakiba Yaghoubi, Keyvan Majd, Georgios Fainekos, Tomoya Yamaguchi, Danil Prokhorov, and Bardh Hoxha. Risk-bounded control using stochastic barrier functions. *IEEE Control Systems Letters*, 5(5):1831–1836, 2020.

[260] Shuo Yang, Shaoru Chen, Victor M Preciado, and Rahul Mangharam. Differentiable safe controller design through control barrier functions. *IEEE Control Systems Letters*, 7:1207–1212, 2022.

[261] Ji Yin, Zhiyuan Zhang, Evangelos Theodorou, and Panagiotis Tsiotras. Trajectory distribution control for model predictive path integral control using covariance steering. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1478–1484. IEEE, 2022.

[262] Ji Yin, Zhiyuan Zhang, and Panagiotis Tsiotras. Risk-aware model predictive path integral control using conditional value-at-risk. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7937–7943. IEEE, 2023.

[263] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021.

[264] Zhaocong Yuan, Adam W Hall, Siqi Zhou, Lukas Brunke, Melissa Greeff, Jacopo Panerati, and Angela P Schoellig. Safe-control-gym: A unified benchmark suite for safe learning-based control and reinforcement learning in robotics. *IEEE Robotics and Automation Letters*, 7(4):11142–11149, 2022, Open source code at https://github.com/utiasDSL/safe-control-gym.

[265] Jun Zeng, Zhongyu Li, and Koushil Sreenath. Enhancing feasibility and safety of nonlinear model predictive control with discrete-time control barrier functions. In *IEEE Conference on Decision and Control*, pages 6137–6144, 2021.

[266] Jun Zeng, Bike Zhang, Zhongyu Li, and Koushil Sreenath. Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility. In *American Control Conference*, pages 3856–3863, 2021.

[267] Jun Zeng, Bike Zhang, and Koushil Sreenath. Safety-critical model predictive control with discrete-time control barrier function. In *2021 American Control Conference*, pages 3882–3889, 2021.

[268] Hongchao Zhang, Zhouchi Li, and Andrew Clark. Model-based reinforcement learning with provable safety guarantees via control barrier functions. In *IEEE International Conf. on Robotics and Automation*, pages 792–798, 2021.

[269] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pages 1–2. Ieee, 2018.

[270] Lifeng Zhou, Vasileios Tzoumas, George J Pappas, and Pratap Tokekar. Resilient active target tracking with multiple robots. *IEEE Robotics and Automation Letters*, 4(1):129–136, 2018.

[271] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *Sixth international conference on intelligent systems design and applications*, volume 2, pages 622–627. IEEE, 2006.

[272] Mang Zhuang and DP Atherton. Automatic tuning of optimum pid controllers. In *IEE Proceedings D (Control Theory and Applications)*, volume 140, pages 216–224. IET, 1993.