DIFFERENTIABLE AND TOLERANT BARRIER STATES FOR IMPROVED EXPLORATION OF SAFETY-EMBEDDED DIFFERENTIAL DYNAMIC PROGRAMMING WITH CHANCE CONSTRAINTS

A Dissertation
Presented to
The Academic Faculty

By

Joshua E. Kuperman

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science in the
Daniel Guggenheim School of Aerospace Engineering
Department of Engineering

Georgia Institute of Technology

May 2023

© Joshua E. Kuperman 2023
DIFFERENTIABLE AND TOLERANT BARRIER STATES FOR IMPROVED EXPLORATION OF SAFETY-EMBEDDED DIFFERENTIAL DYNAMIC PROGRAMMING WITH CHANCE CONSTRAINTS

Thesis committee:

Dr. Evangelos Theodorou (Advisor)  
Aerospace Engineering  
*Georgia Institute of Technology*

Dr. Patricio Vela  
Electrical and Computer Engineering  
*Georgia Institute of Technology*

Dr. Kyriakos Vamvoudakis  
Aerospace Engineering  
*Georgia Institute of Technology*

Date approved: December 12, 2022
# TABLE OF CONTENTS

List of Figures ......................................................... v

List of Acronyms ...................................................... vi

Summary ................................................................. vii

Chapter 1: Introduction ............................................... 1

Chapter 2: Preliminaries: Barrier States ............................ 4

Chapter 3: Tolerant Discrete Barrier States ....................... 7

Chapter 4: T-DBaS-PDP ............................................... 11
  4.1 Problem Formulation ........................................... 11
  4.2 T-DBaS-PDP ................................................... 12

Chapter 5: Applications to Deterministic and Stochastic Safety Functions ........................................ 17
  5.1 Deterministic Safe Trajectory Optimization .................. 17
    5.1.1 Results .................................................. 17
  5.2 Stochastic Safe Trajectory Optimization ...................... 20
    5.2.1 Results .................................................. 21

Chapter 6: Conclusion and Extensions ............................ 25
References
### LIST OF FIGURES

3.1 Intermediate solutions for T-DBaS-DDP and DBaS-DDP. The former method avoids the local minima by taking advantage of the constraint gradient in the unsafe set, while the latter is unable to provide a trajectory that reaches to the goal.  

3.2 The top figure shows an example of the tolerant barrier with the inverse and log barriers. The bottom figures show the gradient functions of the inverse barrier (left) and the tolerant barrier (right) represented by the arrows in which the ellipsoid represents an unsafe region and the color map represents the value of the barrier.  

5.1 Differential Drive trajectory avoiding two rectangular obstacles with their respective safety functions versus time plotted adjacently.  

5.2 Quadrotor trajectory in black. The red spheres are unsafe regions and the dashed blue line represents the trajectory to track. Below is the safety functions of the resulting trajectory. One starts below zero, which represents starting within the middle obstacle.  

5.3 Differential Drive trajectory avoiding two circular obstacles with mean and variance of the obstacle approximations plotted adjacently. Blue regions represent $\mu_h \pm 1\sigma_h, \pm 2\sigma_h, \pm 3\sigma_h$.  

5.4 Quadrotor trajectory avoiding the same obstacles as in Figure 5.2. Mean and variance of the five obstacle approximations plotted adjacently. Blue regions represent $\mu_h \pm 1\sigma_h, \pm 2\sigma_h, \pm 3\sigma_h$. 


A great challenge exists at the intersection of perception and controls – integrating the uncertainty present in perception-based state and obstacle estimation into safe control and trajectory optimization. First, we present the tolerant discrete barrier state (T-DBaS), a novel safety-embedding technique for trajectory optimization with enhanced exploratory capabilities. This approach generalizes the standard discrete barrier state (DBaS) method by accommodating temporary constraint violation during the optimization process while still approximating its safety guarantees. Towards applying T-DBaS to safety-critical autonomous robotics, we combine it with Differential Dynamic Programming (DDP), leading to the proposed safe trajectory optimization method T-DBaS-DDP, which inherits the convergence and scalability properties of the solver. Despite this, the tolerant barrier function parameters require tuning to reach peak performance for a wide array of constraints. To alleviate this requirement, we tune the T-DBaS parameters with the parameterized trajectory optimizer Pontryagin Differentiable Programming (PDP), proposing T-DBaS-PDP, an interpretable and generalizable solver for a variety of optimal control problems. In order to integrate perception uncertainty into safe optimal control, we learn the safety of the system via gaussian processes to create an interpretable, data-driven, and safety-guaranteeable framework. We implement this framework on differential drive and quadrotor dynamics and show its improvement over hand-tuned T-DBaS-DDP.
CHAPTER 1
INTRODUCTION

As many robotics platforms leave the laboratory setting and enter our day-to-day lives, certain applications remain off-limits as much of the current technology lacks interpretability and guarantees of safety, and consequently the trust of those it interacts with. Towards this goal, of particular interest to researchers has been safe reinforcement learning (RL) for its generalizability and ability to leverage data, as well as safe optimal control for its interpretability and derivable guarantees.

In safe RL, an agent learns its dynamics and reward function as well as some representation of safety through interacting with its environment, usually through trial and error. Here, the ability to leverage data comes at the cost of deriving guarantees of safety or success, as only approximations are made. Many promising approaches exist to make RL interpretable and certifiably safe. World models [1] learn a latent representation of the environment and the dynamics, which can be decoded to see the reconstruction of the environment, analogous to the agent’s imagination. Exciting work exists towards adding explicit safety into latent models, such as [2] in which a control barrier function is defined over the latent space or [3] that derives bounds on the task objective and safety. In addition, [4] adds an differentiable CBF-QP layer to a network for obstacle avoidance in autonomous driving. Despite these successes, a catch-22 exists where in order for an agent to learn what is safe, it must learn what is unsafe through failure. The lack of prior knowledge of the environment or interpretability of the controller results in a controller that is not suitable for deployment on physical systems where failure results in costly damage to the robot or requires a human operator to repeatedly reset the experiment during learning.

Conversely, in safe optimal control, we explicitly (and interpretably) consider the safety of the system, resulting in efficient algorithms with guarantees for convergence or safety.
Specifically, barrier based methods [5, 6, 7, 8, 9, 10] have shown to be a viable approach in addressing safety in control systems. In safety-critical trajectory optimization especially, a recent approach that combines DDP and discrete barrier states (DBaS) [10] is shown to outperform other barrier based methods such as the quadratic penalty and control barrier functions (CBFs). This is due to the fact that embedding barrier states into the system’s dynamics eliminates the relative-degree requirement of CBF based methods while providing feedback policies of the barrier that enhance safety and robustness. Moreover, when combined with differential dynamic programming (DDP), creating DBaS-DDP, it enjoys standard DDP convergence analysis and allows for optimizing over the barrier state along with the system’s state and control [10]. Nonetheless, despite the safety guarantees of classical barrier functions, their use often prompts limited exploration of the state space. Specifically, only feasible solutions are allowed when searching for the optimal policy, which is not applicable to data-driven approaches. Alternatively, we can treat the constraints as soft ones by incorporating them into the cost function [11, 12, 13], which would be better suited for learning-based methods. Despite this, there also exists the difficulty to consider data while maintaining computational tractability. This is why safe optimal control methods often assume knowledge of the system, whether that be the state, dynamics, obstacles, or objective. This necessitates the study of the intersection of safe optimal control and deep learning to create a data-informed but interpretable optimal controller that explicitly considers safety and does not require trial and error.

Towards this end, one of the more promising works is Pontryagin Differentiable Programming (PDP) [14]. PDP is largely a trajectory optimization method that assumes some parameterization of the dynamics and/or cost and seeks the parameters that minimize an objective function. PDP bridges the gap between optimal control and learning based methods because the parameters can be anything from model properties to state estimations to weights and biases in a neural network. In addition, the forward pass of PDP can be any arbitrary solver, DBaS-DDP for instance. Safe-PDP, proposed in [15], places the constraints
into the cost using a classical barrier function, which again restricts constraint violations and exploration during iteration.

To first alleviate this requirement in DBaS-DDP, we propose Tolerant (T-)DBaS-DDP, a generalization of DBaS-DDP that allows for temporary constraint violation while iteratively improving the solution. Our proposed method utilizes a novel, tolerant barrier function shaped to approximate DBaS’s safety guarantees and have access to the gradient information within the unsafe set, avoiding local minima. The tolerant barrier function is highly tunable with only four parameters that together alter its value and gradient across the safe and unsafe regions. Secondly, we leverage the tolerant barrier function’s advantages over classical barrier functions to improve upon Safe-PDP, creating T-DBaS-PDP. We apply T-DBaS-PDP to find the optimal tolerant barrier function parameters, avoiding hand-tuning. T-DBaS-PDP also generalizes T-DBaS-DDP to learned probabilistic constraints, with potential to learn dynamics and state estimation as well.

The remaining of this paper is organized as follows. In chapter 2, we introduce the constrained optimal control problem and briefly present the DBaS-DDP method. In chapter 3, we propose the novel tolerant barrier state approach and demonstrate the necessary derivations to formulate T-DBaS embedded DDP. Moreover, we discuss how non-vanishing T-DBaS derivatives benefit DDP in addition to guidelines for setting suitable T-DBaS parameters. In chapter 4, we define the parameterized optimal control problem and derive T-DBaS-PDP. Simulation results are provided in chapter 5. Finally, the conclusions of this paper and future research directions are stated in chapter 6.
CHAPTER 2
PRELIMINARIES: BARRIER STATES

Consider the safety-critical finite horizon optimal control problem

\[ J(x_0, U) = \min_{U} \sum_{t=0}^{T-1} l(t, x_t, u_t) + \Phi(x_T) \]  

(2.1)

subject to the discrete-time, nonlinear dynamical system

\[ x_{t+1} = f(t, x_t, u_t) \]  

(2.2)

and the safety condition

\[ h(t, x_t) > 0, \forall t \in [0, T]; \ x_0 \in S \subset \mathbb{R}^n \]  

(2.3)

where \( x_t \in \mathcal{X} \subset \mathbb{R}^n, u_t \in \mathcal{U} \subset \mathbb{R}^m, U := \{u_1, u_2, \ldots, u_{T-1}\}, l : \mathcal{X} \times \mathcal{U} \to \mathbb{R} \) is the running cost, \( \Phi : \mathcal{X} \to \mathbb{R} \) is the terminal cost, \( f : \mathcal{X} \times \mathcal{U} \to \mathcal{X} \) is the dynamics describing the evolution of the system given at state \( x_t \) under the control \( u_t \) and \( h : \mathcal{X} \to \mathbb{R} \) is a continuously differentiable function defining the safe set \( S := \{x_t \in \mathcal{X} | h(x_t) > 0\} \). The goal of the safety constrained trajectory optimization is to compute the optimal control policy \( U^* \) that minimizes (Equation 2.1) such that the safety condition (Equation 2.3) is satisfied over the whole horizon.

By definition, guaranteeing safety is equivalent to rendering the safe set controller invariant over the horizon [10]. The set \( S \subset \mathbb{R}^n \) is said to be controlled invariant with respect to the finite horizon optimal control problem (Equation 2.1)-(Equation 2.3) under the feedback policy \( U^*(x) \), if \( \forall x_0 \in S, x_t \in S \forall t \in [0, T] \).

To generally enforce controlled invariance of \( S \) using barrier functions methods, [9]
introduced barrier states (BaS) which are embedded into the model of the safety-critical system to be driven and stabilized with the original states of the system. In doing so, the safety objectives are expressed as performance objectives in a higher dimensional state space. In [9], the barrier state embedded model, referred to as the safety embedded model, is asymptotically stabilized, which implies safety due to boundedness of the barrier state. Next, discrete barrier states (DBaS) were proposed for discrete time trajectory optimization in [10] with differential dynamic programming (DDP). Namely, the barrier function $B : \mathcal{S} \to \mathbb{R}$ is defined over $h$. A key point in the definition of the barrier function $B(h(x_t))$, is that $B \to \infty$ as $x_t \to \partial \mathcal{S}$. With this, a barrier function over $x$ can be defined as $\beta(x_t) := B(h(x_t))$. In the discrete setting, the barrier state is simply constructed to be

$$\beta_{t+1} = B \circ h \circ f(t, x_t, u_t). \tag{2.4}$$

For multiple constraints, multiple barrier functions can be added to form a single barrier or multiple barrier states [10]. Then, the barrier state vector $\beta \in \mathcal{B} \subset \mathbb{R}^{n_\beta}$, where $n_\beta$ is the dimensionality of the barrier state vector, is appended to the dynamical model resulting in the safety embedded system

$$\hat{x}_{t+1} = \hat{f}(t, \hat{x}_t, u_t), \tag{2.5}$$

where $\hat{x} = [x ; \beta]$ and $\hat{f} = [f ; B \circ h \circ f]$.

One of the benefits of a safety embedded model is the direct transmission of safety constraint information to the optimal controller. This prevents two separate algorithms from fighting one another for control bandwidth, i.e. the controller attempting to maximize performance and a safety filter attempting to maximize safety. This comes at a cost of the user having to specify the weighting between task performance and safety. For the predictive control problem in this work, the following proposition [10] depicts the safety guarantees provided by the embedded barrier state method. Under the control sequence $U(x)$, the safe set $\mathcal{S}$ is controlled forward invariant if and only if $\beta(x(0)) < \infty \Rightarrow \beta_t < \infty$.

∀ t ∈ [1, T]. Thereby, the constrained trajectory optimization problem in (Equation 2.1)-(Equation 2.3) is transformed to an unconstrained one in a higher dimensional space, which can be written as

\[
J(\hat{x}, U) = \min_U \sum_{t=0}^{T-1} l(t, \hat{x}_t, u_t) + \Phi(NT, \hat{x}_T)
\]  

(2.6)

subject to \( \hat{x}_{t+1} = \hat{f}(t, \hat{x}_t, u_t) \).
CHAPTER 3
TOLERANT DISCRETE BARRIER STATES

T-DBaS are motivated to introduce more flexibility during the optimization process by allowing for temporary constraint violations during solver iteration. Such a behavior is particularly useful in several scenarios and can greatly improve the optimization algorithm capabilities.

An example is shown in Figure 3.1, where a vehicle is required to reach a target behind a wide wall. Due to using the classical barrier, standard DBaS-DDP exhibits limited exploration capabilities and cannot find a safe trajectory that reaches to the target. On the contrary, allowing for temporary violations enhances exploration efficiency. An example in which such a scenario occurs is discussed in subsubsection 5.1.1.

Nonetheless, since theoretical safety guarantees would not hold anymore, the barrier should be designed such that its value sharply and continuously increases inside the unsafe regions. Indeed, note that the barrier’s derivatives are being utilized in the barrier state’s dynamics and cost. This means that the barrier’s gradient should point outside the unsafe region with an increasing intensity instead of diminishing quickly from infinity as is the case for classical barriers (see Figure 3.2). In addition, the Hessian should always remain positive definite to ensure proper minimization.

To achieve these objectives, we propose a new tolerant barrier, $\tilde{B}$, where the desired characteristics are acquired by summing the sigmoid and the softplus function. Specifically, for the safety function $h$, the corresponding sigmoid function is defined as

$$\sigma(h) = \frac{1}{1 + e^{c_1h}}$$

where increasing $c_1$ causes $\sigma(h)$ to closely approximate the unit step function with $\frac{\partial \sigma(h)}{\partial h} = \ldots$
Figure 3.1: Intermediate solutions for T-DBaS-DDP and DBaS-DDP. The former method avoids the local minima by taking advantage of the constraint gradient in the unsafe set, while the latter is unable to provide a trajectory that reaches to the goal.

$c_1\sigma(h)(\sigma(h) - 1)$. Similarly, the softplus function is defined as

$$\sigma^+(h) = \frac{1}{c_2} \log(1 + e^{-c_2h})$$  \hfill (3.1)$$

where increasing $c_2$ causes $\sigma^+(h)$ to closely approximate the ReLU function with $\frac{\partial\sigma^+(h)}{\partial h} = \frac{-1}{1 + e^{c_2h}}$. Hence, the constructed barrier function over the safety function takes the form

$$\tilde{B} = p\sigma(h) + m\sigma^+(h)$$  \hfill (3.2)$$

with $\tilde{B}_x = \left(p\frac{\partial\sigma(h)}{\partial h} + m\frac{\partial\sigma^+(h)}{\partial h}\right)\frac{\partial h}{\partial x}$, where $p$ and $m$ are both scaling parameters. $p$ is the height of the sigmoid function and can be intuitively thought of as the penalty incurred by
Figure 3.2: The top figure shows an example of the tolerant barrier with the inverse and log barriers. The bottom figures show the gradient functions of the inverse barrier (left) and the tolerant barrier (right) represented by the arrows in which the ellipsoid represents an unsafe region and the color map represents the value of the barrier.

approaching the boundary into the unsafe set, and $m$ affects the gradient of $\tilde{B}$ within the unsafe region and intuitively serves as the addition penalty for traveling further into the unsafe set.

Note that $\tilde{B}$ is differentiable and does not blow up as we approach the boundary of the unsafe set but instead changes dramatically (high rate of change) and has non-vanishing gradient and Hessian, which will be of a great importance when used to form barrier states. Because $\tilde{B}$ value does not go to infinity at $h = 0$, Equation 2 does not apply; however, $\tilde{B}$ can be designed such that it ensures safety similar to a classical barrier. $\tilde{B}$ can be
designed to mimic the classical barrier function $B$ near the boundaries practically enough, i.e. $\tilde{B}$ penalizes values of $h$ close to zero as aggressively or more aggressively than $B$. In other words, we can design $\tilde{B}$ such that for a bounded solution of $\beta(t)$, $\tilde{B}(h) \approx B(h)$ in the domain $\{h|h > h(k), k = \arg\max_t \beta(t)\}$ which then by Equation 2, guarantees safety. Additionally, inside the unsafe set $\tilde{B}$ is constantly decreasing as $h \to \partial S$, which pushes constraint violating state trajectories into the safe set, and $\lim_{h \to -\infty} \tilde{B}x = -m$, a tunable parameter. This can be seen in Figure 3.2 which shows a possible behavior of the tolerant barrier compared to some known barrier functions such as the inverse barrier and the log barrier, which both have undesirable properties where $h < 0$. We are now in position to develop the tolerant discrete barrier state and embed it into the trajectory optimization problem. Define $\tilde{\beta}_t(x) := \tilde{B} \circ h(x_t)$. Then, the tolerant discrete barrier state (T-DBaS) is given by

$$\tilde{\beta}_{t+1}(x) := \tilde{B} \circ h \circ f(t, x_t, u_t)$$

(3.3)

It is worth mentioning that one can design a single barrier state or multiple barrier states for multiple constraints [10]. Defining $\hat{x} = [x ; \tilde{\beta}]$ and augmenting the system (Equation 2.2) with the T-DBaS dynamics (Equation 3.3) $\hat{f} = [f ; \tilde{B} \circ h \circ f(t, x_t, u_t)]$ transforms the problem to

$$J(\hat{x}, U) = \min_U \sum_{t=0}^{T-1} l(t, \hat{x}_t, u_t) + \Phi(T, \hat{x}_N)$$

(3.4)

subject to $\hat{x}_{t+1} = \hat{f}(t, \hat{x}_t, u_t)$.

T-DBaS-DDP’s efficacy is strongly shown in [16], in which the authors show its exploratory advantages over its classical counterpart, DBaS-DDP, and its convergence advantages over the Augmented Lagrangian DDP.
In this section we propose T-DBaS-PDP, a safety-embedded parameterized optimal control solver generalizable to a variety of optimal control problems with unknown optimal parameters, such as state, dynamics, or safety estimation.

4.1 Problem Formulation

We wish to solve the safe trajectory optimization problem defined as

\[
\xi^* = \min_{\xi} L(\xi) = \min_{x_0:T, u_0:T-1} \left( \sum_{t=0}^{T-1} l(t, x_t, u_t) + \Phi(T, x_T) \right) \tag{4.1}
\]

\[
x_{t+1} = f(t, x_t, u_t), x_0 \tag{4.2}
\]

\[
0 > h(x_{0:T}) \tag{4.3}
\]

where \( \xi = \{x_{0:T}, u_{0:T}\} \), \( l \) is the running cost, \( \Phi \) is the final cost, \( f \) are deterministic dynamics, and \( h \in \mathbb{R}^{nh} \) are a set of inequality constraints. In [16], this problem is solved by augmenting the state vector with the tolerant discrete barrier state, \( \tilde{\beta} \), parameterized by \( \tilde{\theta} = \{p, m, c_1, c_2\} \) as user-input hyper-parameters. While this approach is shown to be successful and competitive with the state-of-the-art in safe trajectory optimization algorithms, it requires much tuning and deliberation of \( \tilde{\theta} \). To alleviate this process and generalize T-DBaS-DDP to any safe trajectory optimization problem, we propose T-DBaS-PDP, which seeks \( \tilde{\theta}^* \), the optimal T-DBaS-DDP parameters in addition to any others involved in the cost or dynamics, \( \tilde{\theta} \in \mathbb{R}^r \). This parameterization is inherited by the cost and dynamics
functions with \( \xi(\tilde{\theta}) = \{x_{0:T}(\tilde{\theta}), u_{0:T-1}(\tilde{\theta})\} \) and transforms the optimal control problem to

\[
\tilde{\theta}^* = \arg\min_{\tilde{\theta}} (L(\xi^*(\tilde{\theta}))) \tag{4.4}
\]

\[
\xi^*(\tilde{\theta}) = \arg\min_{\xi(\tilde{\theta})} \left( \sum_{t=0}^{T-1} \phi(t, \hat{x}_t(\tilde{\theta}), u_t(\tilde{\theta}); \tilde{\theta}) + \phi(T, \hat{x}_T(\tilde{\theta}); \tilde{\theta}) \right) \tag{4.5}
\]

\[
\hat{x}_{t+1}(\tilde{\theta}) = f(t, \hat{x}_t(\tilde{\theta}), u_t(\tilde{\theta}); \tilde{\theta}), \hat{x}_0 \tag{4.6}
\]

where \( \hat{x} \) is the state vector augmented with our tolerant barrier state \( \tilde{\beta}(h(x_t)) = \tilde{\beta}_t \), \( L \) is the outer cost, \( \phi \) is the inner cost, and \( \xi^*(\tilde{\theta}) \) is the optimal trajectory given a stationary value of \( \tilde{\theta} \). All following variables with notation not explicitly defined are associated with the barrier state augmented system if with \( \hat{\cdot} \) and those with \( \tilde{\cdot} \) are associated with the barrier state itself. Here, we expand the definition of \( \tilde{\theta} \) to include additional non-T-DBaS parameters involved in the cost and dynamics functions. \( \dot{\xi}^*(\tilde{\theta}) \) is found with T-DBaS-DDP and \( \tilde{\theta}^* \) is found using PDP by descending the gradient of the outer cost \( \frac{\partial L}{\partial \tilde{\theta}} \). The description of the latter process is what follows.

### 4.2 T-DBaS-PDP

We define the safety-embedded Hamiltonian as

\[
H_t = c_t(t, \hat{x}_t, u_t; \tilde{\theta}) + f(t, \hat{x}_t, u_t; \tilde{\theta})^T \hat{\lambda}_{t+1}
\]

\[
= c_t(t, \hat{x}_t, u_t; \tilde{\theta}) + f(t, x_t, u_t; \tilde{\theta})^T \lambda_{t+1} + f(t, \tilde{\beta}_t, u_t; \tilde{\theta})^T \tilde{\lambda}_{t+1} \tag{4.8}
\]
that satisfies the PMP conditions

\[
\hat{x}_{t+1} = \frac{\partial H_t}{\partial \hat{\lambda}_{t+1}} = f(t, \hat{x}_t, u_t; \tilde{\theta}) \tag{4.9}
\]

\[
\hat{\lambda}_t = \frac{\partial H_t}{\partial \hat{x}_t} = \frac{\partial c_t}{\partial x_t} + \frac{\partial f^T}{\partial \hat{x}_t} \hat{\lambda}_{t+1} \tag{4.10}
\]

\[
0 = \frac{\partial H_t}{\partial u_t} = \frac{\partial c_t}{\partial u_t} + \frac{\partial f^T}{\partial u_t} \hat{\lambda}_{t+1} \tag{4.11}
\]

\[
\hat{\lambda}_T = \frac{\partial H_T}{\partial \hat{x}_T} \tag{4.12}
\]

subject to the initial condition \( x_0 \).

As a reminder, in order to find \( \tilde{\theta}^* \), we wish to find

\[
\frac{\partial L}{\partial \tilde{\theta}} = \frac{\partial L}{\partial \hat{\xi}} \frac{\partial \hat{\xi}}{\partial \tilde{\theta}} \tag{4.13}
\]

Of the two terms, the second can not be found analytically. To this end, we differentiate the PMP conditions w.r.t. \( \tilde{\theta} \) with the new trajectory to find as

\[
\frac{\partial \hat{\xi}}{\partial \tilde{\theta}} = \left\{ \frac{\partial \hat{x}_{0:T}}{\partial \tilde{\theta}}, \frac{\partial u_{0:T-1}}{\partial \tilde{\theta}} \right\} \tag{4.14}
\]
and the differential PMP conditions:

\[
\begin{bmatrix}
\frac{\partial x_{t+1}}{\partial \theta} \\
\frac{\partial \beta_{t+1}}{\partial \theta}
\end{bmatrix} = \begin{bmatrix} F_t & 0 \\ \tilde{F}_t \end{bmatrix} \begin{bmatrix}
\frac{\partial x_t}{\partial \theta} \\
\frac{\partial \beta_t}{\partial \theta}
\end{bmatrix} + \begin{bmatrix} G_t \\ \tilde{G}_t \end{bmatrix} \frac{\partial u_t}{\partial \theta} + \begin{bmatrix} E_t \\ \tilde{E}_t \end{bmatrix}
\]

(4.15)

\[
\begin{bmatrix}
\frac{\partial \lambda_t}{\partial \theta} \\
\frac{\partial \lambda_{t+1}}{\partial \theta}
\end{bmatrix} = \begin{bmatrix} H_{xx}^t & H_{x\beta}^t \\ H_{\beta x}^t & H_{\beta\beta}^t \end{bmatrix} \begin{bmatrix}
\frac{\partial x_t}{\partial \theta} \\
\frac{\partial \beta_t}{\partial \theta}
\end{bmatrix} + \begin{bmatrix} H_{xu}^t \\ \tilde{H}_t \end{bmatrix} \frac{\partial u_t}{\partial \theta} + \begin{bmatrix} F_t & 0 \\ \tilde{F}_t \end{bmatrix} \begin{bmatrix}
\frac{\partial \lambda_{t+1}}{\partial \theta} \\
\frac{\partial \lambda_t}{\partial \theta}
\end{bmatrix} + \begin{bmatrix} H_{xe}^t \\ H_{\beta e}^t \end{bmatrix}
\]

(4.16)

\[
0 = \begin{bmatrix} H_{tx}^u \\ H_{x\beta}^u \end{bmatrix} \begin{bmatrix}
\frac{\partial x_t}{\partial \theta} \\
\frac{\partial \beta_t}{\partial \theta}
\end{bmatrix} + \begin{bmatrix} G_t \\ \tilde{G}_t \end{bmatrix} \frac{\partial \lambda_t}{\partial \theta} + \begin{bmatrix} H_{tx}^u u_t \\ H_{x\beta}^u \end{bmatrix} + H_{tx}^{ue}
\]

(4.17)

\[
\begin{bmatrix}
\frac{\partial \lambda_{t+1}}{\partial \theta} \\
\frac{\partial \lambda_t}{\partial \theta}
\end{bmatrix} = \begin{bmatrix} H_{x\beta}^{T} & H_{x\beta}^{\beta} \\ H_{\beta x}^{T} & H_{\beta\beta}^{T} \end{bmatrix} \begin{bmatrix}
\frac{\partial x_T}{\partial \theta} \\
\frac{\partial \beta_T}{\partial \theta}
\end{bmatrix} + \begin{bmatrix} H_{x\beta}^{T} \end{bmatrix} \frac{\partial \lambda_{t+1}}{\partial \theta} + \begin{bmatrix} H_{x\beta}^{T} \\ H_{\beta\beta}^{T} \end{bmatrix} = 0
\]

(4.18)

with coefficient matrices defined as follows to simplify notation:

\[
F_t = \frac{\partial f}{\partial x_t}, \quad G_t = \frac{\partial f}{\partial u_t}, \quad E_t = \frac{\partial f}{\partial \theta}, \quad H_{xx}^t = \frac{\partial^2 f}{\partial x_t \partial x_t}, \quad H_{x\beta}^t = \frac{\partial^2 f}{\partial x_t \partial \beta}, \quad H_{x\beta}^{T} = \frac{\partial^2 f}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 f}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 f}{\partial \beta \partial \theta},
\]

(4.19)

\[
\tilde{F}_t = \frac{\partial f}{\partial \beta_t}, \quad \tilde{F}_t = \frac{\partial \tilde{f}}{\partial \beta_t}, \quad \tilde{G}_t = \frac{\partial \tilde{f}}{\partial u_t}, \quad \tilde{E}_t = \frac{\partial \tilde{f}}{\partial \theta}, \quad \tilde{H}_{x\beta}^{T} = \frac{\partial^2 \tilde{f}}{\partial x_T \partial \beta}, \quad \tilde{H}_{x\beta}^{\beta} = \frac{\partial^2 \tilde{f}}{\partial \beta \partial \beta}, \quad \tilde{H}_{x\beta}^{\beta} = \frac{\partial^2 \tilde{f}}{\partial \beta \partial \theta},
\]

(4.20)

\[
H_{x\beta}^{T} = \frac{\partial^2 H_t}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 H_t}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 H_t}{\partial \beta \partial \theta}, \quad H_{x\beta}^{T} = \frac{\partial^2 H_T}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 H_T}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial^2 H_T}{\partial \beta \partial \theta},
\]

(4.21)

\[
H_{x\beta}^{T} = \frac{\partial H_T}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \theta}, \quad H_{x\beta}^{T} = \frac{\partial H_T}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \theta},
\]

(4.22)

\[
H_{x\beta}^{T} = \frac{\partial H_T}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \theta}, \quad H_{x\beta}^{T} = \frac{\partial H_T}{\partial x_T \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \beta}, \quad H_{x\beta}^{\beta} = \frac{\partial H_T}{\partial \beta \partial \theta},
\]

(4.23)

We are now in the position to define our auxiliary control system, as done in [14]. Solving the associated optimal control problem results in the desired \(\frac{\partial \tilde{f}}{\partial \theta}\). Our new optimization variables are

\[
X_t = \frac{\partial x_t}{\partial \theta} \in \mathbb{R}^{nxr}, \quad \tilde{X}_t = \frac{\partial \tilde{f}}{\partial \theta} \in \mathbb{R}^{nxr}, \quad U_t = \frac{\partial u_t}{\partial \theta} \in \mathbb{R}^{nxr}
\]

(4.24)
which gives the following optimal control problem:

\[
\begin{bmatrix}
X_{t+1} \\
\tilde{X}_{t+1}
\end{bmatrix} = 
\begin{bmatrix}
F_t & 0 \\
\tilde{F}_t & \tilde{F}_{t}^\beta
\end{bmatrix}
\begin{bmatrix}
X_t \\
\tilde{X}_t
\end{bmatrix} + 
\begin{bmatrix}
G_t \\
\tilde{G}_t
\end{bmatrix}
U_t + 
\begin{bmatrix}
E_t \\
\tilde{E}_t
\end{bmatrix}
\] (4.25)

\[
\bar{J} = \text{Tr} \sum_{t=0}^{T-1} \left( \frac{1}{2} \begin{bmatrix}
X_t' \\
\tilde{X}_t'
\end{bmatrix}' \begin{bmatrix}
H_{xx}^T & H_{x\tilde{\beta}}^T \\
H_{\tilde{x}\beta}^T & H_{\tilde{x}u}^T
\end{bmatrix} \begin{bmatrix}
X_t \\
\tilde{X}_t
\end{bmatrix} + 
\begin{bmatrix}
H_{x\tilde{\beta}}^T \\
H_{\tilde{x}u}^T
\end{bmatrix}' \begin{bmatrix}
X_T \\
\tilde{X}_T
\end{bmatrix} \right)
\] (4.26)

\[
\text{Tr} \left( \frac{1}{2} \begin{bmatrix}
X_T' \\
\tilde{X}_T'
\end{bmatrix}' \begin{bmatrix}
H_{xx}^T & H_{x\tilde{\beta}}^T \\
H_{\tilde{x}\beta}^T & H_{\tilde{x}u}^T
\end{bmatrix} \begin{bmatrix}
X_T \\
\tilde{X}_T
\end{bmatrix} + 
\begin{bmatrix}
H_{x\tilde{\beta}}^T \\
H_{\tilde{x}u}^T
\end{bmatrix}' \begin{bmatrix}
X_T \\
\tilde{X}_T
\end{bmatrix} \right)
\] (4.27)

Where the optimal differential trajectory minimizes the objective,

\[
\frac{\partial \hat{\xi}^*}{\partial \tilde{\theta}} = \text{argmin}_{\hat{\xi}/\partial \tilde{\theta}} [\bar{J}]
\] (4.28)

These differential PMP conditions are exactly those of the original PMP problem. \(\frac{\partial \hat{\xi}}{\partial \tilde{\theta}}\) is then exactly found via the following recursions, assuming \(H_{uu}^T\) is invertible \(\forall t \in [0, T-1]\).

\[
\hat{P}_t = \hat{Q}_t + \hat{A}_t' (I + \hat{P}_{t+1} \hat{R}_t)^{-1} \hat{P}_{t+1} \hat{A}_t,
\] (4.29)

\[
\hat{W}_t = \hat{A}_t' (I + \hat{P}_{t+1} \hat{R}_t)^{-1} (\hat{W}_{t+1} + \hat{P}_{t+1} \hat{M}_t) + \hat{N}_t
\] (4.30)

\[
\hat{P}_T = H_{xx}^T, \quad \hat{W}_T = H_{\tilde{x}\tilde{e}}^T,
\] (4.31)

\[
\hat{A}_t = \hat{F}_t - \hat{G}_t H_{uu}^{-1} H_{\tilde{x}\tilde{\beta}},
\] (4.32)

\[
\hat{R}_t = \hat{G}_t H_{uu}^{-1} \hat{G}_t',
\] (4.33)

\[
\hat{M}_t = \hat{E}_t - \hat{G}_t H_{uu}^{-1} H_{ue},
\] (4.34)

\[
\hat{Q}_t = H_{xx}^T - H_{\tilde{x}\beta} H_{uu}^{-1} H_{u\tilde{e}}^T,
\] (4.35)

\[
\hat{N}_t = H_{\tilde{x}\tilde{e}} - H_{\tilde{x}u} (H_{uu}^{-1}) H_{\tilde{x}e}^T
\] (4.36)

and \(I\) is the identity matrix. Then, \(\frac{\partial \hat{\xi}}{\partial \tilde{\theta}}\) is found by solving for the following terms from
\[ U_t = -(H_t^{uu})^{-1}(\hat{R}_t^{uu} \hat{X}_t + \hat{G}_t(I + \hat{P}_{t+1} \hat{R}_t)^{-1}(\hat{P}_{t+1} \hat{A}_t \hat{X}_t + \hat{P}_{t+1} \hat{M}_t + \hat{W}_{t+1})) \] (4.37)

\[ \hat{X}_{t+1} = \hat{F}_t \hat{X}_t + \hat{G}_t \hat{U}_t + \hat{E}_t \] (4.38)

The T-DBaS embedding will then help PDP converge, as the additional terms make \( H_t^{uu} \)
more positive definite, as well as the other matrices it affects, in the same way that DBaS
embedding regularizes \( \hat{Q}_{uu} \) in DDP. This of course is only true as long as the outer cost is
dependent on the result T-DBaS trajectory from DDP, i.e. \( \partial L / \partial \hat{\beta} \neq 0 \).
CHAPTER 5
APPLICATIONS TO DETERMINISTIC AND STOCHASTIC SAFETY FUNCTIONS

5.1 Deterministic Safe Trajectory Optimization

In order to solve the safety-critical trajectory optimization problem in Equation 4.1 in [15], inequality and equality constraints are added to the PMP conditions and solved for explicitly, however the authors found that this was computationally expensive and instead opted to absorb the constraints in the cost via log-barrier terms. A limitation of this approach is that it requires feasible initial solutions, which are non-trivially found. In our case, the feasibility of the initial solution is dependent on the output of the T-DBaS-DDP solver using an initial guess of \( \tilde{\theta} \). For this reason, we opt to absorb the inequality constraints in the cost as an addition tolerant barrier term:

\[
L_{\text{det}}(\xi^*(\tilde{\theta})) = \sum_{t=0}^{T-1} l(t, x_t, u_t) + \Phi(T, x_T) + \sum_{t=0}^{T} \tilde{B}(h(x_t)) 
\] (5.1)

This approach is favorable, as an infeasible solution still provides smooth and strong penalization of constraint violations. With the resulting \( \frac{\partial L}{\partial \theta} \) from solving the auxiliary control system, we utilize line-search to descend the gradient and find \( \tilde{\theta}^* \).

5.1.1 Results

In this section we exemplify the performance of PDP on the nonlinear systems, differential drive and quadrotor [17], in scenarios which would otherwise require a large amount of hand-tuning of the T-DBaS parameters for optimal performance.
Differential Drive

Here, we wish to control a differential drive agent through a field of two rectangular obstacles. In the case of T-DBaS-DDP, this would require difficult tuning of the T-DBaS parameters, as the agent must make two sharp turns and must not cut corners, despite the soft constraints.

The obstacles are defined as rectangles with
\[ h = |0.2p_x + p_y| + |0.2p_x - p_y| - 1.8 > 0, \]
where \( p_x, p_y \) are the relative distances of the agents from the obstacle centers at \((3, 1.5), (-3, -1.5)\).

For T-DBaS-DDP, we employ a standard quadratic cost with running or path cost matrix \( Q = \text{diag}(1e^{-3}, 1e^{-3}, 1e^{-3}, 0.1) \), control cost matrix \( R = 0.005I \), and terminal cost matrix \( S = \text{diag}(50, 50, 50, 0.05) \). For the outer-level cost, our matrices are the same, except without the barrier state terms. Our tolerant barrier function parameters for the outer-level cost are \( p = 1, m = 100, c_1 = 500, c_2 = 500 \). Note that these T-DBaS parameters do not perform well with standard T-DBaS-DDP, but they create a barrier function that does not penalize in the safe region with a gradient that points strictly away from the unsafe region. This barrier function is approximately negative linear in the negative region with slope \( m \) with a small negative step function of height \( p \). The resulting trajectory is shown in Figure 5.1. T-DBaS-PDP descends the gradient of the outer loss with respect to the parameters and finds the optimal parameters to be \( p = 18714, m = 4629, c_1 = 1640, c_2 = 5449 \). Though these are not the only possible parameters, as T-DBaS-PDP without the barrier state in the dynamics performs nearly identically, but converges faster to the parameters \( p = 13871, m = 3202, c_1 = 9866, c_2 = 5981 \). These parameters are reached by non-augmented T-DBaS-PDP in 2 iterations as opposed to 18 and result in a 0.04% lower cost.

Further work is required into understanding the role of the barrier state in the PDP auxiliary control system and the backward pass. We wish for \( H_{uu} \) to be invertible so the additional terms that come from \( \frac{\partial^2}{\partial u_t \partial u_t} f(\beta_t, u_t; \tilde{\theta}) \tilde{\lambda}_{t+1} \) should have a regularizing effect, just as the barrier state does for \( Q_{uu} \) in the backward pass of DDP. However, in practice there is not a noticeable advantage in performance over the non-augmented system.
Quadrotor

In this example, a quadrotor starts in an unsafe region, portrayed by a spherical obstacle, and is required to exit the unsafe region and track an unsafe trajectory forming a figure eight as in [10].

To perform the task, we construct a single T-DBaS to represent the obstacles by summing the corresponding tolerant barrier functions.

Note that the quadrotor has to avoid the same unsafe region it started in to show the efficacy of the proposed approach in approximating the safe-set invariance of the DBaS approach despite allowing for unsafe initializations.

We split the safety functions into two barrier states, with the first representing only the center obstacle and the second representing the sum of the other four, hence $\tilde{\theta} \in \mathbb{R}^8$. The cost parameters are $Q = \text{diag}(1e-5, 1e-5, 1e-5, 5e-4, 5e-4, 5e-4, 5e-4, 3e-3, 3e-3, 3e-3, 3e-3, 5e-2, 5e-2, 5e-2, 1e-3, 1e-3), R = 10^{-3}I_{4\times 4}$ with the running cost now defined based on the error of being away from the desired trajectory. The outer cost tolerant barrier function parameters are $p = 100, m = 100, c_1 = 500, c_2 = 500$. The obstacle constraint is $h = ||p - o||^2 - r^2 > 0$, where $p$ is the quadrotor’s position, $o$ is the coordinates of the obstacle’s center, and $r$ is its radius.

The result is shown in Figure 5.2. We show the quadrotor trajectory starting within...
an obstacle and exiting to track a figure-eight pattern, while avoiding further obstacles, as well as the one it originally exited. The optimal parameters resulting from PDP are

\[ p_{\{0,1\}} = \{71, 68\}, m_{\{0,1\}} = \{103, 24\}, c_{1,\{0,1\}} = \{36, 71\}, c_{2,\{0,1\}} = \{42, 44\}. \]

Note that \( m_0 \) is significantly higher than \( m_1 \), for the reason that the interior of the center obstacle must be heavily penalized in order to incentivize high controls to exit the unsafe region quickly. Using the same \( \bar{\theta} \) for all obstacles results in high values of \( m \), resulting in exiting the unsafe region quickly but also overly conservative behavior around the remaining obstacles.

Figure 5.2: Quadrotor trajectory in black. The red spheres are unsafe regions and the dashed blue line represents the trajectory to track. Below is the safety functions of the resulting trajectory. One starts below zero, which represents starting within the middle obstacle.

5.2 Stochastic Safe Trajectory Optimization

Now, instead of having prior knowledge of our safety function, we have an estimation of safety \( h_{\text{est},i}(x_t, t) = \mathcal{N}(\mu_{h,i}, \sigma_{h,i}) \forall i \in n_h \). This can come from any differentiable and data-driven function approximator, but for this work we assume a gaussian distribution of \( P_{h,i} = h_{\text{est},i} \) and utilize gaussian processes [18]. We wish to still use deterministic T-DBaS-DDP in the forward pass of PDP for its efficiency, so we assume \( \mu_h \approx h(x_t, t) \) within DDP. The goal of T-DBaS-PDP is to then find the optimal \( \bar{\theta} \) such that conservative behavior is adopted around areas of poor estimation, or relatively high \( \sigma_h \), and vice versa. We define a
new optimal control problem:

$$\xi^* = \min_{\xi} L(\xi) = \min_{x_{0:T},u_{0:T-1}} \left( \sum_{t=0}^{T-1} l(t, x_t, u_t) + \Phi(t, x_T) \right)$$  \hspace{1cm} (5.2)$$

$$x_{t+1} = f(t, x_t, u_t), x_0$$  \hspace{1cm} (5.3)$$

$$r_{thresh} > R_i(\xi), \forall i \in n_h$$  \hspace{1cm} (5.4)$$

$$R_i(\xi) = \frac{1}{T} \sum_{t=0}^{T} \int_{-\infty}^{0} P_{h,i}(x_t, t) dt$$  \hspace{1cm} (5.5)$$

where \(r_{thresh}\) is a user-determined risk threshold and \(R(\xi)\) represents the probability of constraint violation of the trajectory. We again formulate the high-level cost by absorbing the constraint into the cost:

$$L_{stoch}(\xi^*(\tilde{\theta})) = \sum_{t=0}^{T-1} l(t, x_t, u_t) + \Phi(T, x_T) + \sum_{i=0}^{n_h} \tilde{B}(r_{thresh} - R_i(\xi))$$  \hspace{1cm} (5.6)$$

5.2.1 Results

In this section, we show results from similar experiments as subsection 5.1.1, with instead a gaussian process approximation of the safety function.

Differential Drive

Here we solve the same problem as in subsubsection 5.1.1, now with an estimated probabilistic safety function. We now create a gaussian process for learning the shape of the two rectangles. We choose \(r_{thresh} = 0.005\) and sample 200 points in the region between \(x, y \in [-6, 6]\). We require a relatively large amount of samples because the safety function feature dense and non-smooth. Our kernel was found to perform best as a product of the Matérn kernel with \(\nu = 2.5\) and a polynomial kernel of degree 2. Additionally, we employ the same quadratic cost parameters and outer-cost barrier function parameters \(p = 100, m = 100, c_1 = 1000, c_2 = 1000\). We choose higher \(p\) and \(m\) parameters because
here, the corners of the obstacles are the most critical to avoid collision with while being the hardest to model. The combination of uniform sampling and the difficulty to model the corners results in different approximations of each obstacle, resulting in the need for each obstacle to have an independent set of tolerant parameters, resulting in $\tilde{\theta} \in \mathbb{R}^8$.

The results are shown in Figure 5.3. The optimal parameters found were $p_{\{0,1\}} = \{2.26, 3.01\}, m_{\{0,1\}} = \{4.38, 4.55\}, c_{1,\{0,1\}} = \{6.23, 6.20\}, c_{2,\{0,1\}} = \{5.95, 5.77\}$. While the agent enters the region where the estimated obstacle is within $2\sigma_h$, the overall risk threshold is not broken, with the resulting $R_{0,1}(\xi) = \{0.0007, 0.004\}$. In Figure 5.3, the agent avoids both true obstacles in blue by approximately the same margins, yet the risk for the upper obstacle is significantly lower. This indicates that the samples favored a better approximation of this obstacle, which is reflected by slightly lower $p_0, m_0$ penalizations.

**Quadrotor**

Finally, we learn the spherical obstacles for the quadrotor tracking case. We choose the polynomial kernel with degree 2 and 200 samples in the region $x, y \in [-6, 6], z \in [-3, 3]$. A relatively large amount of samples are required due to the extra dimension that must be captured. Results are shown in Figure 5.4.
Figure 5.3: Differential Drive trajectory avoiding two circular obstacles with mean and variance of the obstacle approximations plotted adjacently. Blue regions represent $\mu_h \pm 1\sigma_h, \pm 2\sigma_h, \pm 3\sigma_h$. 

23
Figure 5.4: Quadrotor trajectory avoiding the same obstacles as in Figure 5.2. Mean and variance of the five obstacle approximations plotted adjacently. Blue regions represent $\mu_h \pm 1\sigma_h, \pm 2\sigma_h, \pm 3\sigma_h$. 

24
CHAPTER 6
CONCLUSION AND EXTENSIONS

In this paper, we first proposed T-DBaS, a variation of DBaS with increased exploration capabilities while approximating the safety guarantees of DBaS. The T-DBaS was designed such that its value increases inside the unsafe regions and retains its derivative information which is then leveraged to find non-trivial solutions. Subsequently, we embedded T-DBaS into the DDP framework for optimal control, and developed T-DBaS-DDP for safe trajectory optimization. Finally, we found the optimal T-DBaS parameters via PDP and proposed T-DBaS-PDP. The resulting algorithm was then implemented on a differential drive robot and a quadrotor in various examples. The logical extension of this work would be to implement it in model predictive control fashion to find the optimal parameters online with little computation per timestep. In addition, with further development of neural representations of stochastic safety, dynamics, and state estimation, T-DBaS-PDP is only one step towards a data-driven, safe, and interpretable planner.
REFERENCES


