Safety Embedded Differential Dynamic Programming using Discrete Barrier States

Hassan Almubarak[1], Kyle Stachowicz[2], Nader Sadegh[3] and Evangelos A. Theodorou[4]

Abstract—**Certified safe control is a growing challenge in robotics, especially when performance and safety objectives are desired to be concurrently achieved. In this work, we extend the barrier state (BaS) concept, recently proposed for stabilization of continuous time systems, to enforce safety for discrete time systems by creating a discrete barrier state (DBaS). The constructed DBaS is embedded into the discrete model of the safety-critical system in order to integrate safety objectives into performance objectives. We subsequently use the proposed technique to implement a safety embedded stabilizing control for nonlinear discrete systems. Furthermore, we employ the DBaS method to develop a safety embedded differential dynamic programming (DDP) technique to plan and execute safe optimal trajectories. The proposed algorithm is leveraged on a differential wheeled robot and on a quadrotor to safely perform several tasks including reaching, tracking and safe multi-quadrotor movement. The DBaS-based DDP (DBaS-DDP) is compared to the penalty method used in constrained DDP problems where it is shown that the DBaS-DDP consistently outperforms the penalty method.**

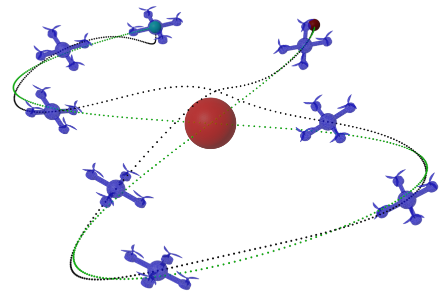

Fig. 1: Quadrotor tracking a predetermined trajectory (green) while avoiding a spherical obstacle. Starting from the green ball, the safe trajectory from DBaS-DDP (black) successfully avoids the obstacle twice to reach the red ball.

I. Introduction

Safety in robotics, in its various forms - including collision avoidance, safe collaboration, etc. - is an important component of expanding the domain of applicability of autonomous robotics. With the increasing demand of autonomy in various industries, this task is increasingly daunting even for known environments. Therefore, there is a clear need for provably safe controls. Yet, the difficulty in unifying safety and performance objectives usually calls for the trade-off between the objectives. In many of the existing safety enforcing techniques, one of the objectives is compromised, which may lead to catastrophic losses. To tackle this problem, this paper develops an uncompromising technique for enforcing safety for discrete time nonlinear systems. Particularly, as long as the performance objectives are feasible, safety is guaranteed. This paper builds on a recently proposed safety integrating technique for stabilization of continuous time systems [3]. Almubarak et al. [3] proposed enforcing safety through barrier states (BaS) which are derived from barrier functions used in optimization to be embedded in the model of the dynamical system. We extend the idea to stabilization and trajectory optimization of discrete time nonlinear systems through the creation of discrete barrier states (DBaS).

Safety, which can be verified through set invariance [8], has been an increasingly sought-after problem in dynamical systems. Recently, barrier-like methods, motivated by barrier methods firstly introduced in optimization, have been a key player to achieve this goal through the creation of barrier certificates [17, 18]. Inspired by control Lyapunov functions and barrier certificates, Wieland and Allgöwer [24] introduced control barrier functions (CBFs) to propose a safety feedback design for continuous time systems. In an attempt to develop safe stabilization, Ames et al. [4] and Romdlony and Jayawardhana [19] proposed spiritually similar but distinct CLF-CBF unification techniques. Ames et al. [4] pioneered the CLF-CBF quadratic program (QP) unification paradigm which was further developed in [5]. The CLF-CBF QP and the developed CBF by Ames et al. [5] have attracted researchers attention to be adopted in various control frameworks and robotic applications [1, 9, 22, 23]. To use CBFs in multi-objective controls which is a typical need in robotic and autonomous systems, however, one needs to trade off between safety and performance objectives [5]. To evade such a trade-off, Almubarak et al. [3] proposed a barrier functions based technique as well to confront such a problem. The proposed technique creates barrier states (BaS) and embeds them in the model of the dynamical system. In doing so, the specified safety condition has been embedded in the stabilization feedback design. It is worth noting, however, that this technique increases the dimension of the model.

For discrete time systems, Agrawal and Sreenath [1] extended the notions of continuous time CBFs and CLF-CBF QPs to discrete ones. Nonetheless, the use of CBFs in the QP framework tends to be more restrictive than the contin-

[1]School of Electrical and Computer Engineering halmubarak@gatech.edu
[2]School of Computer Science kwstach@gatech.edu
[3]George W. Woodruff School of Mechanical Engineering sadegh@gatech.edu
[4]Daniel Guggenheim School of Aerospace Engineering evangelos.theodorou@gatech.edu
Georgia Institute of Technology

uous time case as the optimization problem is non-convex, which limits its applicability, especially for real-time systems. Therefore, they proposed a discrete time exponential barrier function, which was slightly generalized by Ahmadi et al. [2] and called discrete zeroing CBF, in analogy to its counterpart in the continuous time domain [5]. However, although this proposition solves the problem of convexity for linear and quadratic barrier functions, it still needs relaxation to ensure feasibility of the QP problem.

State-constrained DDP is a challenging problem that has been revisited repeatedly in the trajectory optimization literature. Direct methods based on general nonlinear solvers are able to directly incorporate constraints in their transcription, but this comes at the expense of the computational efficiency that DDP enjoys. Murray and Yakowitz [14] describes an early approach to incorporating constraints into DDP, but only considered control constraints. This was later improved using active-set QP solver methods [21], but the state-constrained case has remained a difficult open problem. One common approach is an application of the Augmented Lagrangian [16] [10] technique, which iteratively finds Lagrange multipliers for the constraint with first-order convergence. Xie et al. [25] proposed an active-set approach to the constrained DDP problem, which calculates active-set conditions in the backwards pass and solves a QP at each stage of the forwards pass. Aoyama et al. [6] presented a related algorithm that switches online between an Augmented Lagrangian and an active-set method for faster global convergence. Finally, interior-point methods have been applied in Pavlov et al. [15] to achieve local second-order convergence in the presence of nonlinear constraints. However, these algorithms often have difficulty with highly locally-nonlinear constraints and require substantial tuning and good-quality warm-starts to achieve satisfactory results.

A. Contributions and Organization of the Paper

In this paper, we develop the discrete barrier state (DBaS) concept to enforce safety for nonlinear discrete time systems in Section II. Thereafter, a DBaS is embedded in the system's model forcing the control search to take place in the set of safe controls, which avoids compromising the performance or safety objectives. In Section III, we employ the safety embedding methodology to construct safe stabilization. Additionally, a safe stabilization example to validate the methodology numerically is presented. Subsequently, Section IV leverages the safety embedding technique with differential dynamic programming (DDP) to develop safe trajectory optimization. Moreover, we show that the generated trajectories are guaranteed to be safe as long as the DDP regular convergence conditions are satisfied. The safety embedded DDP, or DBaS-DDP, is used on a simplified omnidirectional robot to safely reach a target point as a proof of concept in Section V. To test the proposed technique on more complicated robot dynamics in a more complicated environment, a differential wheeled robot is to navigate through a potentially unsafe course to arrive to a target between the obstacles. We further exploit the DBaS-DDP to plan and execute safely optimized trajectories for a quadrotor using a 12^{th} dimensional model to achieve different tasks including target reaching and tracking (Fig. 1). Following this further, we compare the DBaS-DDP with the penalty method DDP and demonstrate that it exhibits improved performance and robustness characteristics. Moreover, a multi-quadrotor safe tracking example is provided where two quadrotors are to safely follow similar track starting from two opposite locations. Finally, concluding remarks and future directions are provided in Section VI.

II. Discrete Time Barrier States

Consider the discrete-time nonlinear safety critical control system

$$x(k+1) = f(x(k), u(k)) \tag{1}$$

where $k \in \mathbb{Z}_0^+$ is the time step, $x(k) \in \mathcal{D} \subset \mathbb{R}^n$, $u(k) \in \mathcal{U} \subset \mathbb{R}^m$ and $f : \mathcal{D} \to \mathcal{D}$ is continuous. Throughout the work, we will use the subscript formulation to indicate the time step of the state or the control, e.g. the system (1) will be written as $x_{k+1} = f(x_k, u_k)$. For this system, consider the set $\mathcal{S}$ defined as the superlevel set of a smooth function $h : \mathcal{D} \to \mathbb{R}$ such that

$$\begin{aligned} \mathcal{S} &:= \{x_k \in \mathcal{D} \mid h(x_k) \geq 0\} \\ \mathcal{S}^\circ &:= \{x_k \in \mathcal{D} \mid h(x_k) > 0\} \\ \partial\mathcal{S} &:= \{x_k \in \mathcal{D} \mid h(x_k) = 0\} \end{aligned} \tag{2}$$

where $\mathcal{S}^\circ$ and $\partial\mathcal{S}$ are the interior and the boundary of the set $\mathcal{S}$, respectively. Throughout the paper, we refer to $\mathcal{S}^\circ$ as the safe set. To enforce safety, one needs to satisfy the invariance property given by the following definitions.

Definition 1. The set $\mathcal{S}^\circ \subset \mathbb{R}^n$ is said to be forward invariant for the dynamical system $x(k+1) = f(x(k))$ if $\forall x(0) \in \mathcal{S}^\circ, x(k) \in \mathcal{S}^\circ \ \forall k \in \mathbb{Z}^+$. Equivalently,

$$h(x_k) > 0 \ \forall k \geq 0; \ \ x(0) \in \mathcal{S}^\circ \tag{3}$$

We refer to this as the safety condition.

Definition 2. The set $\mathcal{S}^\circ \subset \mathbb{R}^n$ is said to be *controlled invariant* for the system in (1) if a continuous feedback controller $u_k = K(x_k)$ exists such that for the closed-loop system $x_{k+1} = f(x_k, K(x_k))$, the set $\mathcal{S}^\circ$ is forward invariant. Accordingly, the controller $u_k = K(x_k)$ is said to be safe.

To render $\mathcal{S}^\circ$ controlled invariant for the discrete control system (1), we define the barrier function $B : \mathcal{S}^\circ \to \mathbb{R}$, also known as an interior penalty function in the optimization literature [7], to be a smooth function on the interior of $\mathcal{S}$ that goes to infinity as $x_k \in \mathcal{S}^\circ$ approaches a point of $\partial\mathcal{S}$. Mathematically,

$$x_k \in \mathcal{S}^\circ, \tilde{x} \equiv \lim_{k\to\infty} x_k \in \partial\mathcal{S} \Rightarrow B(x_k) \to \infty, k \to \infty$$

Examples of favored barrier functions with such properties over the set $\mathcal{S}$ defined by $h(x_k)$ include logarithmic barriers such as $B(h(x_k)) = -\log(h(x_k))$ and $B(h(x_k)) = -\log\left(\frac{h(x_k)}{1+h(x_k)}\right)$ and the Carroll barrier, also called the inverse or the reciprocal barrier function, $B(h(x_k)) = \frac{1}{h(x_k)}$. Other barriers can also be defined. Clearly, it should be sufficient to force B to be bounded for all k to guarantee safety, i.e. keeping $x_k \in \mathcal{S}^\circ$. In light of this, Definition 1, Definition 2 and the properties of the barrier function B, the following proposition follows.

Proposition 1. *A continuous feedback controller $u_k = K(x_k)$ is safe, that is it renders $\mathcal{S}^\circ$ controlled invariant, if and only if $B(x(0)) < \infty \Rightarrow B(h(x_k)) < \infty \; \forall k \in \mathbb{Z}^+$.*

Proof. The proof follows directly from the definitions above.

$\Rightarrow$. Suppose there exists the continuous control law $u_k = K(x_k)$ such that $\mathcal{S}^\circ$ is controlled invariant for the control system (1). Then, by Definition 2, $\mathcal{S}^\circ$ is forward invariant w.r.t. the closed loop system $x_{k+1} = f(x_k, K(x_k))$ and consequently, by Definition 1 and the definition of $\mathcal{S}^\circ$, $h(x_k) > 0 \; \forall k \geq 0$. Hence, by the properties and definition of barrier functions, $B(h(x_k)) < \infty \; \forall k \geq 0$.

$\Leftarrow$. Assume $B(x(0)) < \infty \Rightarrow B(h(x_k)) < \infty \; \forall k \in \mathbb{Z}^+$ under the continuous control action $u_k = K(x_k)$. Then by the properties and definition of the barrier functions, $h(x_k) > 0 \; \forall k \geq 0$. As a result, by Definition 1, $\mathcal{S}^\circ$ is forward invariant w.r.t. the closed loop system and hence u_k is said to be safe by Definition 2. □

In multi-objective safety-critical control, possible safety constraints and control performance objectives conflicts appear. A main objective of this paper is to design a safe discrete control technique which meets the safety requirements while avoiding possible conflicts between control objectives and safety constraints without any relaxation. To achieve this goal, the safety constraint is embedded in the system's model used to achieve control performance objectives by means of discrete barrier states (DBaS).

A. Discrete Barrier State (DBaS)

Let us define the barrier function to be $\beta(x_k) := B(h(x_k))$, that is for example for the inverse barrier, $\beta(x_k) = B(h(x_k)) = \frac{1}{h(x_k)}$. Without loss of generality, assume that the origin is a fixed point for the system (1). Define $w_k := \beta(x_k) - \beta^0$, where $\beta^0 = \beta(0)$, which implies that $w_k = 0$ is a fixed point. Consequently, we compute the difference equation for β as

$$\Delta\beta(x_k) = \beta(x_{k+1}) - \beta(x_k) = B(h(f(x_k, u_k)) - \beta(x_k) \tag{4}$$

Notice that from the definition of w_k, it should follow that $\Delta w_k = \Delta\beta_k$. However, some controllability issues may be faced due to augmenting a redundant state. Luckily, this issue can be avoided by perturbing the barrier state equation. Specifically, we add a regularization term $\gamma\Omega(w_k, x_k)$ to the difference equation and by the definition of w_k we get the discrete barrier state (DBaS)

$$w_{k+1} = \gamma\Omega\big(w_k + \beta^0, h(x_k)\big) + B(h(f(x_k, u_k)) - \beta^0 \tag{5}$$

where $|\gamma| < 1$ corresponds to the eigenvalue of w_k of the linearized system, and Ω is a perturbation function such that

$\beta(x)$	$\Omega(w + \beta^0, h)$
$-\log(\frac{h}{1+h})$	$h(e^{w+\beta^0} - 1)^2 - e^{w+\beta^0} + 1$
$1/h$	$h(w + \beta^0)^2 - (w + \beta^0)$
$2\tanh^{-1}(e^{-h})$	$\tanh(\frac{w+\beta^0}{2}) - e^{-h}$

TABLE I: Some barrier functions and corresponding Ω's [3].

$\Omega\big(\beta(x_k), h(x_k)\big) = 0$, i.e. $\Omega \to 0$ as $w_k \to \beta - \beta^0$ and equivalently $\Delta w_k = \Delta\beta_k$. Notice that the design parameter γ is related to the discrete rate at which $w_k \to \beta - \beta^0$. Table I provided in [3] provides some barrier functions and possible corresponding Ωs although other barriers and Ωs can be chosen as long as they meet the properties mentioned above. It must be noted, however, that in many applications, Ω may not be needed ($\gamma = 0$) as it is only needed to ensure stabilizability of the system. In fact, in the simulation examples provided in Section V for trajectory optimization applications, $\gamma = 0$ is picked, i.e. the DBaS equation is not perturbed.

In the following proposition, we show that the perturbation, when needed, does not affect the barrier properties, i.e. safety guarantees still hold.

Proposition 2. *Assume $x(0) \in \mathcal{S}^\circ$ and that $w(0) = \beta(x(0)) - \beta(0)$. The DBaS w_k produced by the perturbed discrete state equation (5) along the states produced by discrete system (1) is bounded if and only if $\beta(x_k) < \infty \; \forall k \in \mathbb{Z}^+$ which implies $x(k) \in \mathcal{S}^\circ \; \forall k \in \mathbb{Z}^+$.*

Proof. Starting with the DBaS state equation (5) for time instant $k = 0$,

$$w_1 = \gamma\Omega\big(w(0) + \beta^0, h(x(0))\big) + B(h(x_1)) - \beta^0$$

Then, given that $w(0) = \beta(x(0)) - \beta^0$,

$$w_1 = \gamma\Omega\big(\beta(x(0)), h(x(0))\big) + B(h(x_1)) - \beta^0$$

Notice that by the definition of Ω, $\Omega\big(\beta(x(0)), h(x(0))\big) = 0$ and therefore $w_1 = B(h(x_1)) - \beta^0 = \beta(x_1) - \beta^0$. Similarly, for the nexst time step $k = 1$,

$$w_2 = \gamma\Omega\big(w(1) + \beta^1, h(x_1)\big) + B(h(x_2)) - \beta^0$$

Hence, from the previous time step $w(1) = \beta(x_1) - \beta^0$, we get

$$w_2 = \gamma\Omega\big(\beta(x_1), h(x_1)\big) + B(h(x_2)) - \beta^0 = \beta(x_2) - \beta^0$$

Clearly, by induction, $w_{k+1} = \beta_{k+1} - \beta^0$ which intuitively implies that w_k is bounded if and only if β_k is bounded $\forall k \in \mathbb{Z}_0^+$. □

In some robotic applications, e.g. in some obstacle avoidance problems, it is more suitable to represent the safe region by a set of smooth functions. In such a case, one may need to use multiple barrier states, as done in the early development of barrier states in [3]. This is not practical in many applications, however. Carelessly increasing the dimensionality of the system could bring lots of difficulties, e.g. the infamous curse of dimensionality. Thankfully, multiple disconnected regions can be represented with only one DBaS by carefully summing the barrier functions. It is worth mentioning that the barrier state that represent multiple constraints, however, becomes highly nonlinear. Moreover, in optimal control settings, one may lose some design flexibility, e.g. penalizing different undesired regions differently. Therefore there is a trade-off between representing multiple constraints with one barrier state and the difficulty of the control design problem. Next, we provide the barrier state that can be used to represent multiple unsafe regions.

Fortunately combining the barrier functions to create a barrier state for the discrete case is simpler than the continuous case [3]. For q constraints, similar to what is done in optimization, the barrier function can be chosen to be $\beta(x) = \sum_{i=1}^{q} B(h^{(i)}(x_k))$, where $h^{(i)}(x_k)$ represents the i^{th} function describing the region of interest. Then, following the same procedure used to define the barrier state in (5), we get

$$\beta(x_{k+1}) = \sum_{i=1}^{q} B \circ h^{(i)}\big(f(x_k, u_k)\big)$$

and therefore

$$\begin{aligned} w_{k+1} =& \gamma\Omega\big(w_k + \beta^0 - \sum_{j=1, j\neq i}^{q} B \circ h^{(j)}(x_k), h^{(i)}(x_k)\big) \\ &+ \sum_{i=1}^{q} B \circ h^{(i)}\big(f(x_k, u_k)\big) - \beta^0 \end{aligned} \tag{6}$$

where $\beta^0 = \sum_{i=1}^{q} B \circ h^{(i)}(0)$.

Since we have developed our DBaS, we are now in a position to propose the safety embedded model used to design the safety embedded control for safety critical systems. To develop the safety embedded model, the barrier state (6) is appended to the model of the safety critical system (1) giving

$$\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k) \tag{7}$$

where $\hat{x}_k = \begin{bmatrix} x_k \\ w_k \end{bmatrix}$ and $\hat{f} : \mathcal{D} \to \mathcal{D}$ is a vector field representing the system's dynamics in (1) and the barrier state's dynamics in (6). It must be noted that $\hat{f}$ is continuous due to continuity of f and smoothness of h and B.

As a consequence of the development above, the forward invariance of $\mathcal{S}^\circ$, i.e. safety of the safety-critical system, can be tied to stability of the generated trajectories of the safety embedded system (7) for stabilization tasks. Similarly, the same concept can be used in trajectory optimization where boundedness of the DBaS implies the generation of safe trajectories. In the next sections, we use the developed DBaS to design a safely stabilizing control which we implement on simple nonlinear and linear systems to show the efficacy of the proposed technique. Then we use a finite horizon trajectory optimization technique, namely differential dynamic programming (DDP), to generate safely optimized trajectories via embedding a DBaS in the algorithm. In each section, safety guarantees are provided.

It is worth noting that for discrete-time systems, the CBF safety condition is no longer affine in u_k, since the controller shows up inside the barrier function, even for linear systems which results in non-quadratic and possibly non-convex optimization. To solve this problem, the discrete time exponential CBF was proposed in [1] which was slightly relaxed in [2]. On the other hand, for the DBaS method, although a similar phenomenon is faced, this is not an issue as it does not affect the development or the safety guarantees, as we will show in the examples in the following sections.

III. Safe Stabilization via Discrete Barrier States

Theorem 3. *Suppose that there exists a continuous feedback controller $u_k = K(\hat{x}_k)$ that renders the safe point of interest, e.g. the origin, of the safety embedded closed loop system $\hat{x}_{k+1} = \hat{f}(\hat{x}_k, K(\hat{x}_k))$ asymptotically stable. As a result, the safe set $\mathcal{S}^\circ$ is controlled invariant with respect to the safety critical system, i.e. the controller u_k is safe and it renders the trajectories of the safety critical closed loop system $x_{k+1} = f(x_k, K(\hat{x}_k))$ safe.*

Proof. The proof is analogous to that in [3] for continuous time systems and is omitted. □

A. Discrete Linear Control Application

For some control design techniques, controllability is needed and stabilizability is not enough. Therefore, one can perturb the DBaS model by choosing Ω to be a linear term in w_k to guarantee controllability. That is, the DBaS can be given by

$$w_{k+1} = \gamma w_k + B\big(h(f(x_k, u_k))\big) - \beta^0 \tag{8}$$

Now we show that this perturbation does not affect the safety guarantees, that is if the origin of the safety embedded system (7) is rendered asymptotically stable, the barrier state w_k generated by (8) is bounded if and only if the barrier β_k is bounded.

Proposition 4. *The DBaS w_k produced by the perturbed discrete state equation (8) along the states produced by the asymptotically stable discrete time system $x_{k+1} = f(x_k, K(x_k))$, with respect to the safe origin, is bounded if and only if $\beta(x_k) < \infty \ \forall k \in \mathbb{Z}^+$ which implies $x(k) \in \mathcal{S}^\circ \ \forall k \in \mathbb{Z}^+$.*

Proof. $\Rightarrow$. Suppose that w_k resulting from (8) along (1) is bounded for all k. Clearly, $\beta_k = w_{k+1} - \gamma w_k + \beta^0$ is bounded for all k.

$\Leftarrow$. Suppose that β_k is bounded under the asymptotically stable discrete system $x_{k+1} = f(x_k, K(x_k))$ with respect to the safe origin. Asymptotic stability of x_k implies asymptotic stability of β_k. Therefore, $w_{k+1} = \gamma w_k + \beta_k - \beta^0$ is asymptotically stable for $|\gamma| < 1$ and consequently bounded input bounded output stable. Hence, w_k is bounded. □

B. Stabilization Example

Consider the discrete system used by Agrawal and Sreenath [1] with a linear safety constraint to test the discrete CBF,

$$x_{k+1} = \begin{bmatrix} 0.5\sin(x_{1,k}) + x_{1,k} + 2x_{2,k} + u_k \\ 0.5\sin(x_{2,k}) + 2x_{1,k} + 2_{2,k} + 2u_k \end{bmatrix} \tag{9}$$

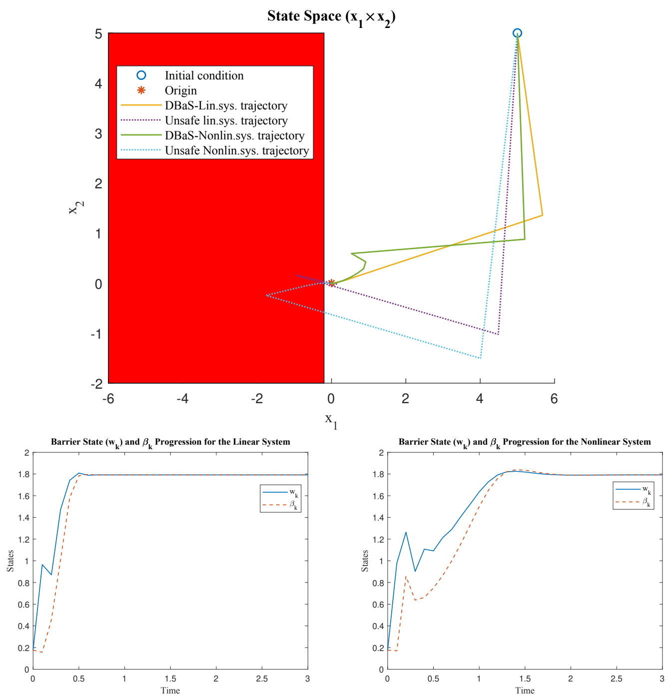

Fig. 2: Safe stabilization of the origin of the nonlinear system (9) and the linearized system using a safety embedded linear control designed by the pole placement method. Designing a linear controller using the original model yields unsafe solutions crossing the unsafe red region. It is also shown that the perturbed DBaS w converges to the barrier β.

The linear system will have the matrices $A = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. Define the linear function $h(x_k)$ that defines the safe set as $h(x_k) = Hx_k + F$ such that $\mathcal{S}^\circ := \{x_k \in \mathcal{D} \mid Hx_k + F > 0\}$ where $H = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and $F = 0.2$, i.e. x_1 is not to cross -0.2. Choosing $\beta(x_k) = -\log(\frac{h}{1+h})$ and $\gamma = -0.5$, the augmented linearized system will be

$$x_{k+1} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 2 & 0 \\ -4.1667 & -8.3333 & -0.5000 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 2 \\ -4.1667 \end{bmatrix} u_k$$

To show the efficacy of the proposed DBaS method, we use the simple pole placement method to design a safe stabilizing controller to safely stabilize the origin of the linearized system and the original nonlinear system. Notice that the safety embedded system is nonlinear due to nonlinearity of w_k even for linear $h(x_k)$. We pick the closed loop eigenvalues to be $[-0.1, -0.1, -0.1]$. It is worth mentioning that one can use a discrete quadratic linear regulator (DLQR) to safely stabilize the system, which only needs stabilizability and gives us the chance to penalize the states including the barrier state. Fig. 2 shows that the safety embedded discrete linear controller is able to safely stabilize the origin of the systems. Additionally, it is shown that the DBaS w converges to the barrier function β for the linear systems as well as the nonlinear one.

Next, we use the proposed safety embedding methodology for discrete time systems with differential dynamic programming (DDP) to generate safely optimized trajectories.

IV. Safety Embedded Differential Dynamic Programming (DDP)

In this section, we employ the proposed DBaS methodology for safe trajectory optimization. Specifically, we apply differential dynamic programming (DDP) [13, 11, 12] to the safety-embedded control system (7) to generate safe and optimized trajectories, which is of a big interest in many robotic applications. This can be looked at as a constrained DDP problem, which has been a particularly challenging problem in the DDP literature. To generate optimal and feasible trajectories, DDP performs a sequence of iterative backward and forward passes along the optimality conditions, i.e. the value function and its derivatives resulting from expanding the Bellman equation, and the system's dynamics respectively. Under the required conditions and assumptions, the DDP algorithm converges to a feasible optimal trajectory.

A. Differential Dynamic Programming

Consider the finite horizon optimal control problem

$$V_k(x) = \min_{U_k} \sum_{i=k}^{N-1} l(x_i, u_i) + \Phi(x_N) \tag{10}$$

subject to the dynamical system (1) and the safety condition (3), where $U_k = \{u_k, u_{k+1} + \cdots + u_{N-1}\}$, l and Φ are the the running cost and terminal cost respectively. Utilizing the Dynamic Programming (DP) optimality principle, we can set $V_N(x) = l_f(x_N)$ and reduce the problem to a sequence of minimizations to be performed over a single control instead of performing the minimization over the whole sequence of controls U_k. Hence, we get the well known Bellman equation

$$V_k(x_k) = \min_{u_k} [l(x_k, u_k) + V_{k+1}(f(x_k, u_k))] \tag{11}$$

Clearly, one needs to solve this backwards in time.

The Differential Dynamic Programming (DDP) algorithm iteratively solves the optimal control problem starting by solving the Bellman equation (11) backwards on a nominal trajectory $(\bar{x}, \bar{u})$ to find the optimal control sequence. Then, the new nominal trajectory for the system is computed forwardly. The process is repeated until convergence is achieved. In order to do this, the DDP algorithm expands the Bellman equation around the nominal trajectory to its second order.

Using the proposed DBaS technique to render $\mathcal{S}^\circ$ forward invariant, the constrained finite horizon optimal control problem reduces to an unconstrained optimal control problem that minimizes (10) subject to the safety embedded dynamics (7). That is,

$$V_k(\hat{x}) = \min_{U_k} \sum_{i=k}^{N-1} l(\hat{x}_i, u_i) + \Phi(\hat{x}_N) \tag{12}$$

subject to $\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k)$. Therefore, the associated Bellman Equation can be given by $V_k(\hat{x}_k) = \min_{u_k} [l(\hat{x}_k, u_k) +$

$V_{k+1}(\hat{f}(\hat{x}_k, u_k))]$. For the DDP equations used in the algorithm, define

$$
\begin{aligned}
H_{\hat{x}_k} &= l_{\hat{x}_k} + V^{\mathrm{T}}_{\hat{x}_{k+1}} \hat{f}_{\hat{x}_k} \\
H_{u_k} &= l_{u_k} + V^{\mathrm{T}}_{\hat{x}_{k+1}} \hat{f}_{u_k} \\
H_{\hat{x}\hat{x}_k} &= l_{\hat{x}\hat{x}_k} + \hat{f}^{\mathrm{T}}_{\hat{x}_k} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{\hat{x}_k} + V_{\hat{x}_{k+1}} \hat{f}_{\hat{x}\hat{x}_k} \\
H_{uu_k} &= l_{uu_k} + \hat{f}^{\mathrm{T}}_{u_k} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{u_k} + V_{\hat{x}_{k+1}} \hat{f}_{uu_k} \\
H_{\hat{x}u_k} &= l_{\hat{x}u_k} + \hat{f}^{\mathrm{T}}_{\hat{x}_k} V_{\hat{x}\hat{x}_{k+1}} \hat{f}_{u_k} + V_{\hat{x}_{k+1}} f_{\hat{x}u_k}
\end{aligned}
\tag{13}
$$

Then, the optimal variation may be given by

$$
\delta u^*_k = -H^{-1}_{uu_k} \left(H^{\mathrm{T}}_{u_k} + H_{\hat{x}u_k} \delta \hat{x}_k \right) \tag{14}
$$

where $\delta u_k = u_k - \bar{u}_k$ is the deviation from the k^{th} action in the nominal sequence $\bar{U}_k$. Now, as we need to minimize the expanded Bellman equation, setting each power of approximation to zero leads to the Riccati equations for V_k, $V_{\hat{x}_k}$ and $V_{\hat{x}\hat{x}_k}$ that are solved to get

$$
\begin{aligned}
V_k &= V_{k+1} - \frac{1}{2} H_{u_k} H^{-1}_{uu_k} H^{\mathrm{T}}_{u_k} \\
V_{\hat{x}_k} &= H_{\hat{x}_k} - H_{\hat{x}u_k} H^{-1}_{uu_k} H_{u_k} \\
V_{\hat{x}\hat{x}_k} &= \frac{1}{2} (H_{\hat{x}\hat{x}_k} - H_{\hat{x}u_k} H^{-1}_{uu_k} H_{u\hat{x}_k})
\end{aligned}
\tag{15}
$$

which are the equations used for the backward propagation. Consequently, one can compute V_k and its gradient and Hessian along the states' trajectory as well as the optimal variation δu backwards from $k = N-1$ to $k = 1$ with the initialization $V_N(x_N) = l_f(x_N)$. Next, the new trajectory is computed forwardly

$$
V_k = V_{k+1} - \frac{1}{2} H_{u_k} H^{-1}_{uu_k} H^{\mathrm{T}}_{u_k} \tag{16}
$$

$$
u_k = \bar{u}_k - H^{-1}_{uu_k} \left(H_{u_k} + H_{\hat{x}u_k} \delta x_k \right) \tag{17}
$$

$$
\hat{x}_{k+1} = \hat{f}(\hat{x}_k, u_k) \tag{18}
$$

where $\delta x_k = x_k - \bar{x}_k$ is the deviation from the k^{th} state in the nominal sequence of states $\bar{X}_k$. This is repeated until convergence.

Note that for the DDP problem to be well-defined, we must have $H_{uu_k} \succ 0$ [12]. For this to be the case it is sufficient to have that $V_{xx_k} \succ 0$ for all k. However, in the general case there is a distinct possibility that l_{xx} is indefinite: it is perfectly reasonable for the cost function to be locally non-convex and in fact this is necessarily the case in the obstacle-avoidance problem. In contrast, the DBaS cost remains a convex function of the states, meaning that for a cost function $l(\hat{x}, u) = l_x(x, u) + l_w(w)$ its hessian $l_{\hat{x}\hat{x}}$ remains positive.

Theorem 5. *Let $\Phi_{\hat{x}\hat{x}}, l_{uu} \succ 0$, and drop second-order dynamics terms $\bar{f}_{\hat{x}\hat{x}}$. Further, assume $\hat{x} = [x \; w]^{\mathrm{T}}$ where x is the state of the safety critical system and w is the discrete barrier state, and let $l(\hat{x}, u)$ be of the form $l(\hat{x}, u) = \ell_x(x, u) + \ell_w(w)$, where $\ell_{ww} \succ 0$ and $\begin{pmatrix} \ell_{xx} & \ell_{xu} \\ \ell_{ux} & \ell_{uu} \end{pmatrix} \succ 0$. Then, for all k, $V_{\hat{x}\hat{x}_k}$ is positive definite.*

Proof. By induction: for the base case, we have by assumption that $V_{\hat{x}\hat{x}_N}$ is positive-definite, as it is simply $\Phi_{\hat{x}\hat{x}}$.

Then, we want to show that if $V_{\hat{x}\hat{x}_{k+1}}$ is positive definite we also have that $V_{\hat{x}\hat{x}_k}$ is positive definite. Examine the second-order expansion of the Bellman equation:

$$
\begin{aligned}
\delta \hat{x}^{\mathrm{T}}_k V_{\hat{x}\hat{x}_k} \delta \hat{x}_k &= \min_u \begin{pmatrix} \delta x_k \\ \delta u_k \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \ell_{xx_k} & \ell_{xu_k} \\ \ell_{ux_k} & \ell_{uu_k} \end{pmatrix} \begin{pmatrix} \delta x_k \\ \delta u_k \end{pmatrix} \\
&\quad + \delta w^{\mathrm{T}}_k \ell_{ww_k} \delta w_k + \delta \hat{x}^T_{k+1} \bar{V}_{\hat{x}\hat{x}_{k+1}} \delta \hat{x}_{k+1} \\
&\geq \delta \hat{x}^T_{k+1} V_{\hat{x}\hat{x}_{k+1}} \delta \hat{x}_{k+1} > 0 \\
\delta u^{\mathrm{T}}_k H_{uu_k} \delta u_k &= \delta u^{\mathrm{T}}_k \left(l_{uu} + f^{\mathrm{T}}_u V_{\hat{x}\hat{x}_{k+1}} f_u \right) \delta u \\
&\geq (f_u \delta u)^{\mathrm{T}} V_{\hat{x}\hat{x}_{k+1}} (f_u \delta u_k) \geq 0
\end{aligned}
$$

So $V_{\hat{x}\hat{x}_k}$ is positive definite, and furthermore $H_{\bar{u}\bar{u}}$ is also positive definite. By induction, this holds for all k. □

A natural question to ask is whether it is advantageous to include the barrier state in the model of the safety critical system's dynamics instead of simply adding the barrier function to the cost function in the optimization problem as in some constrained DDP approaches. Those approaches are known as penalty methods, which are often used for their simplicity of implementation. In those methods, the modified optimization problem given by the cost function $l(x, u) = \beta(x) + l'(x, u)$, where β is a barrier function, appears at the surface level to be equivalent to the barrier state formulation with $\gamma = 0$. Indeed, it is true that any local optimum for the DBaS based optimal control problem is also an optimum for the penalty method. The two mechanisms are substantially different, nevertheless. First, in many robotic applications, this new cost function is highly locally non-convex, which can make these types of problems very difficult to solve. Theorem 5 explains part of the practical improvement seen when using barrier states over simple penalty methods: it moves some nonlinear terms from the cost function to the dynamics, removing local non-convexity from the problem. In this sense, the DBaS-DDP has a regularizing effect on the algorithm when applied to highly non-convex cost functions. In practice, this is of a great value as any cost function that causes the optimal plan to avoid a convex and bounded unsafe region will necessarily be non-convex. It is worth mentioning that for the simulation experiments presented in Section V, the DBaS-DDP did not need any explicit regularization, i.e. adding a multiple of the identity matrix to the matrices H_{uu_k}, H_{xx_k} and H_{xu_k} unlike some constrained DDP methods [25, 6] including the penalty method. Moreover, it is important to note that in the case of DBaS-DDP, the optimizer has richer information and can anticipate the forces it will face from the barriers due to embedding the barrier state in the dynamics resulting in a smoother and easier convergence. This is also due to the fact that in the DBaS-DDP case, the solver optimizes for the barrier state, δw_k as well as δx_k and δu_k unlike the penalty method case where it only optimizes for δx_k and δu_k. Additionally, the feedback term of the barrier state directly supplies the controller with local information not available otherwise, as the barrier state w_k is propagated using the full nonlinear dynamics. It should be recalled, however, that those advantages come at the cost of increasing the dimensionality of the model.

A comparison between DBaS-DDP and the ordinary penalty method is presented in Section V.

Under certain conditions, with the incorporation of line search, standard DDP is able to guarantee that a single iteration of DDP will improve the trajectory's cost. Similarly, we show that in our formulation, a single iteration of DDP-DBaS is able to find a *safe* trajectory with improved cost.

Theorem 6 (Improvement of Safe Trajectory)**.** *Let* $(\bar{x}, \bar{u})$ *be a safe nominal trajectory, and let* $\delta u = K\delta x + \epsilon k$*. If* δu_k *is nonzero for some* k*, then there exists some* $0 < \epsilon \leq 1$ *such that* $J(\bar{x}+\delta x, \bar{u}+\delta u) < J(\bar{x}, \bar{u})$ *and such that* $\bar{x}+\delta x, \bar{u}+\delta u$ *is a safe trajectory.*

Proof. Because $\bar{x}, \bar{u}$ is safe and the safe set is open, there exists some neighborhood U of safe trajectories near $\bar{x}, \bar{u}$. Define $J(\pi)$ as the objective function:

$$J(\pi) = \sum_{k=1}^{N-1} l(\hat{x}_k, u_k) + \Phi(\hat{x}_N)$$

with the update rule $\delta u_k = u_k - \bar{u}_k = K_k \delta x_k + \epsilon k_k$. Find partial derivatives with respect to u_k using Bellman's principle:

$$\Delta J = [l(\hat{x}_k, u_k) + V(f(\hat{x}_k, u_k))] - [l(\bar{x}_k, \bar{u}_k) + V(f(\bar{x}_k, \bar{u}_k))]$$

$$\frac{\partial}{\partial u_k}\Delta J = H_{u_k}, \ \frac{\partial^2}{\partial u_k^2}\Delta J = H_{uu_k}$$

Rewrite the objective function ΔJ as a function of the parameter ϵ, with Taylor expansion around zero:

$$\Delta J(\epsilon) = \sum_{k=1}^{N-1}\left[H_{u_k}\frac{\partial u_k}{\partial \epsilon}\epsilon + \frac{1}{2}\frac{\partial u_k}{\partial \epsilon}^{\mathrm{T}} H_{uu_k}\frac{\partial u_k}{\partial \epsilon}\epsilon^2\right] + \mathcal{O}(\epsilon^3)$$

Finally, because $u_k = \bar{u}_k + K_k \delta\hat{x}_k + \epsilon k_k$, we have $\frac{\partial u_k}{\partial \epsilon} = k_k = -Q_{uu}^{-1}$:

$$\Delta J(\epsilon) = \sum_{k=1}^{N-1}\left[\frac{\epsilon^2}{2} H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}} - \epsilon H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}}\right] + \mathcal{O}(\epsilon^3)$$
$$= (\frac{1}{2}\epsilon^2 - \epsilon)\sum_{k=1}^{N-1}\left[H_{u_k} H_{uu_k}^{-1} H_{u_k}^{\mathrm{T}}\right] + \mathcal{O}(\epsilon^3)$$

By Theorem 5, we know that H_{uu_k} is positive definite. Then, the summation is positive and so for $\epsilon \leq 1$ the lower-order terms are negative. In addition, by making ϵ small we can reduce the higher-order terms to be arbitrarily small compared to the reduction in cost.

Finally, because there is an open neighborhood U of safe trajectories around the nominal trajectory, by making ϵ small we will find a trajectory within the safe set. In particular, because a trajectory is safe if and only if it has a bounded cost by design, any ϵ that leads to a reduction in cost yields a safe trajectory. □

V. Safety Embedded DDP Application Examples

A. Planar Double-Integrator

As a simple proof-of-concept we apply DBaS-DDP to the planar double-integrator problem. In this problem, a simple omnidirectional robot navigates from an initial position to a prespecified goal. We used a very simple acceleration-based model for the robot (all examples are discretized with $\Delta t = 0.01$):

$$f\left(\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\right) = \begin{pmatrix} x + v_x\Delta t \\ y + v_y\Delta t \\ v_x + u_1\Delta t \\ v_y + u_2\Delta t \end{pmatrix}$$

The barrier state was then defined, as in (6), for the safe set $\mathcal{S}^\circ = \{x \in \mathbb{R}^2 | \bigcap_i^q h_i > 0\}$ for q obstacles where $h_i(x) = \|x - o_i\|^2 - r_i^2$ given o_i and r_i to be the center and radius of the i^{th} obstacle. For this systems, we used the inverse barrier formulation and imposed a quadratic cost on the terminal position and a quadratic cost on the barrier state w_k with coefficient q_w.

Fig. 3 shows that DBaS-DDP was able to successfully plan a collision-free course starting from the initial point $(0, 0)$ to the goal $(3, 3)$. Experiments on the effect of changing the penalization parameter q_w on the DBaS-DDP solution is also shown in Fig. 3. As should be expected, a very small q_w led to a more lenient solution, i.e. the robot was allowed to get closer to the obstacles but never hit them. It is worth mentioning that no high penalization was needed as the safety property was inherent in the dynamics used to generate the optimized trajectories.

To empirically test the results of Theorem 5, we compare the eigenvalues of $H_{uu_k}^{-1}$ and the conditioning number (defined by $\max_x \frac{\|H_{uu_k} x\|}{\|x\|}$) between the naive penalty method and DBaS-DDP. We find that for the penalty method in this point-mass method, we have a minimum eigenvalue (across an entire run of DDP) of -0.173, meaning that although the penalty method is able to find some non-intersecting path in this case, it is only despite numerical ill-conditioning. Comparatively, using DBaS the minimum eigenvalue is 0.1, meaning that H_{uu_k} is numerically well-conditioned across the entire sweep.

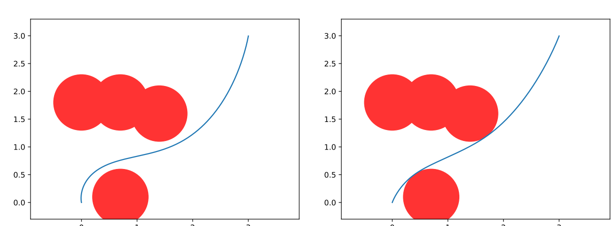

Fig. 3: Planar double-integrator problem with barrier state cost coefficients $q_w = 10^{-3}$ (left) and 10^{-7} (right) in which the red circles represent obstacles that should be avoided.

B. Differential Wheeled Robot

The system dynamics is given by

$$\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{\theta}\end{bmatrix}=\begin{bmatrix}v\cos\theta\\v\sin\theta\\\omega\end{bmatrix}$$

where x and y are the robots coordinates (position in 2D plane), θ is the angle of orientation of the robot, v is the translational speed and w is the rotational speed of the robot. The dynamics can be rewritten as

$$\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{\theta}\end{bmatrix}=\frac{1}{2}\begin{bmatrix}r\cos\theta & r\cos\theta\\r\sin\theta & r\sin\theta\\\frac{r}{d} & -\frac{r}{d}\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$

where r is the wheels radius (assuming the same radius for both wheels), d is the distance measured between the center of the wheels and u_1 and u_2 are the rotational speed of the right and light wheels respectively. The robot is to safely navigate a course with obstacles and arrive to the final desired point. For this problem, the parameters were $\Delta t = 0.02$, horizon $N = 800$, $q_w = 10^{-5}$ and $Q_f = 10I_{4\times4}$. Fig. 4 shows that the robot safely arrived to the target. It is worth mentioning that the naive penalty approach failed to successfully complete the task.

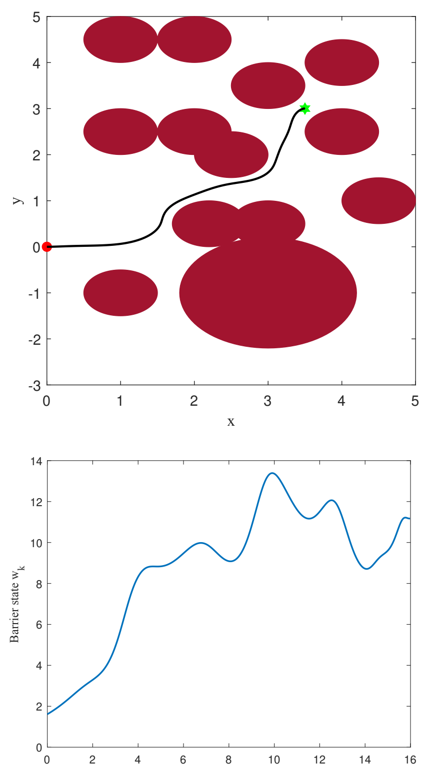

Fig. 4: Differential wheeled robot navigates through a course with obstacles (drak red) to be avoided. The robot successfully reached the target (green) starting from the red point. The bottom figure shows progression of the barrier state over time.

C. Single Quadrotor

1) Reaching Task: We applied the discrete barrier state based DDP (DBaS-DDP) to a quadrotor model as described in Sabatino [20]. The quadrotor was to perform a *reaching* problem safely, i.e. to fly from some initial state to some arbitrary final state in the presence of some obstacles without collision. To perform this task, we used a large quadratic terminal cost on the deviation of the final state from the goal and a small quadratic instantaneous cost on control inputs:

$$l(x,u,w)=\frac{1}{2}u^{\mathrm{T}}Ru+q_w w^2 \quad \Phi(x)=\frac{1}{2}x^{\mathrm{T}}Q_f x$$

Where $R = 0.1I$, $Q_f = 100I$, and $q_w = 10^{-4}$. The safe set is again defined as the complement of three spheres, each of which has center o_i and radius r_i, and thus the functions that define the safe set can be given by:

$$h_i(x,y,z)=(x-o_{i_x})^2+(y-o_{i_y})^2+(z-o_{i_z})^2-r^2$$

for $i = 1, 2, 3$ and x, y and z are the three-dimensional space vectors. By construction, whenever $h_i > 0\ \forall i$, the quadrotor is safe.

A solution to the quadrotor reaching problem found by DBaS-DDP is shown in Fig. 5.

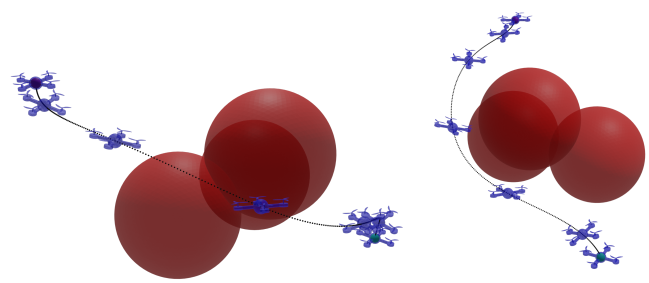

Fig. 5: Quadrotor reaching problem with three spherical obstacles (red) in the space. The quadrotor safely performs the reaching task starting from the initial position (green ball) to the final position (dark ball).

Next, we compared DBaS-DDP against the ordinary cost function penalty method to test the barrier-state formulation's robustness and superiority. For the penalty method we used a cost function similar to the DBaS-DDP cost function:

$$l(x,u,w)=\frac{1}{2}u^{\mathrm{T}}Ru+q_b\beta(h(x_k)) \quad \Phi(x)=\frac{1}{2}x^{\mathrm{T}}Q_f x$$

In these trials, the quadrotor was placed at a random position and given the task of reaching another position while avoiding obstacles. The reaching task was performed in two different environments, an *easy* one and a *hard* one. In the *easy* environment, the quadrotor must navigate around two moderately distanced spheres placed between its starting point and the target. This problem should not pose any difficulties to an algorithm, as it is essentially free of local minima in the cost landscape. In the *hard* environment, three spheres are placed such that the quadrotor can narrowly fit between them. This environment acts as a pathological case for constrained optimization problems, as the small area between the spheres makes the problem prone to local minima causing possible failures to reach the target. We define *success* by whether the final position of the planned path is within a small threshold, 0.5 meters, of the target position and we define *robustness* by the percentage of success.

Table II shows the results of this robustness test. DBaS-DDP was able to find a good trajectory for all of the easy cases, while the standard penalty method failed to find a trajectory that reached the goal in nearly 20% of the trials. The *hard* test posed much more difficulty to both methods. Nonetheless, the DBaS-based formulation was still able to find a valid solution 90% of the time. On the other hand, the penalty method completely failed to find a valid trajectory in the majority of these cases. We hypothesize that the second-order expansion in standard DDP easily becomes indefinite in the penalty case, causing the optimization procedure to fail to make progress towards a local optimum. Fig. 6 shows a random run for the hard environment experiment.

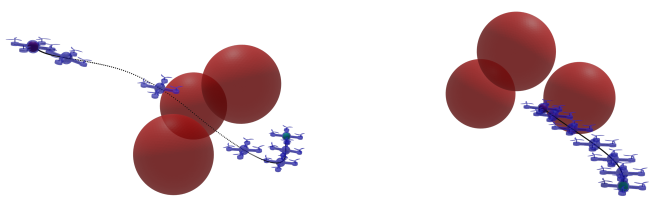

Fig. 6: Quadrotor reaching task with tight squeeze using DBaS-DDP (left) compared to the penalty method (right). Using the DBaS-DDP solver, the quadrotor was able to reach the goal (dark ball) safely while the penalty based DDP solver failed to navigate through the obstacles.

Experiment	Success (DBaS)	Mean Cost (DBaS)	Success (Penalty)	Mean Cost (Penalty)
Easy	100%		81%	
Hard	90%	86	37%	330

TABLE II: Results of robustness testing comparing DBaS-DDP to the penalty formulation.

2) Tracking Task: The technique of barrier states can also generalize to the *tracking* problem, in which we want to safely track some (possibly unsafe) reference trajectory. To put the DBaS-DDP to the test, we attempted to track the trajectory defined by the parametric equations:

$$x(s) = \sin(2s), y(s) = \cos(s), z(s) = 0$$

In these equations, s is given by $s(t) = \frac{(\pi t/25)^2}{\pi t/25+1}$. This form ensures that it ramps up smoothly from zero velocity at time $t = 0$. Then, the squared deviation of the quadrotor's trajectory from this path is penalized in the cost function. In addition, we place a spherical obstacle at the origin. Because the prespecified trajectory is unsafe, the quadrotor must navigate around the obstacle to remain in the safe set. Fig. 1 shows an execution trace from this experiment. The quadrotor was able to successfully track the trajectory in a very aggressive maneuver without losing safety.

D. Multiple Quadrotors

We also attempted an experiment in which the DBaS-DDP was used to safely control a system of two quadrotors to complete the same track starting from facing positions without colliding. A *barrier* was placed between the two quadrotors so that they maintain a safe distance of at least 0.5 meters from one another. That is, the safe set was defined by $h(x_k) = ||\breve{x}_1 - \breve{x}_2||^2 - 0.5^2$, where $\breve{x}_i = [x, y, z]_i^{\mathrm{T}}$ for $i = 1, 2$. Fig. 7 and Fig. 8 show the results of the safe multi-quadrotor tracking problem where the quadrotors faced each other two times but completed the track successfully without collision. Each quadrotor can be looked at as a moving obstacle but in a min-min framework.

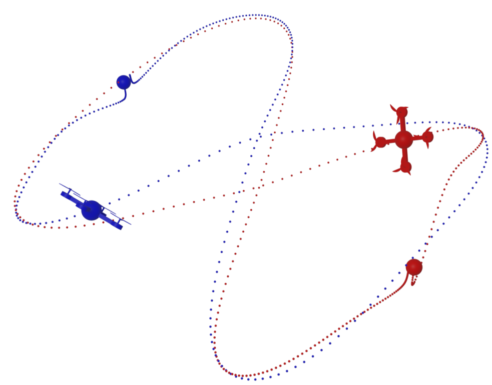

Fig. 7: Two quadrotors with a DBaS defined by a maximum allowed distance of 0.5 meters. The blue ball is the start and the end position for the blue quadrotor and similarly for the red ball and the red quadrotor. The quadrotors safely followed a similar, potentially unsafe predetermined track starting from opposite positions.

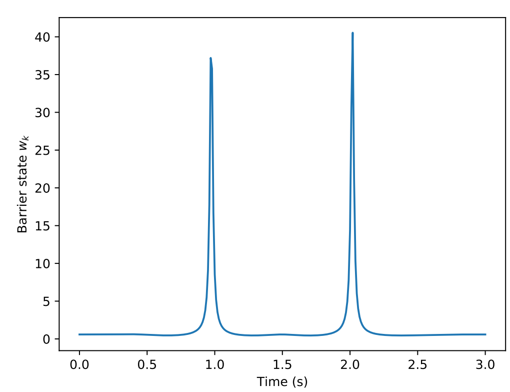

Fig. 8: Progression of the DBaS over time for the multi-quadrotor tracking problem.

VI. Conclusion

In this work, the newly proposed barrier state method for stabilization of continuous time systems was extended to stabilization and trajectory optimization of discrete time systems. The extension, named the discrete barrier state (DBaS) method, was shown to provide safety guarantees in stabilization tasks and validated with a numerical example. Additionally, the constructed DBaS was used in DDP framework to solve constrained DDP problems. The advantages and disadvantages enjoyed by the DBaS-DDP optimization method were also addressed. The safety embedded DDP was implemented on a 2D double integrator example as a proof of concept. Furthermore, to show the efficacy of the proposed safety embedded DDP, a variety of successful simulation examples of a quadrotor performing safety-critical planning and execution tasks was presented as well as a comparison with the penalty-method DDP.

A few limitations of the proposed work include the assumption of full knowledge of the dynamics of the safety-critical system which clearly affects the DBaS's dynamics. A future work that can address this is learning or quantifying the system's uncertainties, e.g. using Gaussian processes, which then will be included in the DBaS dynamics. Similarly, our method assumes full knowledge of the safety constraints. In practical applications, these constraints can be synthesized from sensor measurements, for example by detecting obstacles using motion or vision sensors. Additionally, the proposed DBaS-DDP uses DDP for planning and execution. It will be more practical to put the DBaS in an MPC-DDP framework to be applied on real-time systems, which is intended to be a next step for future work. Future work will also include considerations of adversarial actions by applying the DBaS-DDP to min-max optimal control problems as well as risk sensitive DDP. Implementing the proposed work on hardware is also a planed future work where MPC-DDP will be used.

References

[1] 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*, 2017.

[2] Mohamadreza Ahmadi, Andrew Singletary, Joel W Burdick, and Aaron D Ames. Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4797–4803. IEEE, 2019.

[3] Hassan Almubarak, Nader Sadegh, and Evangelos A. Theodorou. Safety Embedded Control of Nonlinear Systems via Barrier States. In *arXiv preprint arXiv:2102.10253*, 2021.

[4] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

[5] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[6] Yuichiro Aoyama, George Boutselis, Akash Patel, and Evangelos A Theodorou. Constrained Differential Dynamic Programming Revisited. *arXiv preprint arXiv:2005.00985*, 2020.

[7] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization*. 2020.

[8] Franco Blanchini. Set invariance in control. *Automatica*, 35 (11):1747–1767, 1999.

[9] Jason Choi, Fernando Castaneda, Claire J Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. In *Robotics: Science and Systems*, 2020.

[10] T. A. Howell, B. E. Jackson, and Z. Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679, 2019. doi: 10.1109/IROS40897.2019.8967788.

[11] David H. Jacobson. *Differential dynamic programming methods for determining optimal control of non-linear systems*. PhD thesis, Electrical Engineering, Centre for Computing and Automation, Imperial College of Science and Technology, University of London., 1967.

[12] David H Jacobson and David Q Mayne. *Differential dynamic programming*. North-Holland, 1970.

[13] 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.

[14] Daniel M Murray and Sidney J Yakowitz. Constrained differential dynamic programming and its application to multireservoir control. *Water Resources Research*, 15(5):1017–1027, 1979.

[15] Andrei Pavlov, Iman Shames, and Chris Manzie. Interior point differential dynamic programming. *IEEE Transactions on Control Systems Technology*, 2021.

[16] B. Plancher, Z. Manchester, and S. Kuindersma. Constrained unscented dynamic programming. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5674–5680, 2017. doi: 10.1109/IROS.2017.8206457.

[17] Stephen Prajna. Barrier certificates for nonlinear model validation. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 3, pages 2884–2889. IEEE, 2003.

[18] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *International Workshop on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

[19] Muhammad Zakiyullah Romdlony and Bayu Jayawardhana. Uniting control Lyapunov and control barrier functions. In *53rd IEEE Conference on Decision and Control*, pages 2293–2298. IEEE, 2014.

[20] F. Sabatino. Quadrotor control: modeling, nonlinearcontrol design, and simulation. 2015.

[21] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175, 2014. doi: 10.1109/ICRA.2014.6907001.

[22] Andrew J Taylor and Aaron D Ames. Adaptive safety with control barrier functions. In *2020 American Control Conference (ACC)*, pages 1399–1405. IEEE, 2020.

[23] Li Wang, Evangelos A Theodorou, and Magnus Egerstedt. Safe learning of quadrotor dynamics using barrier certificates. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2460–2465. IEEE, 2018.

[24] Peter Wieland and Frank Allgöwer. Constructive safety using control barrier functions. *IFAC Proceedings Volumes*, 40(12): 462–467, 2007.

[25] 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.