C106B Discussion 10: Control Barrier Functions

1 Introduction

Today, we'll talk about:

1. Defining safe sets
2. Control barrier functions
3. Control barrier function quadratic programs

If you're interested in learning more about this material, we highly recommend reading *Control Barrier Functions: Theory and Applications.*

2 Defining Safe Sets

Many systems in robotics, such as autonomous aircraft and vehicles, are *safety-critical* systems. When dealing with these systems, it's important that we're able to formally *prove* that our systems will be stable! To accomplish this, we'll need a few definitions.

Definition 1 ***Safe Set***
The safe set of a system, $\mathcal{C}$, is the set containing all of the state vectors $x \in \mathbb{R}^n$ where the system is said to be safe.

$$\mathcal{C} \subseteq \mathbb{R}^n \tag{1}$$

For high dimensional nonlinear systems, these safe sets can be challenging to visualize! Can we come up with a simple *function* of the state vector that helps us describe if our system will be safe? Consider the following rules for a function $h(x) : \mathbb{R}^n \to \mathbb{R}$ and a safe set $\mathcal{C}$.

$$h(x) \geq 0,\ x \in \mathcal{C} \tag{2}$$
$$h(x) = 0,\ x \in \partial\mathcal{C} \tag{3}$$
$$h(x) > 0,\ x \in int(\mathcal{C}) \tag{4}$$

When the state vector is *inside* or on the boundary of the safe set, the function $h(x)$ should be greater than or equal to zero. If the state vector x is *outside* of the safe set, the function $h(x)$ should be less than zero! This gives us a condition to test if a state vector x is *safe* or *unsafe*!
Furthermore, to gain access to some nice mathematical properties further down the line, we'll assume that the function h is *continuous* and *differentiable* with respect to the state vector x.

Problem 1: Consider the following scenario. Imagine we have a turtlebot with a state vector $q \in \mathbb{R}^3$:

$$q = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \tag{5}$$

Where (x, y) are the coordinates of the center of the turtlebot and ϕ is the heading angle. Suppose we have a circular obstacle of radius r_o located at (x_o, y_o). If we consider collisions with the obstacle *unsafe*, come up with a function $h(q)$ according to the rules above that determines if the turtlebot is in a safe or unsafe set. *Note: You may ignore the radius of the turtlebot for this simple example.*

3 Control Barrier Functions

Can we somehow use the $h(x)$ function that encodes the safe set of our system to *guarantee* our system remains within the safe set for all time? Let's think about some ways we can ensure $h(x) > 0$.
Let's imagine we have a control affine system of the form:

$$\dot{x} = f(x) + g(x)u \tag{6}$$

We'd like to somehow figure out an input u to the system that keeps the system inside the safe set. To do this, we may identify a *constraint* on the system that involves u and enforces safety. Currently, our expression for $h(x)$ has no input in it, yet the expression for $\dot{x}$ does! We know $\dot{h}(x) = \frac{\partial h}{\partial x}\dot{x}$ involves $\dot{x}$ - let's try working with this expression.
Let's consider the following differential equation:

$$\dot{h} = -\gamma h \tag{7}$$

We know that the solution to this equation is given by $h(t) = \exp(-\gamma t)h_0$, and that this function is *always* greater than zero for $h_0 \geq 0$! Thus, if we enforce the constraint:

$$\dot{h} \geq -\gamma h \tag{8}$$

We'll get that $h(t) \geq \exp(-\gamma t)h_0$. Now, all that we need to do is bring the input into this constraint! Taking the derivative of h along the trajectories of the system, this gives us the constraint:

$$L_f h + L_g h u \geq -\gamma h \tag{9}$$

This leads us to the following definition:

Definition 2 ***Control Barrier Function (Informal Definition)***
A function $h(x)$ that encodes the safe set of the system is called a control barrier function if there exists an input u and a constant $\gamma > 0$ such that for all $x \in \mathcal{C}$:

$$L_f h + L_g h u \geq -\gamma h \tag{10}$$

Thus, if we can satisfy this constraint, we'll be able to *guarantee* the safety of the system!

Problem 2: The turtlebot system has dynamics:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{11}$$

If a control barrier function for this system is $h(q) = x - x_{safe}$, write out the constraint:

$$\dot{h} \geq -\gamma h \tag{12}$$

4 The CBF-QP Controller

Now, we have a constraint that guarantees us the safety of our system (provided we can find an appropriate input). How can we actually identify what that input might be? When finding an input to keep our system in the safe set, we have a few criteria.
Our system might have other tasks, such as trajectory tracking, that we'd like it to execute. We'd like to ensure our system is kept safe while still being able to track desired trajectories and perform other similar tasks. To accomplish this goal, we formulate the control barrier function quadratic program (CBF-QP) controller.

Definition 3 ***CBF-QP Controller***
Suppose we have a system $\dot{x} = f(x) + g(x)u$ *with a control barrier function* $h(x)$ *and a nominal controller* $k(x)$*. To find an input to the system that guarantees safety while allowing for trajectory tracking, we may solve the optimization problem:*

$$u_{safe} = \arg\min_{u \in U} ||u - k(x)||^2 \tag{13}$$

$$s.t.\ L_f h + L_g h u \geq -\gamma h \tag{14}$$

Note that for a fully nonlinear system $\dot{x} = f(x, u)$, we would simply use a barrier constraint of the form $\dot{h} \geq -\gamma_h$, as we can no longer take Lie derivatives to get the derivative of h along the trajectories of the system.