**Coursework 3: Control Barrier Functions and Control Lyapunov Functions for Obstacle Avoidance**

First Student ID: 11537608

Second Student ID: 14131209

# Task 2.1

So, the obstacle is defined by:

# Task 2.2

For the first derivative, We use the chain rule:

In simplified form:

Second derivative:

Since , so we plug these in:

Simplify:

Barrier Function Recap:

0th Derivative (just B):

1st Derivative:

Still no direct dependence on a or w.

2nd Derivative:

Now both control inputs a and w appear explicitly so The relative degree of B with respect to control inputs a and w is 2.

**Why Do We Need a Higher-Order Control Barrier Function (HOCBF)?**

Because:

* Standard Control Barrier Functions (CBFs) are only suitable when the relative degree is 1, meaning control inputs appear in the first derivative of the barrier function.
* In our case, the inputs only show up in the second derivative ().

Therefore, to ensure that the system remains in the safe set B(x,y) ≥ 0, and to incorporate the control inputs aaa and ω into the safety condition, we must use a Higher-Order Control Barrier Function (HOCBF).

# Task 2.3

If goes to zero:

If the term goes to zero, robot's orientation 0 aligns with the direction pointing toward the goal.

If the term goes to zero, robot's position converges to the goal (

So, if goes to zero, it implies that robot is at the goal or facing towards the goal.

If goes to zero:

v goes to , so if goes to zero, The robot's speed matches the desired cruising speed.

When both and approach 0, the robot reaches the goal position (), is oriented toward the goal direction, and moves at the nominal velocity of 1 m/s.

Derivative of

Recall , where a is linear acceleration.

Derivative of :

Then:

Now take the derivative:

So

So depends on both control inputs: w and v

Allows for direct control over the evolution of the Lyapunov function

Why can't we just use a distance-only CLF like ?  
If we define:

Then:

This depends only on v, not on w.

That means you cannot influence the orientation via control, and thus:

The robot might fail to orient properly

* It might approach the goal from the wrong direction
* It might even circle or oscillate around the goal without stabilizing its heading
* A distance-only CLF is insufficient because it cannot stabilize both position and orientation.

The chosen V1V\_1V1​ incorporates orientation error and ensures the robot is aligned with the goal direction.

# A graph of a graph showing a red circle and a green line AI-generated content may be incorrect.A graph of a graph showing a red circle and a green line AI-generated content may be incorrect.A graph of a graph showing a red circle and a blue line AI-generated content may be incorrect.Task 3.1

Fig. 1: Fig. 1: Fig. 1:

A graph of a graph showing a line of a path

AI-generated content may be incorrect.A graph of a graph showing a red circle and a blue line

AI-generated content may be incorrect.A graph of a graph showing a line and a red circle

AI-generated content may be incorrect.

Fig. 1: Fig. 1: Fig. 1: 00

The parameter k1k\_1k1​ in the CBF constraint directly impacts the size and enforcement strength of the forward-invariant set, which is the set of states where the robot is guaranteed to remain safe (i.e., not violate the safety constraint B(x) ≥ 0.

From to :

The robot cuts close to the obstacle but still respects the safety boundary and the forward-invariant set is relatively large, meaning the system tolerates being closer to the boundary. The CBF constraint is less aggressive, allowing CLF (goal-seeking) behavior to dominate more.

From to :

The robot path becomes more conservative, staying farther from the obstacle.The forward-invariant set shrinks, as the system must maintain a stricter distance from the obstacle and there's more deviation from the direct goal path to respect safety.

:

The robot trajectory becomes unstable or erratic because the safety constraint is so dominating that the QP becomes numerically stiff or infeasible in places, and the robot may go far off course. This indicates that too high a ​ can destabilize the controller, reduce feasibility, and shrink the forward-invariant set so much that the robot avoids safe but goal-efficient regions.

Based on the figures, offers the best balance between safety and goal-reaching efficiency. At this value, the robot consistently avoids the obstacle while following a smooth and direct trajectory toward the goal. Lower values like or result in minimal deviation but bring the robot dangerously close to the obstacle, which weakens safety guarantees. In contrast, very high values like or lead to over-conservatism or instability, with the robot veering significantly off path or exhibiting erratic behavior due to the dominance of the safety constraint in the optimization. The key trade-off lies between safety enforcement and task performance: too low compromises safety, while too high ​ sacrifices goal efficiency or feasibility. Thus, provides practical middle ground, ensuring the robot stays within the forward-invariant set without unduly compromising its ability to reach the target.