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

First Student ID: Jiaxin Zhou (1537608)

Second Student ID: Armita Hemmati(14131209 )

# Task 2.1

So, the obstacle is defined by:

# Task 2.2

For the first derivative, We use the chain rule:

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 a 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 :

So

So depends on both control inputs: w and v and 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. As the result the robot might fail to orient properly and it might approach the goal from the wrong direction or 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 ​ incorporates orientation error and ensures the robot is aligned with the goal direction.

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

Fig. 1: Fig. 2: Fig. 3:

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. 4: Fig. 5: Fig. 6: 00

The parameter 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.

At very low values like , the robot path remains overly cautious near the obstacle and fails to reach the goal, as the safety condition is too weakly enforced to maintain both progress and constraint satisfaction. From , 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. At , 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.

# Task 3.3

# Task 3.4

Penalty parameter is to make the system weigh the Control Lyapunov Function (CLF)and safety constraint (CBF). In other words, it determines how much the system needs to satisfy the soft constraint. When is 0.01, the system satisfies the hard constraint by sacrificing CLF, which causes the system to be too slack to reach the target position even though the system is in a safe distance. As equals to 10, the system finds the balance between the constraint. After bypassing the obstacle, the robot still moved a little bit around the obstacle, making the path gentler and the speed more stable. Continue to increase to 5000, it put more emphasis on the CLF, which result in that it modifies the path after bypass the obstacle and finally goes straightly to the target. However, too much large can leads no solution to the system, the result of =8000 is shown in figure. In conclusion, the less is, the more system will slack on the CLF constrain.

A graph with a red circle and a blue line

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

AI-generated content may be incorrect.

Fig. 7: Fig. 8:

A graph with a red circle and green line

AI-generated content may be incorrect.图表, 折线图

AI 生成的内容可能不正确。

Fig. 9: Fig. 10:

CLF is a stable function which can lead the robot finally get to target and determine the convergence speed of this function. In other words, mainly used to adjust the speed and acceleration of the robot. From the Figure, when , robot is very conservative and almost only wanders around the edge of obstacle’s safe range rather than target. As the grows up to 0.1, it can make robot towards the right direction, but the speed of convergence is not enough, which means it still to be increased. As the , the robot starts directly from the initial point and bypasses the obstacle, then quickly approaches the target point. It is more efficient, and path is shorter than the previous trajectory. Finally, when is too large, the robot is uncontrollable, trajectory is extremely far from the point and obstacle. The reason for this phenomenon is that it is so aggressive for the CLF, which forces system to convergence in an unrealizable speed and controller output an extreme input to the robot. Hence, setting up an appropriate value of can make system have good performance on driving operation.

A graph with a red circle and green line

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

AI-generated content may be incorrect.

Fig. 11: Fig. 12:

A graph with a blue line and green dot

AI-generated content may be incorrect.图表, 折线图

AI 生成的内容可能不正确。

Fig. 13: Fig. 14: