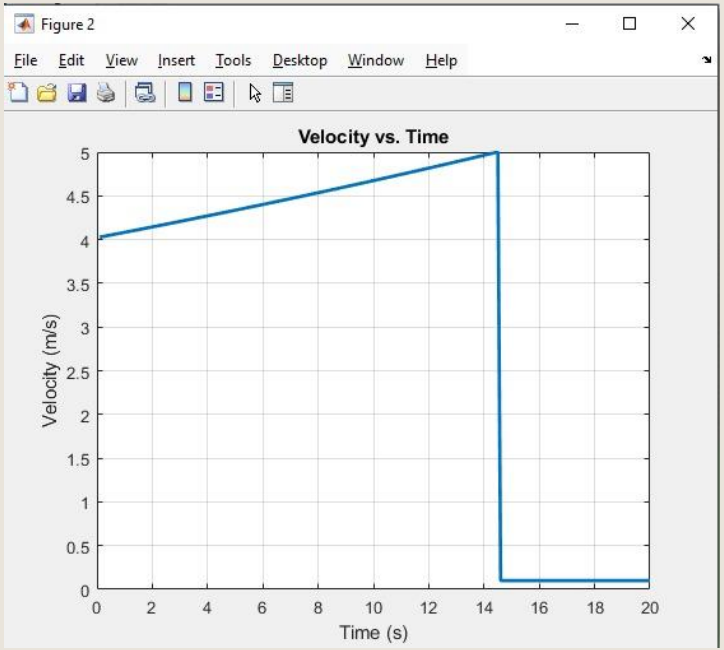
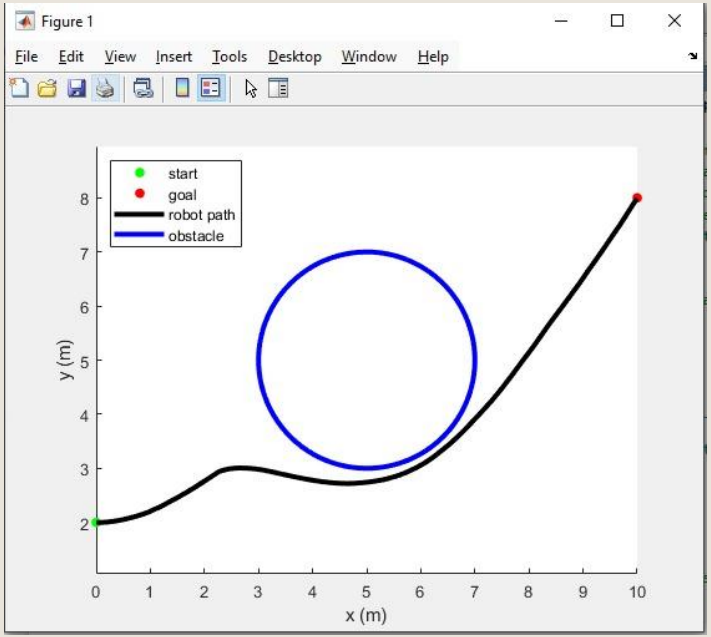
Velocity Control

kappa = 10



Obstacle Avoidance

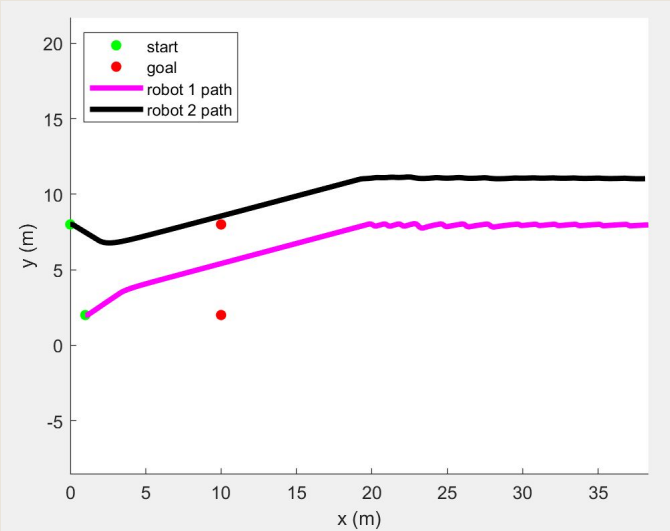
safe_distance = 3.0
kappa = 10



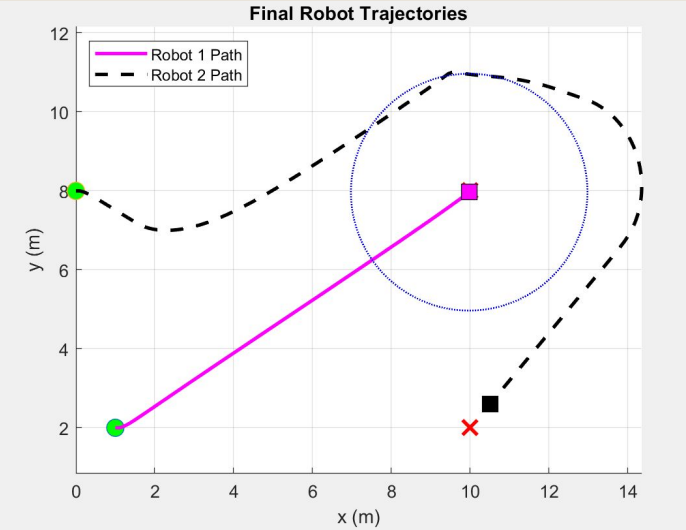
Two Robots

Two Robots Criss-Crossing

safe_distance = 3.0
kappa = 5



Without priority



With priority

Operation Flow

Pick who goes first → Dynamic Priority (simple distance check)

- Measure how far each active robot is from its goal.
- Closest = **Priority 1**, next = **Priority 2**, farthest = **Priority 3**.

Aim for the goal → Nominal Controller (pure pursuit)

- For each robot, compute the turn it *wants* to point straight at its goal.

Check safety → Control-Barrier Functions (CBF / HOCBF)

- Robot-to-obstacle: stay outside the red circle.
- Robot-to-robot: stay at least *safe distance* apart.

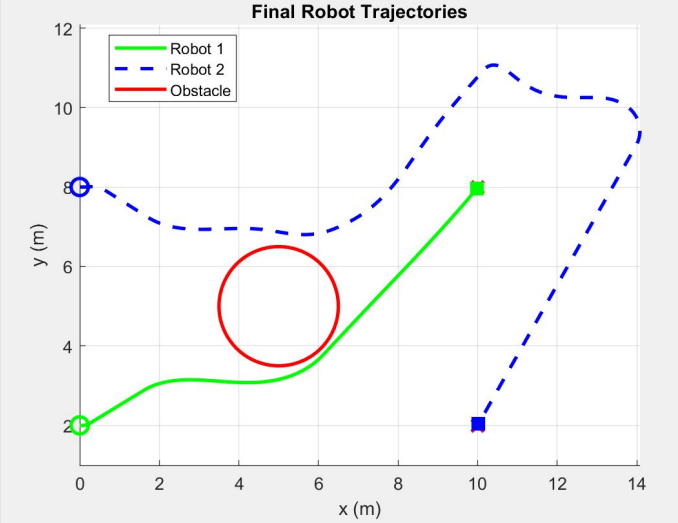
Adjust the turns → Quadratic Programming (QP with priority weights)

- **Objective:** penalizes deviation from the nominal turn rates. Higher-priority robots keep more of their original turn; lower-priority ones give way.
- **Constraints:** Affine CBF inequalities that enforce robot–robot and robot–obstacle safety margins.

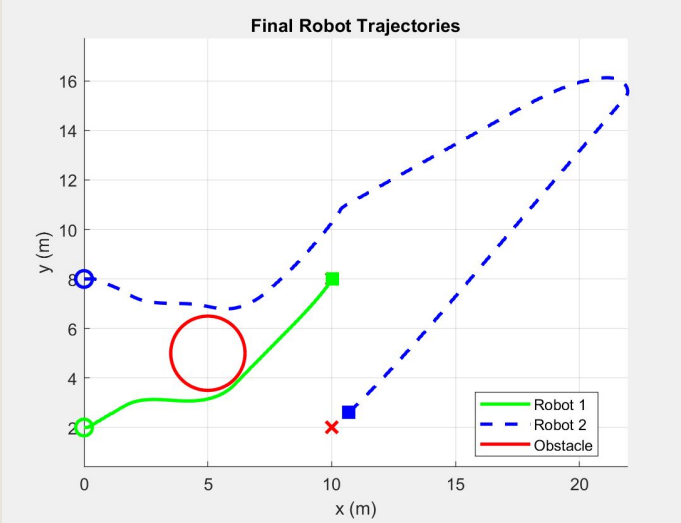
Two Robots

Two Robots Criss-Crossing with Obstacle

safe_distance = 3.0
kappa = 10



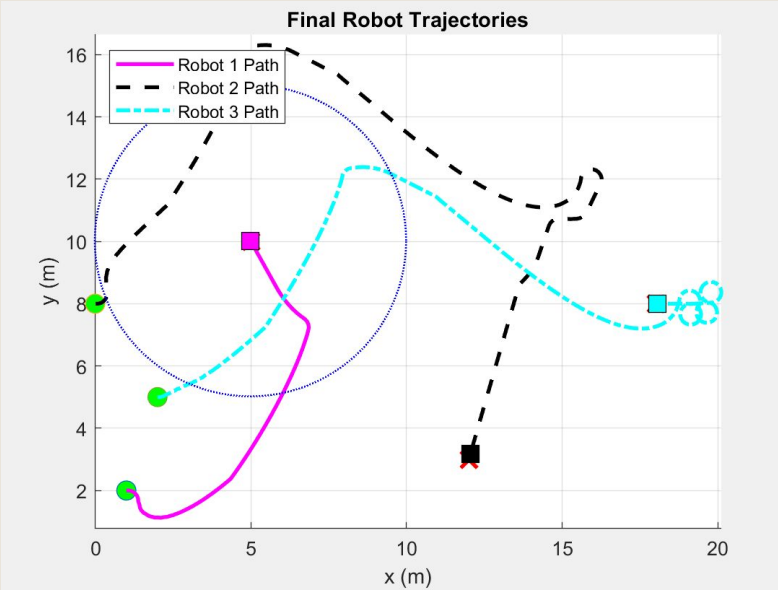
safe_distance = 3.0
kappa = 30



To create weird behaviors:
(1) Adjust kappa (higher → overshooting)
(2) Increase safe distance between robots

Three Robots Criss-Crossing

safe_distance = 5.0
kappa = 10



Three Robots Criss-Crossing with Obstacle

safe_distance = 3.0
kappa = 10