

Last Time:

- formalizing safe sets
- computing safety filters

Lecture 4

EAIS S'25

Andrea Bajcsy

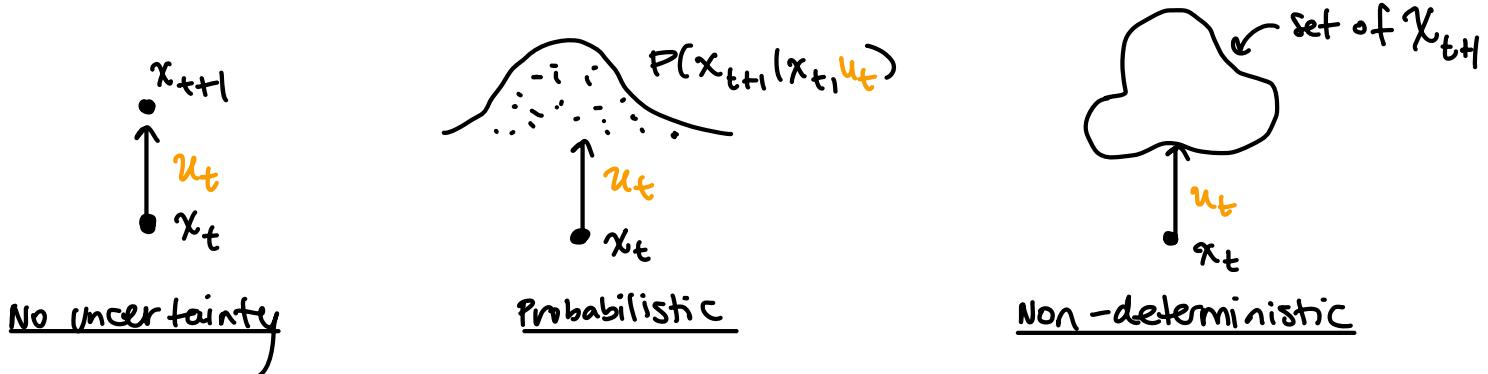
This Time:

- robustifying safety - why?
- zero-sum dynamic games
- dynamic programming for games

Robustifying Safety

So far, we have assumed that our dynamical system perfectly evolves via $\dot{x} = f(x, u)$ with no uncertainty.

This isn't realistic for many real-world scenarios (e.g. friction!)

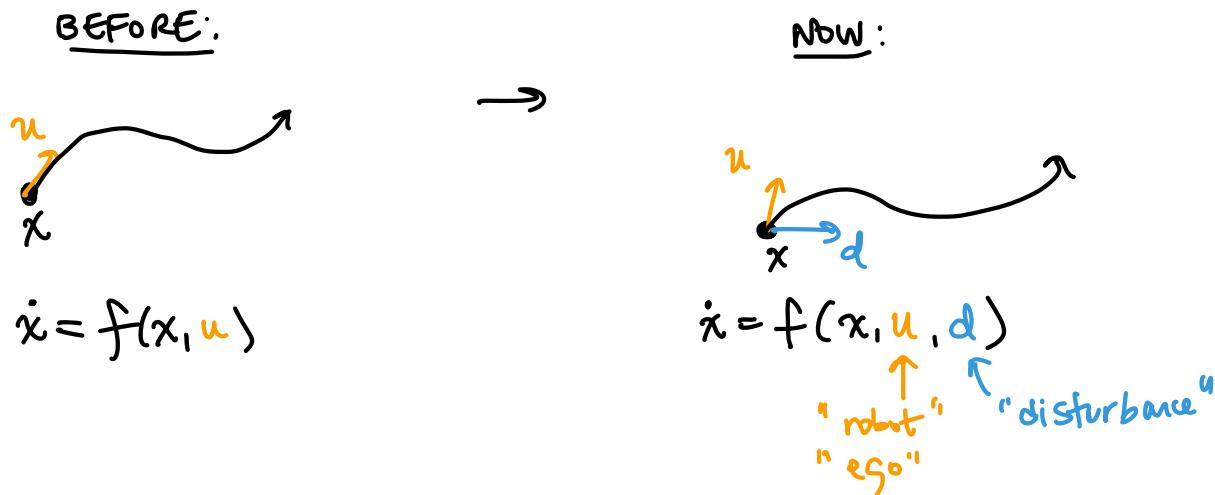


There are two ways to model uncertainty:

- 1) probabilistic uncertainty (i.e. "I have observed data")
- 2) non-deterministic uncertainty (i.e. "I have minimal additional info.")

How should we handle the design of our safety filter (+ analysis) to handle uncertainty?

Typically, we do this via modelling another "input" that influences the state evolution:



In robust safety, we take a non-deterministic view of uncertainty and assume that $d \in D$ is chosen from some bounded set and we want our robot to be ROBUST to the WORST POSSIBLE sequence of d 's!

ROBUST BACKWARDS REACHABLE TUBE (BRT) of a set $F \subset X$

and dynamical system $\dot{x} = f(x, u, d)$ is:

$$BRT(t) := \{x \in X : \forall u(\cdot) \in U_t^T, \exists d(\cdot) \in D_t^T, x_{x_t}^{u,d}(\tau) \in F \text{ for some time } \tau \in [t, T]\}$$

for all things the robot could do there is something the disturbance could do

that leads the robot to failure

This is the set of all starting states from which no matter the controller's effort, the disturbance can push system into F . The way we will formulate an "optimal control" problem whose solution represents this unsafe set will be via:

<u>zero-sum</u> there is a winner + loser	<u>dynamic</u> game <u>evolves</u> over time	<u>games</u> result / outcome depends on 2+ players / inputs
--	---	---

Our "game" formulation could look something like this:

$$V(x, t) = \max_{u(\cdot) \in \mathcal{U}_t^T} \min_{d(\cdot) \in \mathcal{D}_t^T} J(x, u(\cdot), d(\cdot))$$

s.t. $\dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau)) \quad \forall \tau \in [t, T]$

$$u(\tau) \in \mathcal{U}$$
$$d(\tau) \in \mathcal{D}$$

Here, the robot (u) is trying to maximize the objective $J(\cdot, \cdot, \cdot)$ and the other player (d) is minimizing. In other words, the robot is optimizing the worst-case objective.

Importance of Information Patterns

When we have 2 players reacting to each other, their optimal strategy will depend on what information they each have access to.

Example Suppose there are 2 boxes, each with 2 slots. Each slot contains prize money. Player A (You u) wants to maximize prize money while Player B (competition organizer) wants to minimize Player A's prize money.

Box 1	
s1	s2
2,000	10

Box 2	
s1	s2
1	6,000

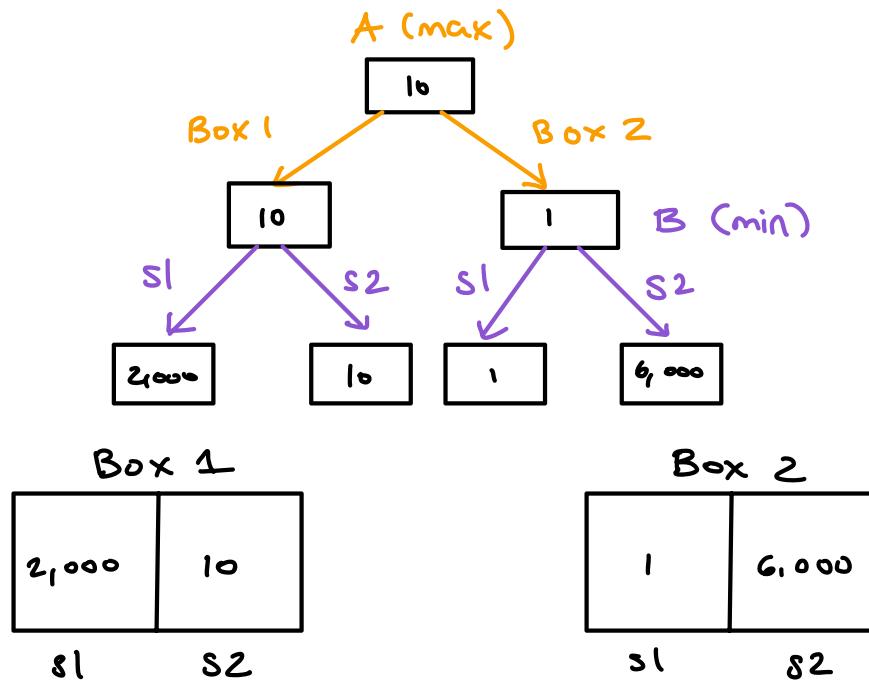
Player A

$u \in [\text{Box 1}, \text{Box 2}]$
you choose Box \rightarrow

Player B

$d \in [\text{slot 1}, \text{slot 2}]$
they choose slot \uparrow

Suppose Player A goes first:



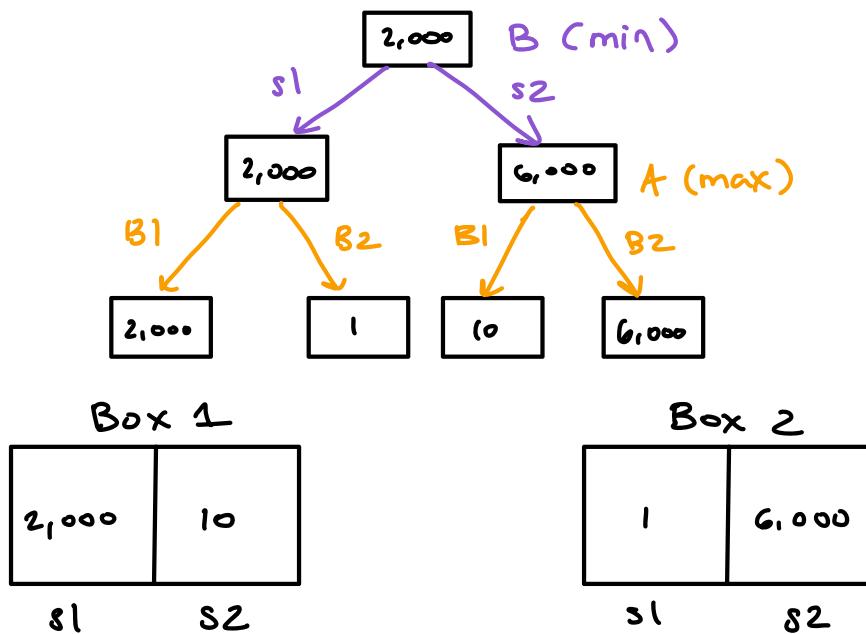
Best outcome for A is to pick Box 1 & get Rew = \$10.

Mathematically :

$$\max_{u \in [B1, B2]} \left(\min_{d \in [s1, s2]} J(u, d) \right) = 10$$

"computed 2nd" "computed first" but "plays 2nd" \Rightarrow "best response" strategy
 but "plays first"

Suppose Player B goes first:



Best strategy for player B is to pick slot 1 and pay player A a reward of \$ 2,000.

$$\min_{d \in [s_1, s_2]} \left(\max_{u \in [B_1, B_2]} J(u, d) \right) = \$2,000$$

This phenomenon we just saw can be stated via

minimax inequality:

$$\max_a \left(\min_b J(a, b) \right) \leq \min_b \left(\max_a J(a, b) \right)$$

Von Neumann 1928: equal when A, B are compact, convex sets +

$J(\cdot, b)$ is concave for fixed b

$J(a, \cdot)$ is convex for fixed a

In dynamic games the outcome depends on WHEN and

WITH WHAT INFORMATION

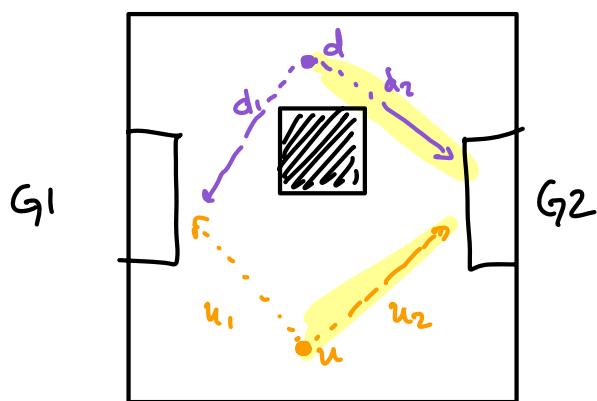
each player decides their inputs

NOW.

just talked about this
(order of play)

OPEN-LOOP INFO PATTERN:

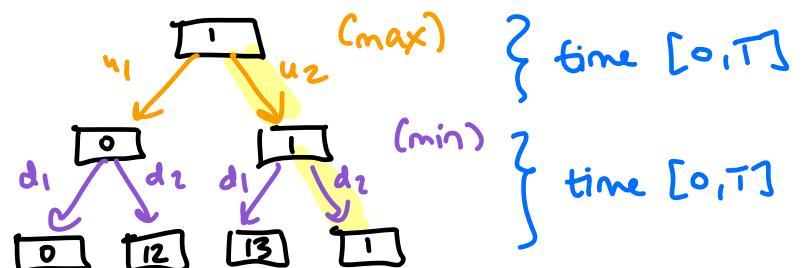
$$\max_{u(\cdot) \in U_t^T} \min_{d(\cdot) \in D_t^T} J(x, u(\cdot), d(\cdot))$$



$u(\cdot)$ "declares" entire signal and then $d(\cdot)$ gets to see this and interrupts!

⊕ OVERLY PESSIMISTIC!

Here u wants to reach G_1 or G_2 without being intercepted by d . BUT d wants to intercept u .

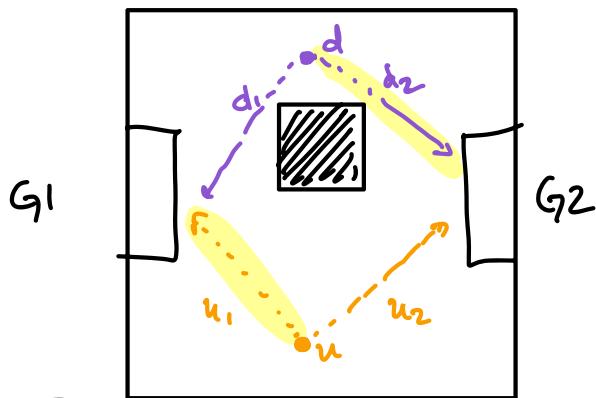


$J \rightarrow$ "dist btwn. agents"

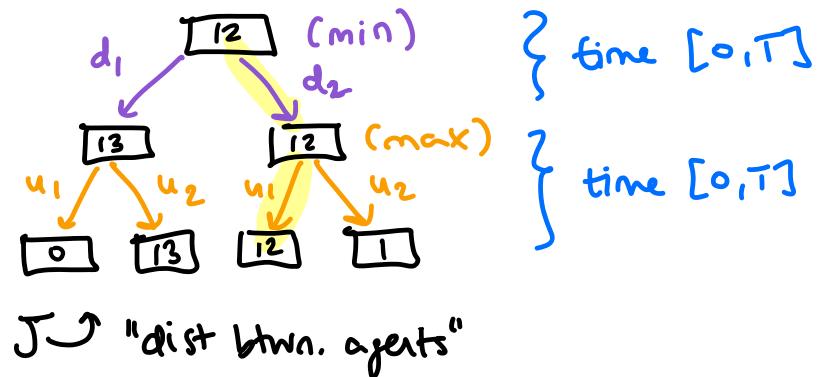
Let's swap the order of play again, but we are playing over continuous signals (or sequences of decisions):

OPEN-LOOP INFO PATTERN:

$$\min_{d(\cdot) \in D_t^T} \max_{u(\cdot) \in U_t^T} J(x, u(\cdot), d(\cdot))$$



$d(\cdot)$ "declares" entire signal
and then $u(\cdot)$ gets to see this
and picks the other goal!



↗ OVERLY OPTIMISTIC!

CLOSED-LOOP INFORMATION PATTERNS

The above formulation is not suitable for many practical systems.
We would like the controller (u) to ADAPT over time as system evolves, BUT respect the fact that @ any time t , we only have information up to time t .

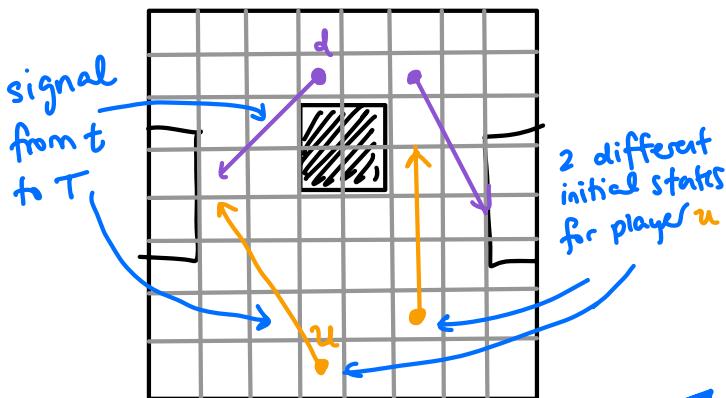
We can model this by solving for feedback policies:

$$V(x, t) = \max_{\pi_u} \min_{\pi_d} J(x, \pi_u(\cdot), \pi_d(\cdot))$$

$\pi_u: X \rightarrow U$
 $\pi_d: X \rightarrow D$

Before ($\max_{u(\cdot) \in U_t^T} \min_{d(\cdot) \in D_t^T} \dots$)

$$\max_{u(\cdot) \in U_t^T} \min_{d(\cdot) \in D_t^T} \dots$$

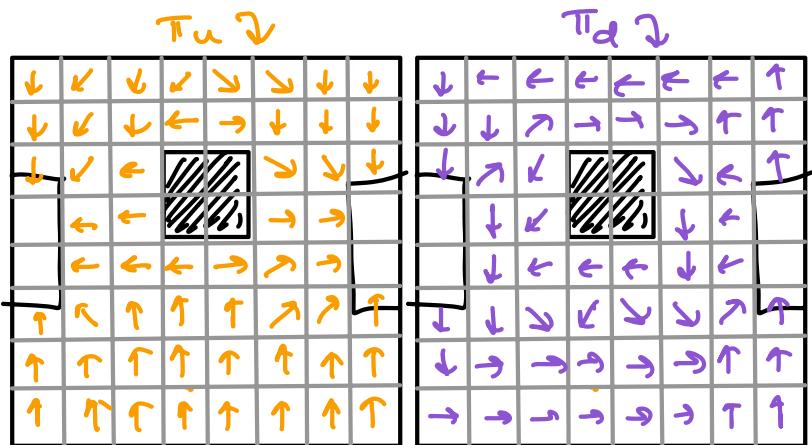


from any initial state
(denoted by • icon), each agent has to commit to entire control signal or sequence that they cannot change in future.

⇒ this is open-loop

Now ($\max_{\pi_u} \min_{\pi_d} \dots$)

$$\max_{\pi_u} \min_{\pi_d} \dots$$



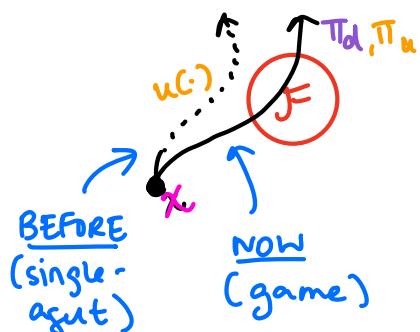
for each initial state, the players commit to a policy - or a map from each state to their action.

⇒ this is closed-loop

④ NOTE: for this illustrative example, since u and d are 2 separate agents, the state space is $X := X_u \times X_d$ and policies map from where BOTH players are, to actions:

$$\pi_u(x_u, x_d) \rightarrow u, \pi_d(x_u, x_d) \rightarrow d$$

Bringing this back to safety analysis, we just have to choose a objective function which lets us remember the closest we ever got to failure.



$$J(x, \pi_u, \pi_d, t) := \min_{t' \in [t, T]} l(x_{\pi_u \pi_d}^{t, t'}(t'))$$

minimum future distance influenced by u and d

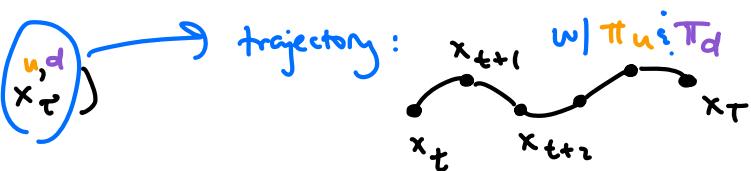
We can apply the game-theoretic principle of optimality called the Tenet of Transition by Rufus Isaacs:

"If play proceeds from one position (state) to a second, and V is thought of as known to the second, then it is determined at the first by demanding that players optimize (i.e. make minimax) the increment of V during the transition."

example (discrete-time)

$$\begin{aligned}
 V_t(x_t) &:= \max_{\pi_u(x)} \min_{\pi_d(x)} \min_{\tau \in \{t, \dots, T\}} l(x_\tau^{u,d}) \\
 &= \max_{u^t} \min_{d^t} \dots \max_{u^T} \min_{d^T} \min_{\tau \in \{t, \dots, T\}} l(x_\tau^{u,d}) \\
 &= \max_{u^t} \min_{d^t} \min \left\{ l(x_t), \underbrace{\max_{u^{t+1}} \min_{d^{t+1}} \dots \max_{u^T} \min_{d^T} \min_{s \in \{t+1, \dots, T\}} l(x_s^{u,d})}_{:= V_{t+1}(f(x_t, u_t, d_t)) \text{ by } \underline{\text{Tenet of Transition}}} \right\} \\
 &= \max_{u^t} \min_{d^t} \min \left\{ l(x_t), V_{t+1}(f(x_t, u_t, d_t)) \right\} \\
 &= \min \left\{ l(x_t), \max_{u^t} \min_{d^t} V_{t+1}(f(x_t, u_t, d_t)) \right\}
 \end{aligned}$$

Applying this to the continuous or discrete-time zero sum safety critical games we formulated above, we get two key equations that allow us to extend dynamic programming tools to the robust (dynamic game) setting:



Hamilton-Jacobi-Issacs Variational Inequality (HJI-VI)

$$\min \left\{ \ell(x) - V(x, t), \frac{\partial V}{\partial t} + \max_{u \in U} \min_{d \in D} \frac{\partial V}{\partial x} \cdot f(x, u, d) \right\} = 0$$

$$V(x, T) = \ell(x)$$

↑ add this ? solve "subgame" over just instantaneous actions

ROBUST SAFETY BACKUP (discrete-time):

$$V_t(x_t) = \min \left\{ \ell(x_t), \max_{u_t} \min_{d_t} V_{t+1}(f(x_t, u_t, d_t)) \right\}$$

$$V_T(x_T) = \ell(x)$$

SUMMARY

Continuous-Time

Problem

$$V(x, t) := \max_{\Pi_u(x)} \min_{\tau' \in [t, T]} \ell(x^u(\tau'))$$

$\min_{\Pi_d(x)}$ for robust

Dynamic Programming "Backup"

$$\min \left\{ \ell(x) - V(x, t), \frac{\partial V(x, t)}{\partial t} + \max_{u \in U} \frac{\partial V(x, t)}{\partial x} \cdot f(x, u) \right\} = 0$$

$\min_{d \in D}$ for robust

Discrete - Time

Problem

$$V_t(x_t) := \max_{\Pi_u(x)} \min_{\tau' \in \{t, \dots, T\}} \ell(x^u(\tau'))$$

$\min_{\Pi_d(x)}$ for robust

Dynamic Programming "Backup"

$$V_t(x_t) = \min \left\{ \ell(x_t), \max_{u_t \in U} V_{t+1}(f(x_t, u_t)) \right\}$$

$\min_{d_t \in D}$ for robust

Numerical Comparison of Robust vs. Non-Robust Unsafe Set

Recall: state is (x, y, θ) of Dubins' car

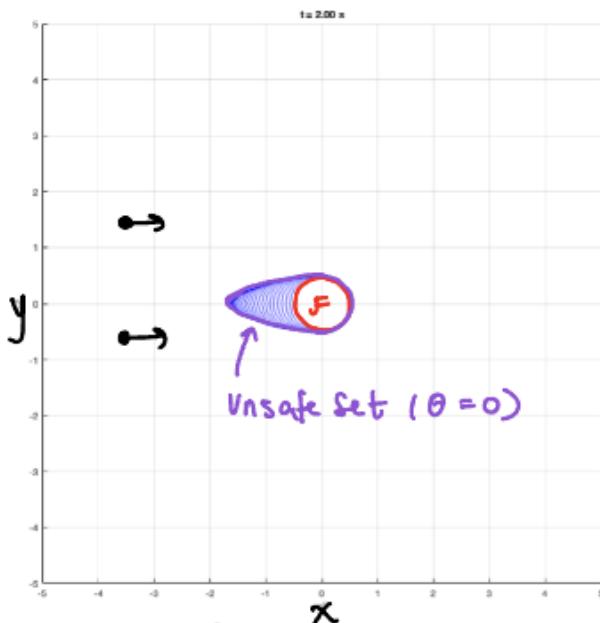
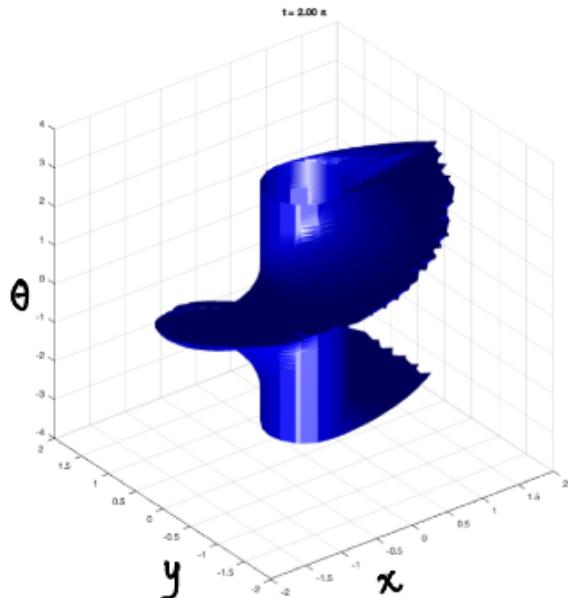
dynamics are

$$\begin{aligned}\dot{x} &= v \cos \theta + \frac{dx}{dt} & \in [-0.5, 0.5] \\ \dot{y} &= v \sin \theta + \frac{dy}{dt} & \equiv D \\ \dot{\theta} &= u & \in [-0.5, 0.5] \\ & \equiv u\end{aligned}$$

failure is cylinder @ origin with radius 0.3

Non-Robust BRT

$$\max_u \min_{\tau \in [t, T]} \ell(x(\tau))$$



Robust BRT

$$\max_{\pi_u} \min_{\pi_d} \min_{\tau \in [t, T]} \ell(x(\tau))$$

