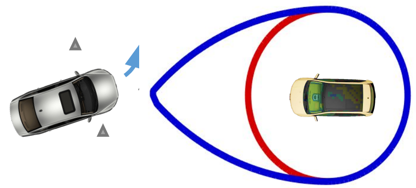
HJ Reachability Analysis II

CMPT 882

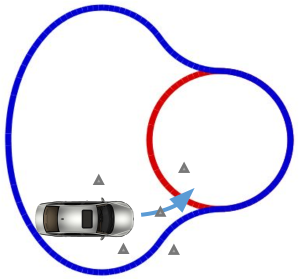
Mar. 4

Terminology

- Minimal backward reachable set
  - $\mathcal{A}(t) = \{\bar{x}: \exists\Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control minimizes size of reachable set

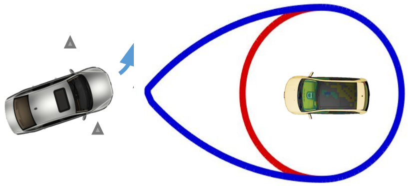


- Maximal backward reachable set
  - $\mathcal{R}(t) = \{\bar{x}: \forall\Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control maximizes size of reachable set



Reaching vs. Avoiding

- Avoiding danger



  - BRS definition

$$\mathcal{A}(t) = \{\bar{x}: \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$$

  - Value function

$$V(t,x) = \min_{\Gamma[u]} \max_{u} l\big(x(0)\big)$$

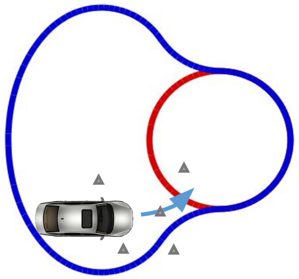
  - HJ PDE

$$\frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right] = 0$$

  - Optimal control

$$u^* = \arg\max_{u} \min_{d} \left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d)$$

- Reaching a goal



  - BRS definition

$$\mathcal{R}(t) = \{\bar{x}: \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$$

  - Value function

$$V(t,x) = \max_{\Gamma[u]} \min_{u} l\big(x(0)\big)$$

  - HJ PDE

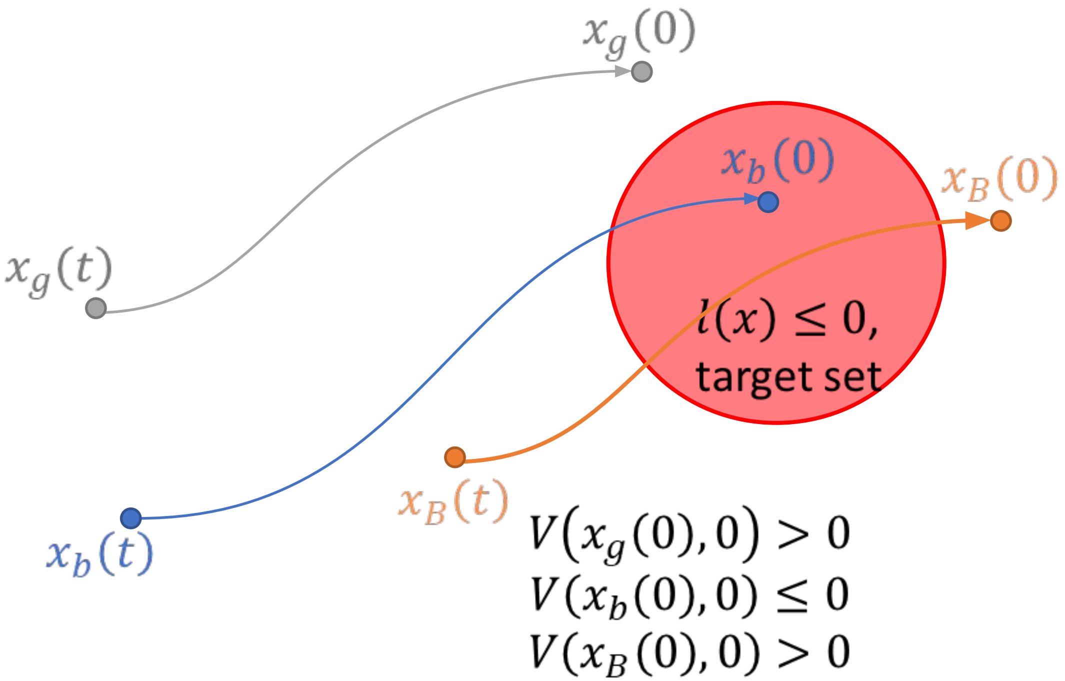
$$\frac{\partial V}{\partial t} + \min_{u} \max_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right] = 0$$

  - Optimal control

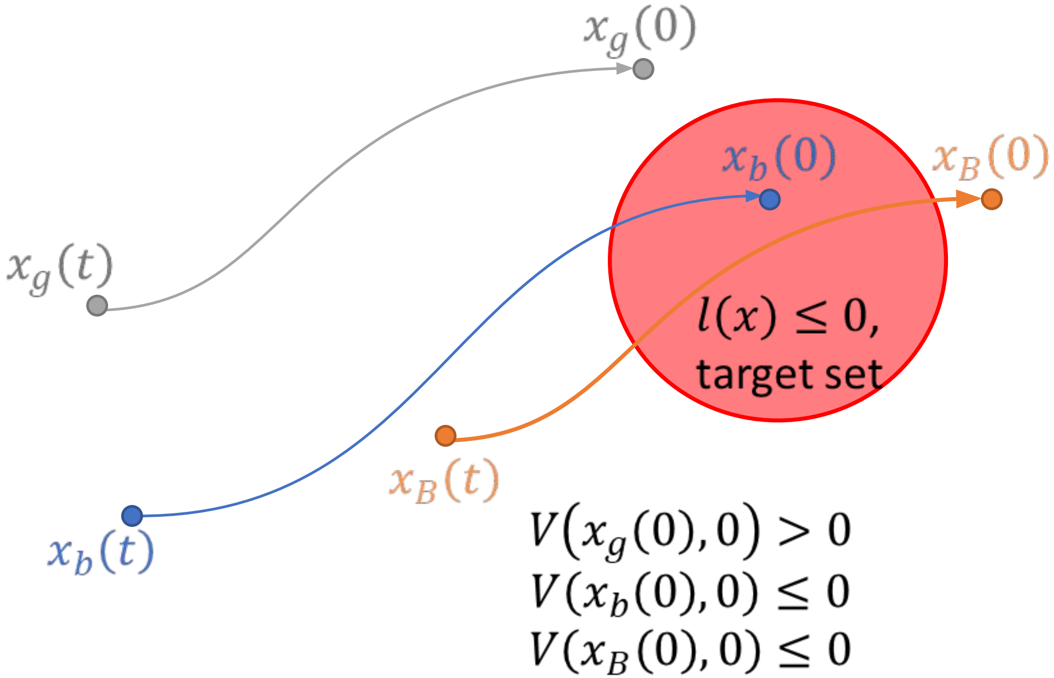
$$u^* = \arg\min_{u} \max_{d} \left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d)$$

"Sets" vs. "Tubes"

- Backward reachable set (BRS)
  - Only final time matters
  - Initial states that passing through target are not necessarily in BRS
  - Not ideal for safety

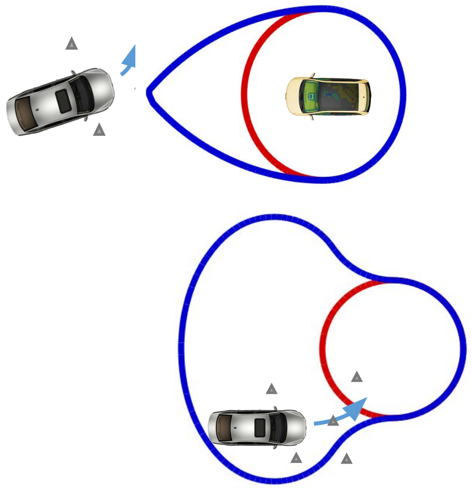


- Backward reachable tube (BRT)
  - Keep track of entire time duration
  - Initial states that pass thorugh target are in BRT
  - Used to make safety guarantees



Terminology

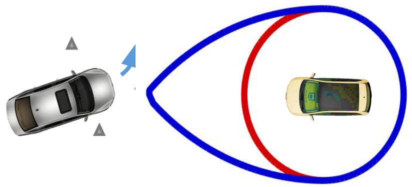
-



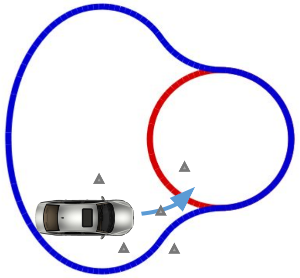
Terminology

- Minimal backward reachable set
  - $\mathcal{A}(t) = \{\bar{x}: \exists\Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control minimizes size of reachable set

- Maximal backward reachable set
  - $\mathcal{R}(t) = \{\bar{x}: \forall\Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, x(0) \in \mathcal{T}\}$
  - Control maximizes size of reachable set

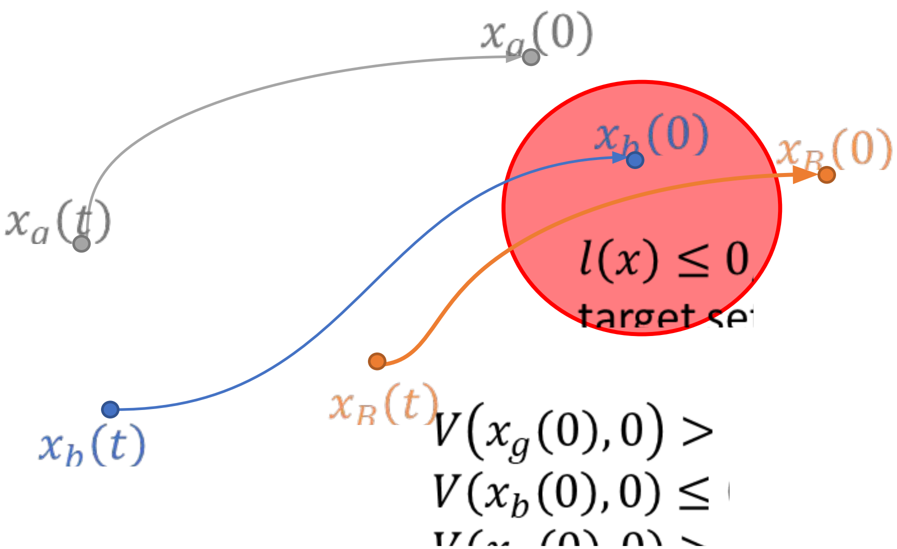


- Minimal and maximal backward reachable tube
  - $\bar{\mathcal{A}}(t) = \{\bar{x}: \exists\Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, \exists \boldsymbol{s} \in [\boldsymbol{t}, \boldsymbol{0}], x(s) \in \mathcal{T}\}$
  - $\bar{\mathcal{R}}(t) = \{\bar{x}: \forall\Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, \exists \boldsymbol{s} \in [\boldsymbol{t}, \boldsymbol{0}], x(s) \in \mathcal{T}\}$

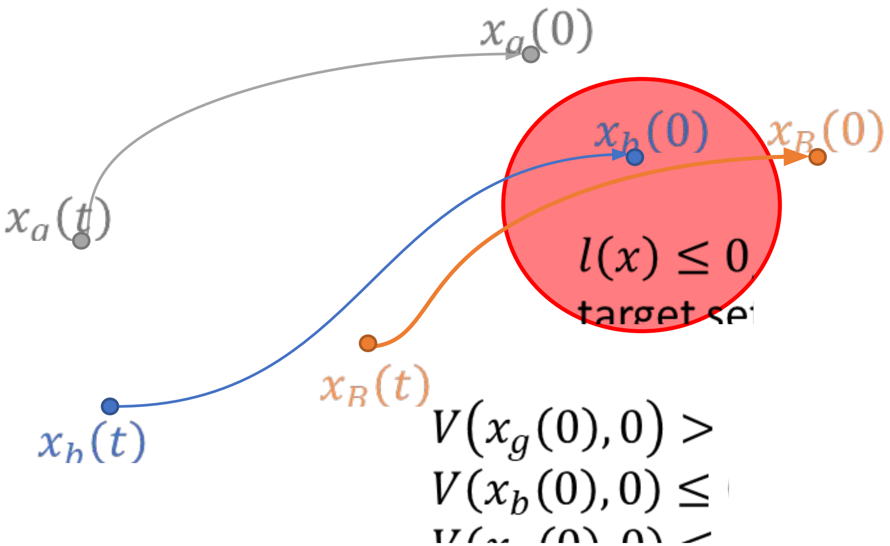


"Sets" vs. "Tubes"

- Backward reachable set (BRS)



- Backward reachable tube (BRT)



- Value function definition
  - $V(t,x) = \min_{\Gamma[u]} \max_{u} l\big(x(0)\big)$
- Value function obtained from

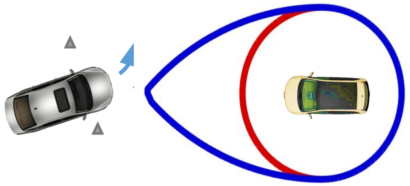
$$\frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right] = 0$$

- Value function definition
  - $V(t,x) = \min_{\Gamma[u]} \max_{u} \min_{s\in[t,0]} l\big(x(s)\big)$
- Value function obtained from

$$\min\left\{ \frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right], l(x) - V(t,x) \right\} = 0$$

Reaching vs. Avoiding: Backward Reachable Tubes

- Avoiding danger



  - BRT definition

$$\bar{\mathcal{A}}(t) = \left\{ \begin{matrix} \bar{x} \colon \exists \Gamma[u](\cdot), \forall u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, \\ \exists \boldsymbol{s} \in [\boldsymbol{t}, \boldsymbol{0}], x(s) \in \mathcal{T} \end{matrix} \right\}$$

  - Value function

$$V(t,x) = \min_{\Gamma[u]} \max_{u} \min_{\boldsymbol{s} \in [\boldsymbol{t},\boldsymbol{0}]} \boldsymbol{l}\big(\boldsymbol{x}(\boldsymbol{s})\big)$$

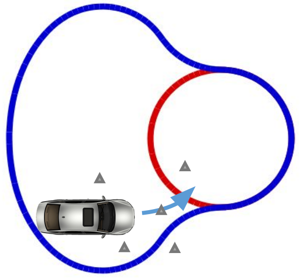
  - **HJ Variational Inequality**

$$\min\left\{ \frac{\partial V}{\partial t} + \max_{u} \min_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right], l(x) - V(t,x) \right\} = 0$$

  - Optimal control

$$u^* = \arg\max_{u} \min_{d} \left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d)$$

- Reaching a goal



  - BRT definition

$$\bar{\mathcal{R}}(t) = \left\{ \begin{matrix} \bar{x} \colon \forall \Gamma[u](\cdot), \exists u(\cdot), \dot{x} = f(x,u,d), x(t) = \bar{x}, \\ \exists s \in [t, 0], x(s) \in \mathcal{T} \end{matrix} \right\}$$

  - Value function

$$V(t,x) = \max_{\Gamma[u]} \min_{u} \min_{\boldsymbol{s} \in [\boldsymbol{t},\boldsymbol{0}]} \boldsymbol{l}\big(\boldsymbol{x}(\boldsymbol{s})\big)$$

  - **HJ Variational Inequality**

$$\min\left\{ \frac{\partial V}{\partial t} + \min_{u} \max_{d} \left[\left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d) \right], l(x) - V(t,x) \right\} = 0$$

  - Optimal control

$$u^* = \arg\min_{u} \max_{d} \left(\frac{\partial V}{\partial x} \right)^{\top} f(x,u,d)$$

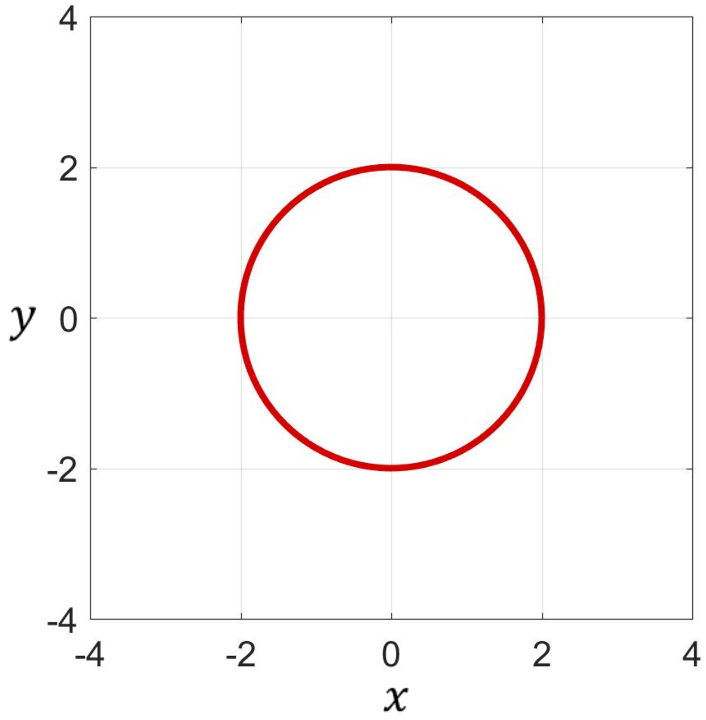
Hamilton-Jacobi (HJ) Reachability Theory

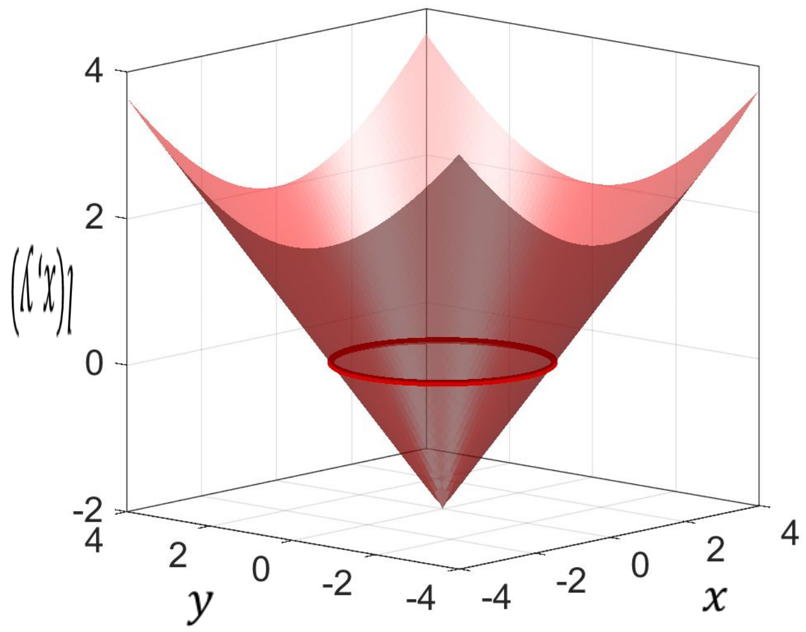
- Target set: $\mathcal{T} = \{x: l(x) < 0\}$

- Value function: $V(t, x)$

$$V\big(t, x(t)\big) = \min_{\Gamma[u](\cdot)} \max_{u(\cdot)} \min_{s \in [t,0]} l\big(x(s)\big)$$

subject to $\dot{x} = f(x, u, d), t \leq 0$





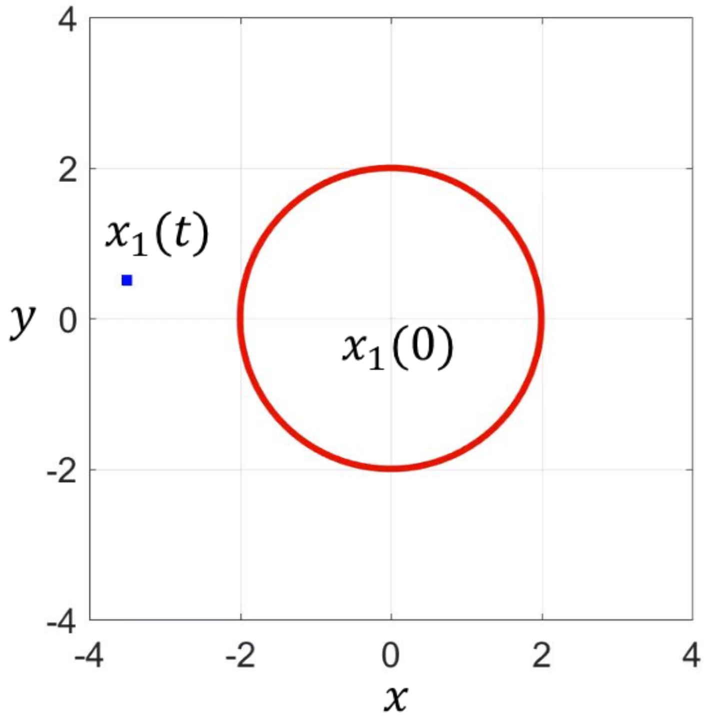
Hamilton-Jacobi (HJ) Reachability Theory

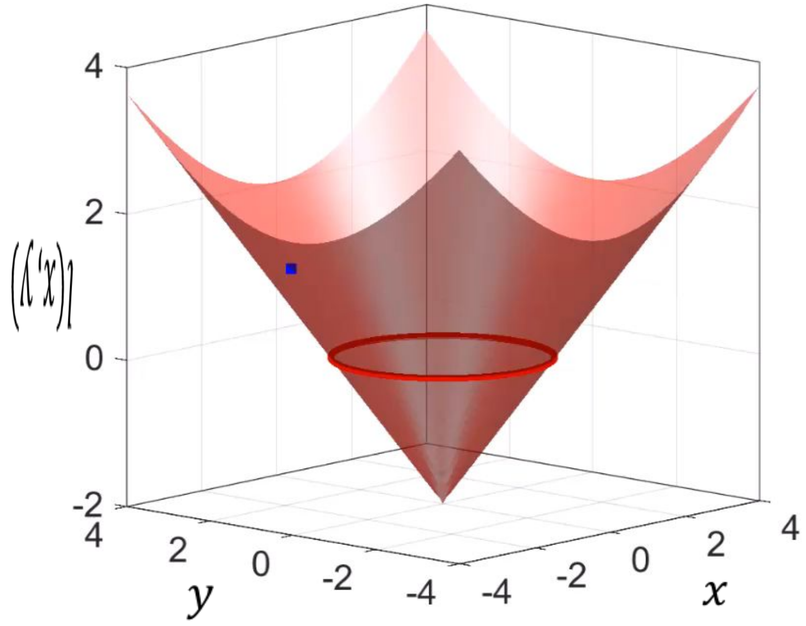
- Target set: $\mathcal{T} = \{x : l(x) < 0\}$

- Value function: $V(t, x)$

$$V\big(t, x(t)\big) = \min_{\Gamma[u](\cdot)} \max_{u(\cdot)} \min_{s \in [t,0]} l\big(x(s)\big)$$

subject to $\dot{x} = f(x, u, d), t \leq 0$





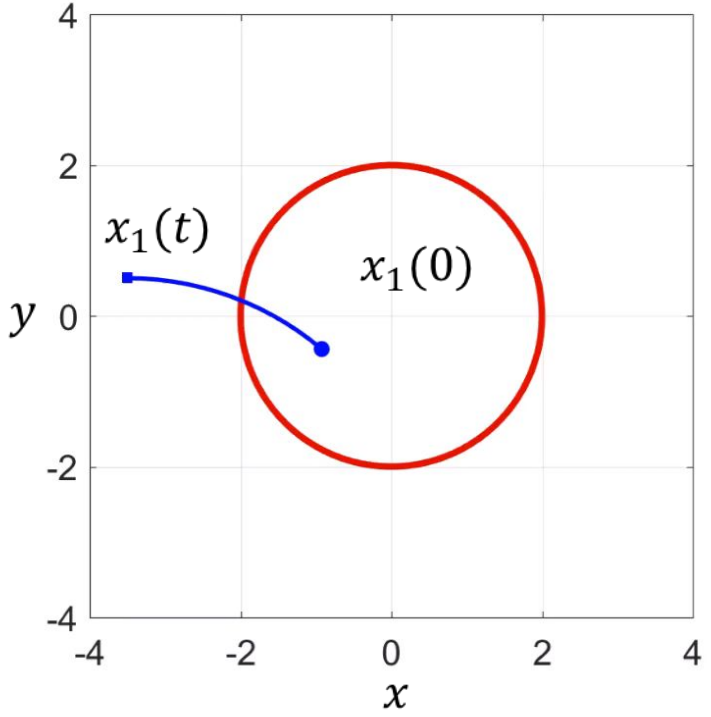
Hamilton-Jacobi (HJ) Reachability Theory

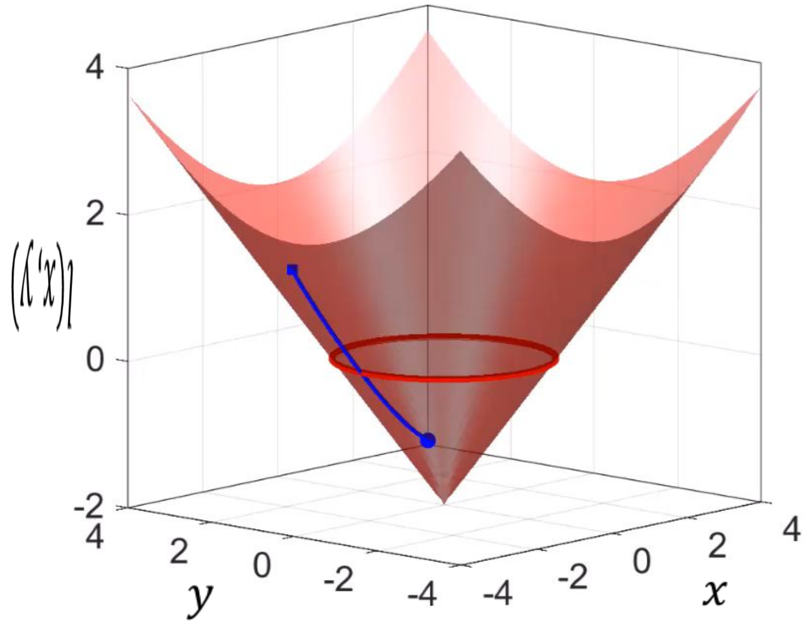
- Target set: $\mathcal{T} = \{x: l(x) < 0\}$

- Value function: $V(t, x)$

$$V\big(t, x(t)\big) = \min_{\Gamma[u](\cdot)} \max_{u(\cdot)} \min_{s \in [t,0]} l\big(x(s)\big)$$

subject to $\dot{x} = f(x, u, d), t \leq 0$





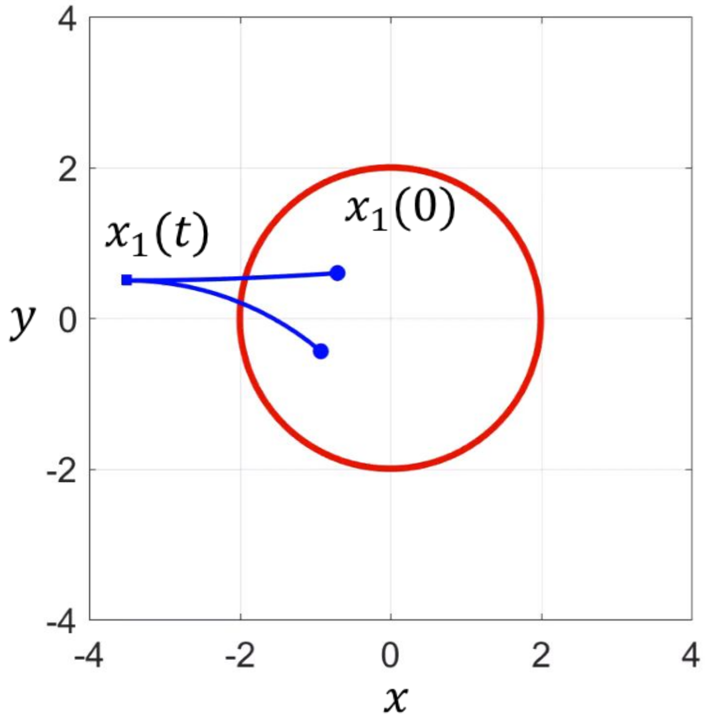
Hamilton-Jacobi (HJ) Reachability Theory

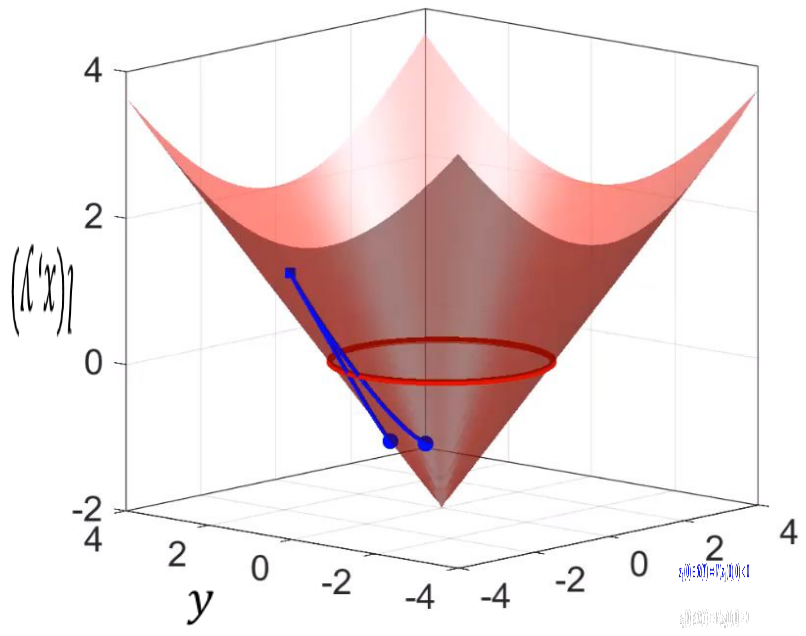
- Target set: $\mathcal{T} = \{x: l(x) < 0\}$

- Value function: $V(t, x)$

$$V\big(t, x(t)\big) = \min_{\Gamma[u](\cdot)} \max_{u(\cdot)} \min_{s\in[t,0]} l\big(x(s)\big)$$

subject to $\dot{x} = f(x, u, d), t \leq 0$





$\boldsymbol{x_1(t) \in \mathcal{R}(t) \Leftrightarrow V\big(t, x_1(t)\big) < 0}$

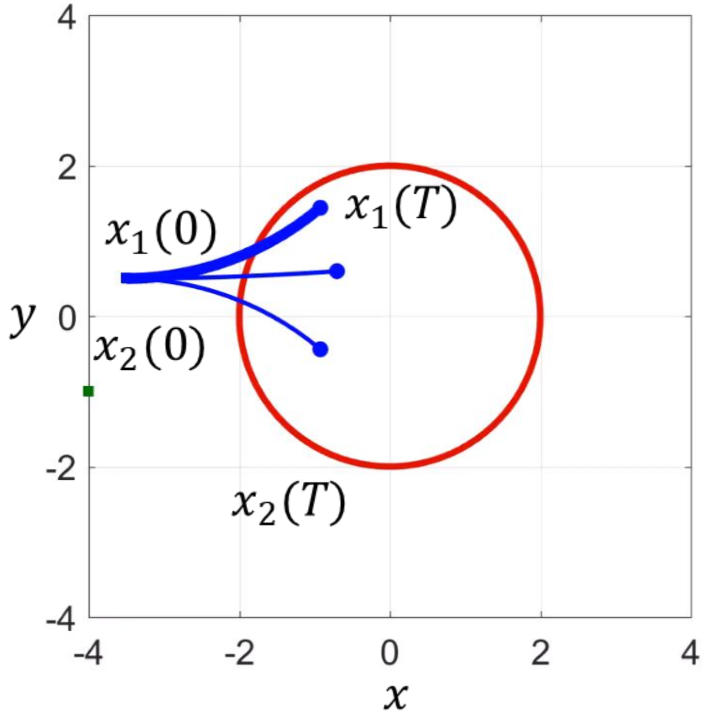
$z_2(0) \notin \mathcal{R}(T) \Leftrightarrow V(z_2(0), 0) \geq 0$

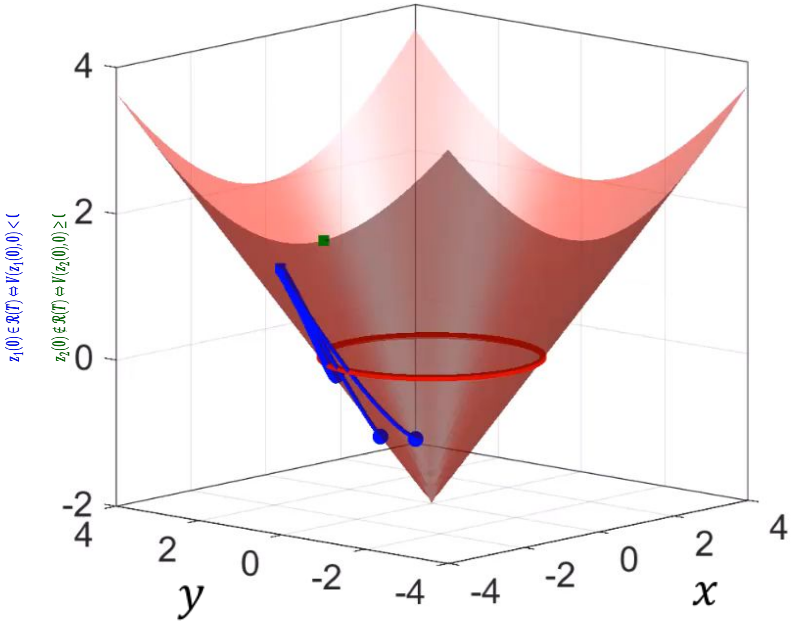
Hamilton-Jacobi (HJ) Reachability Theory

- Target set: $\mathcal{T} = \{x : l(x) < 0\}$
- Value function: $V(t,x)$

$$V\big(t, x(t)\big) = \min_{\Gamma[u](\cdot)} \max_{u(\cdot)} \min_{s\in[t,0]} l\big(x(s)\big)$$

subject to $\dot{x} = f(x,u,d), t \leq 0$





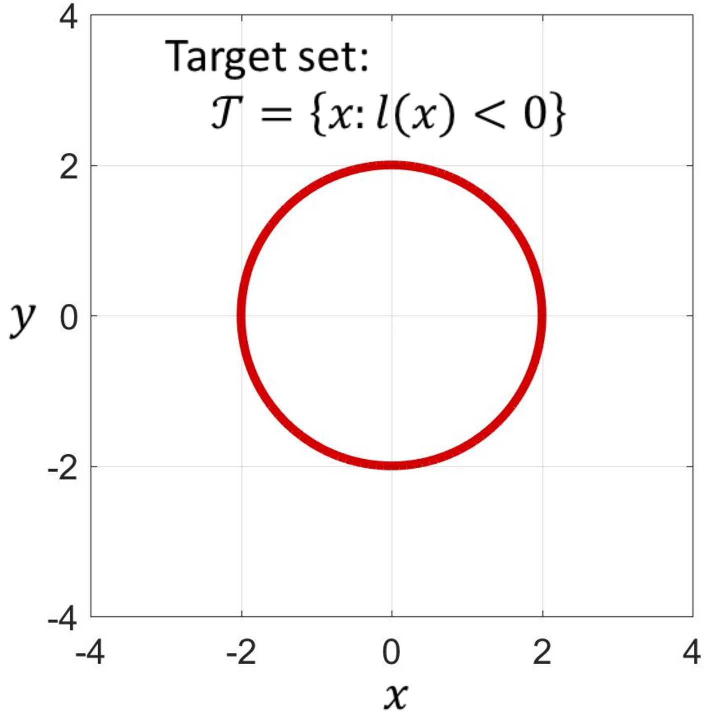
$\boldsymbol{x_1(t) \in \mathcal{R}(t) \Leftrightarrow V\big(t, x_1(t)\big) < 0}$

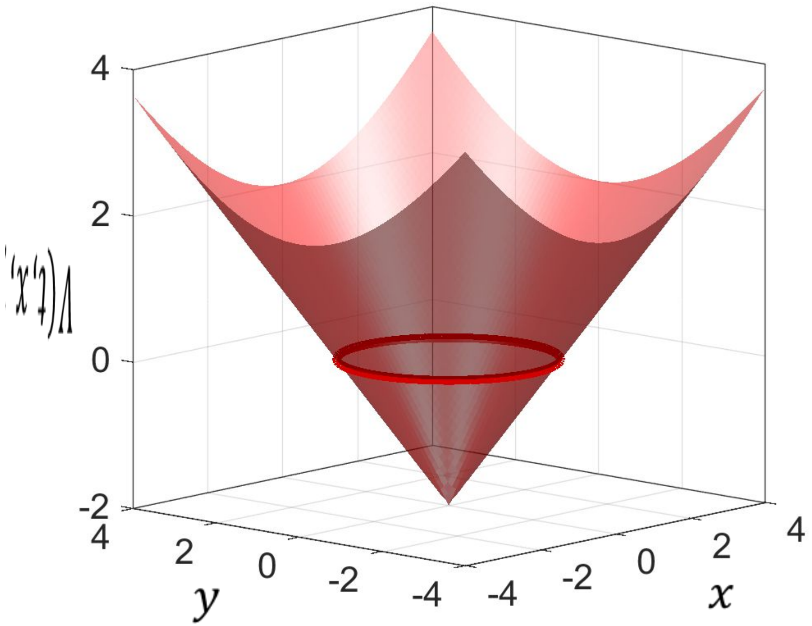
$\boldsymbol{x_2(t) \notin \mathcal{R}(t) \Leftrightarrow V\big(t, x_2(t)\big) \geq 0}$

Hamilton-Jacobi (HJ) Reachability Theory

- Hamilton-Jacobi Variational Inequality:

$$\frac{dV}{dt} + \min\left\{0, \max_u \min_d \left(\frac{\partial V}{\partial x}\right)^{\top} f(x,u,d)\right\} = 0, t \leq 0$$





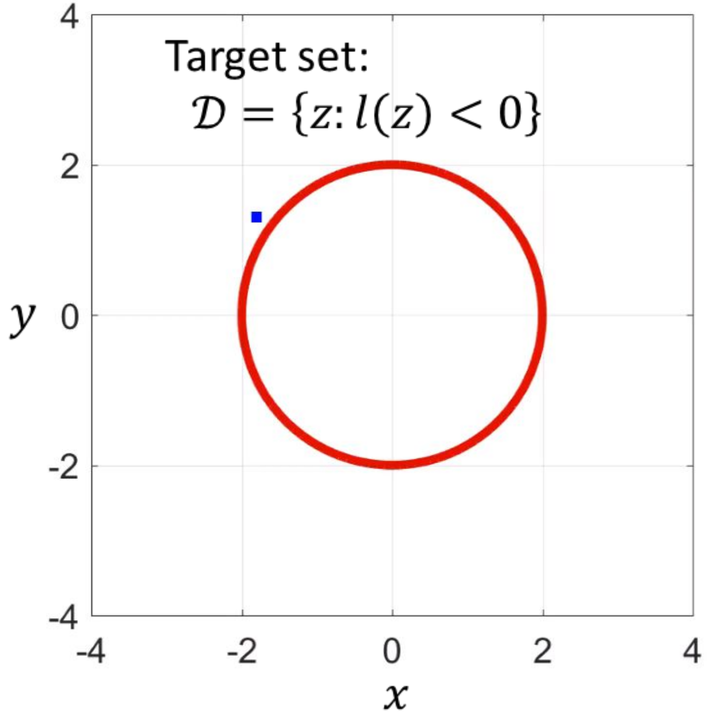
Reachable set:
$\mathcal{R}(t) = \{x: V(t,x) < 0\}$

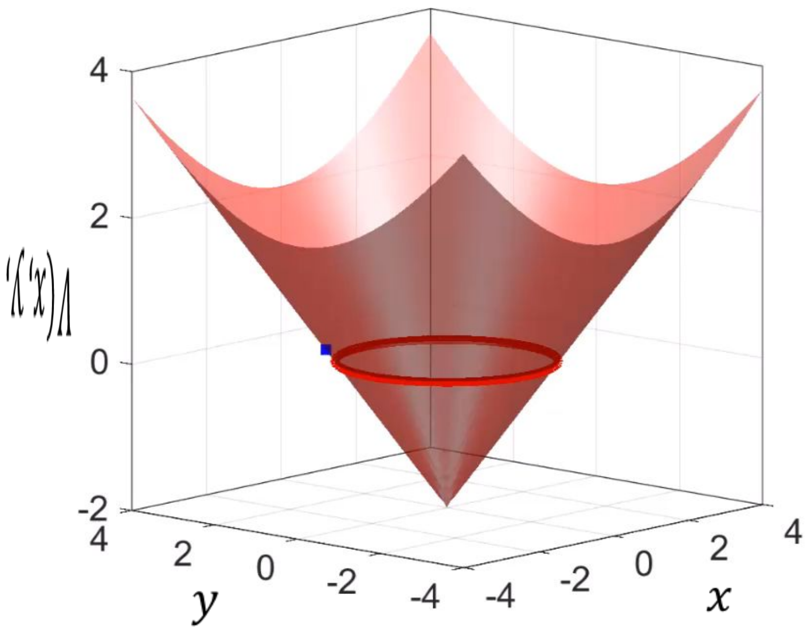
Hamilton-Jacobi (HJ) Reachability Theory

- **Hamilton-Jacobi PDE**: based on the dynamic programming principle

$$\frac{dV}{dt} + \min\left\{0, \max_{u} \min_{d} \nabla V \cdot f(z,u,d) \right\} = 0, t \in [0,T]$$

$$V(z,T) = l(z)$$





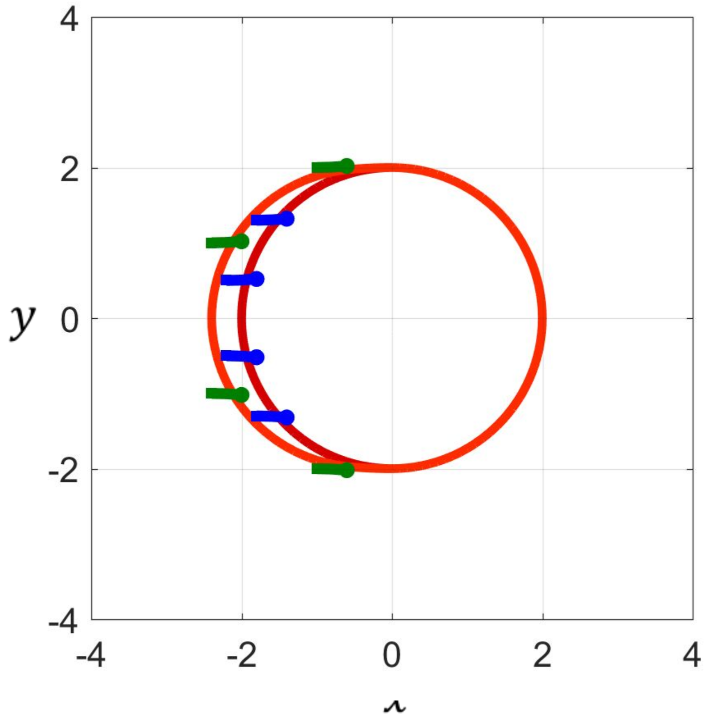
Reachable set:
$\mathcal{R}(t) = \{z\colon V(z,t) < 0\}$

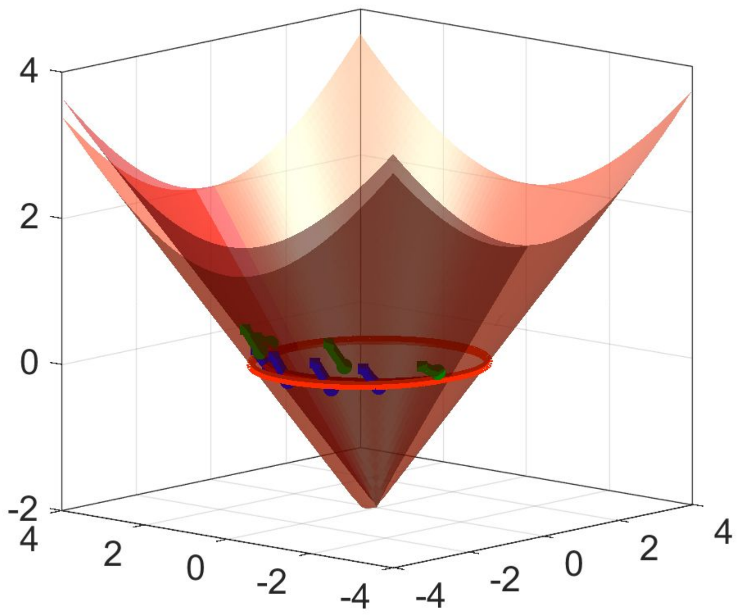
Hamilton-Jacobi (HJ) Reachability Theory

- **Hamilton-Jacobi PDE**: based on the dynamic programming principle

$$\frac{dV}{dt} + \min\left\{0, \max_u \min_d \nabla V \cdot f(z,u,d)\right\} = 0, t \in [0,T]$$

$$V(z,T) = l(z)$$





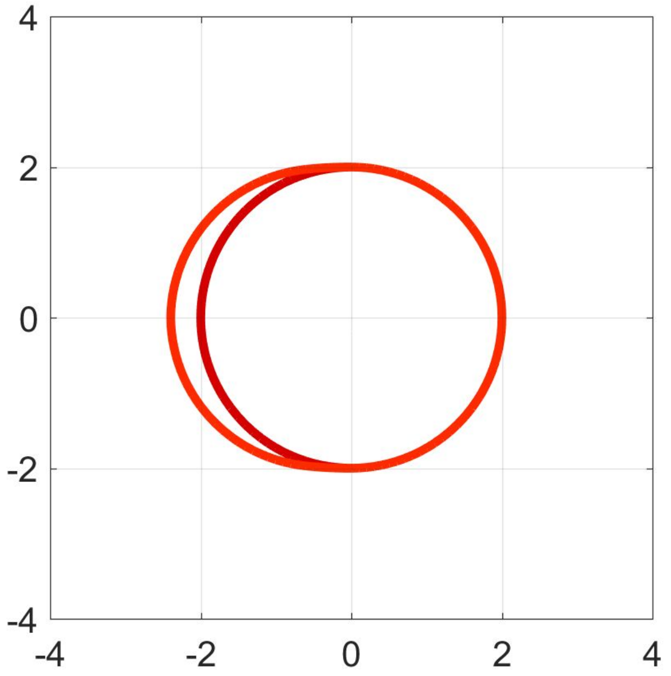
Reachable set:
$\mathcal{R}(t) = \{z: V(z,t) < 0\}$

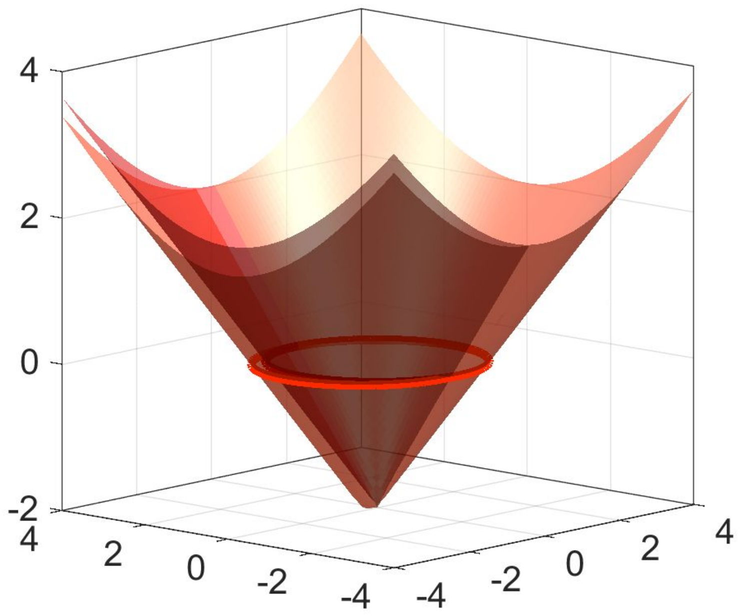
Hamilton-Jacobi (HJ) Reachability Theory

- **Hamilton-Jacobi PDE**: based on the dynamic programming principle

$$\frac{dV}{dt} + \min\left\{0, \max_u \min_d \nabla V \cdot f(z, u, d)\right\} = 0, t \in [0, T]$$

$$V(z, T) = l(z)$$





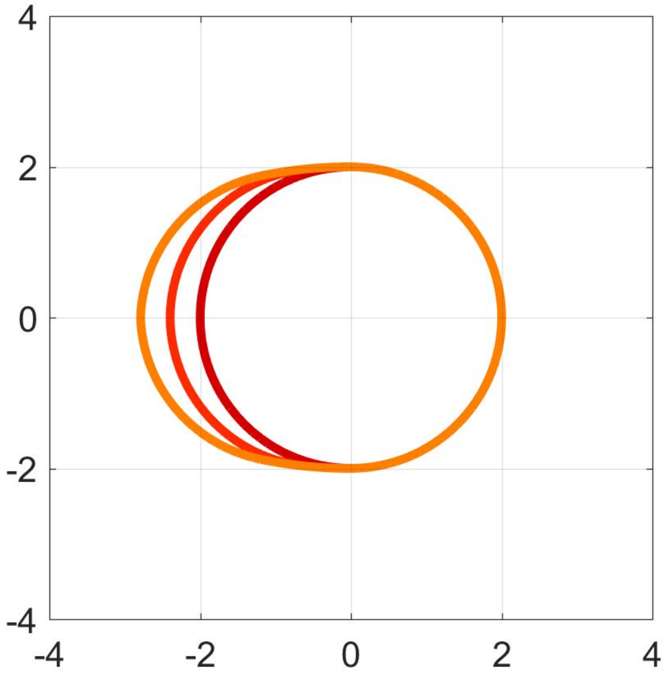
Reachable set:
$\mathcal{R}(t) = \{z: V(z, t) < 0\}$

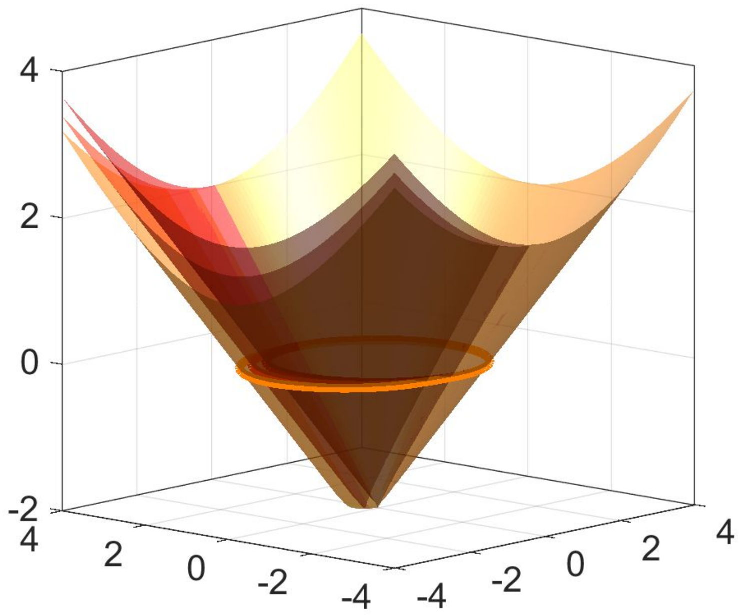
Hamilton-Jacobi (HJ) Reachability Theory

- **Hamilton-Jacobi PDE**: based on the dynamic programming principle

$$\frac{dV}{dt} + \min\left\{0, \max_u \min_d \nabla V \cdot f(z,u,d)\right\} = 0, t \in [0,T]$$

$$V(z,T) = l(z)$$





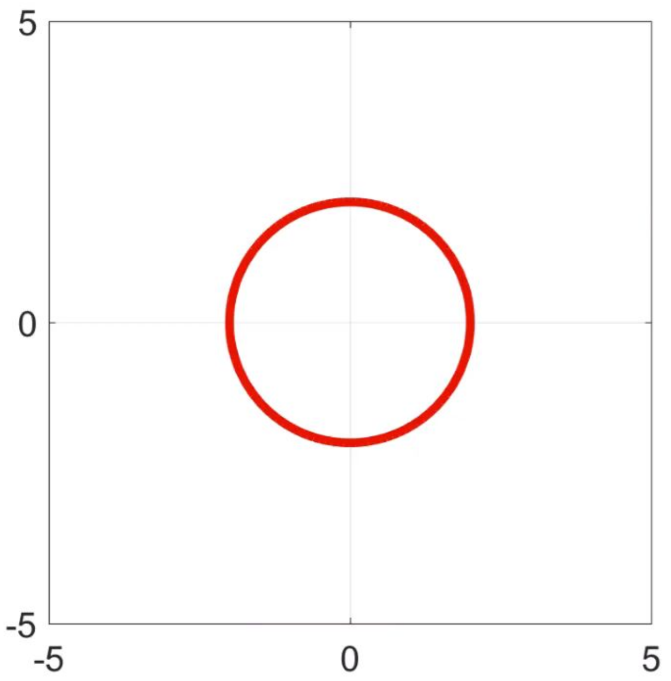
Reachable set:
$\mathcal{R}(t) = \{z : V(z,t) < 0\}$

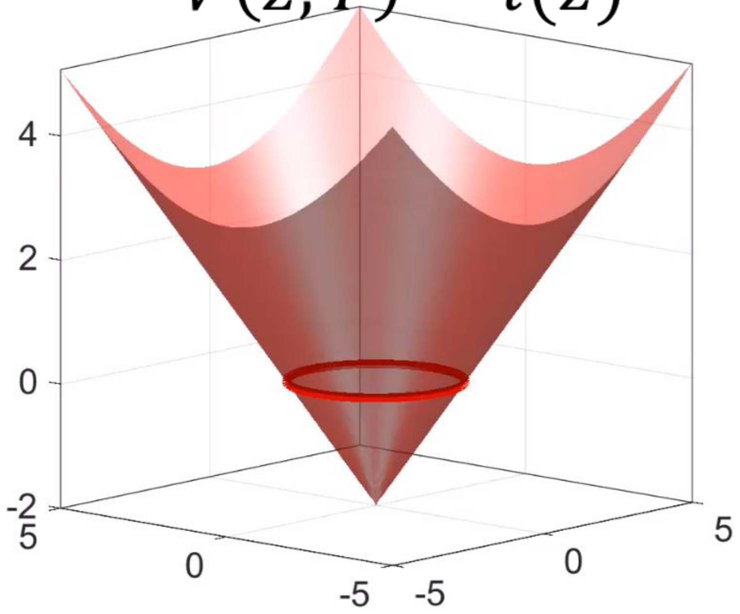
Hamilton-Jacobi (HJ) Reachability Theory

- **Hamilton-Jacobi PDE**: based on the dynamic programming principle

$$\min\left\{\frac{\partial V}{\partial t} + \max_u \min_d \left[\left(\frac{\partial V}{\partial x}\right)^{\mathsf{T}} f(x,u,d)\right], l(x) - V(t,x)\right\} = 0$$

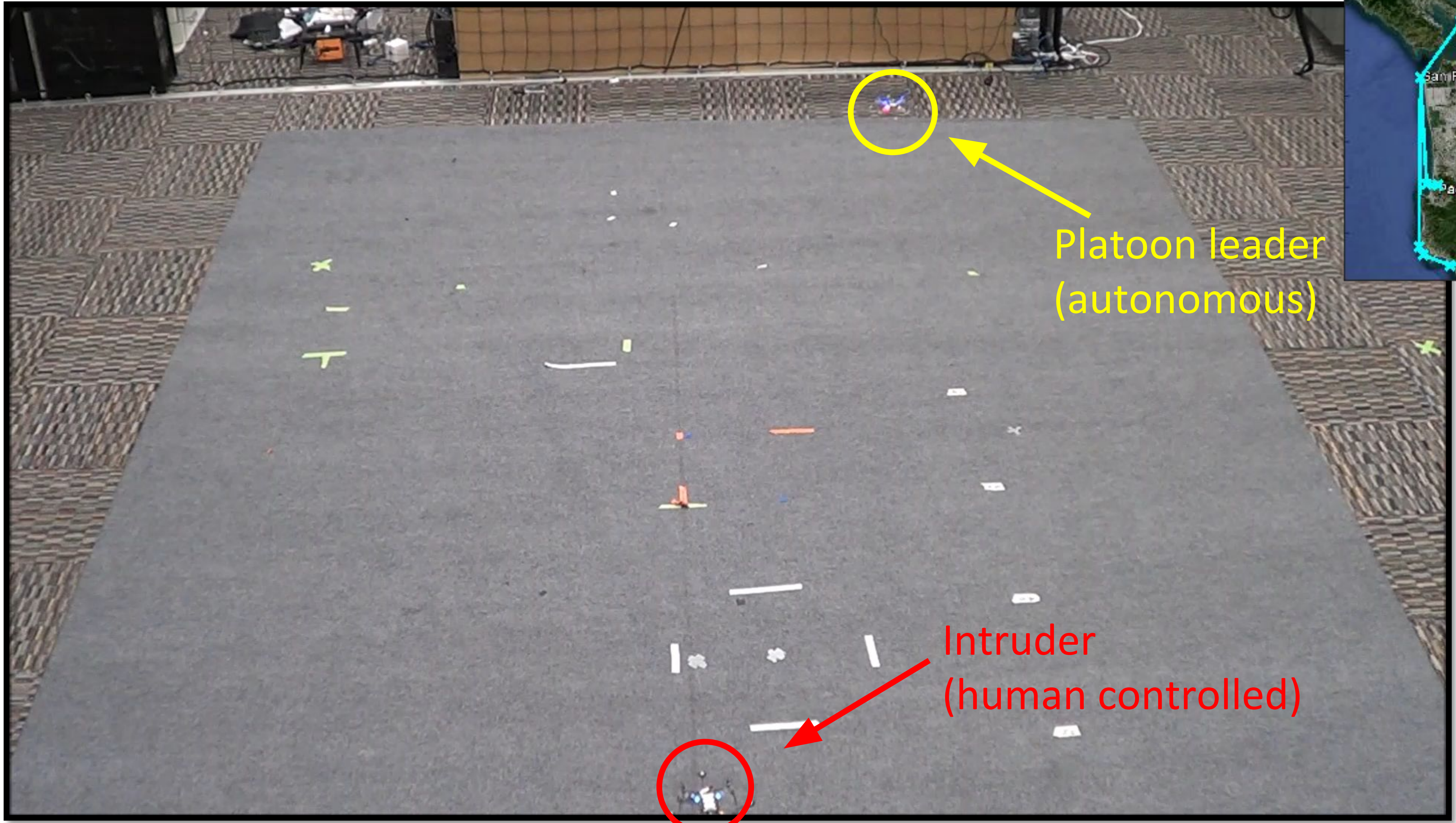
$$V(z,T) = l(z)$$

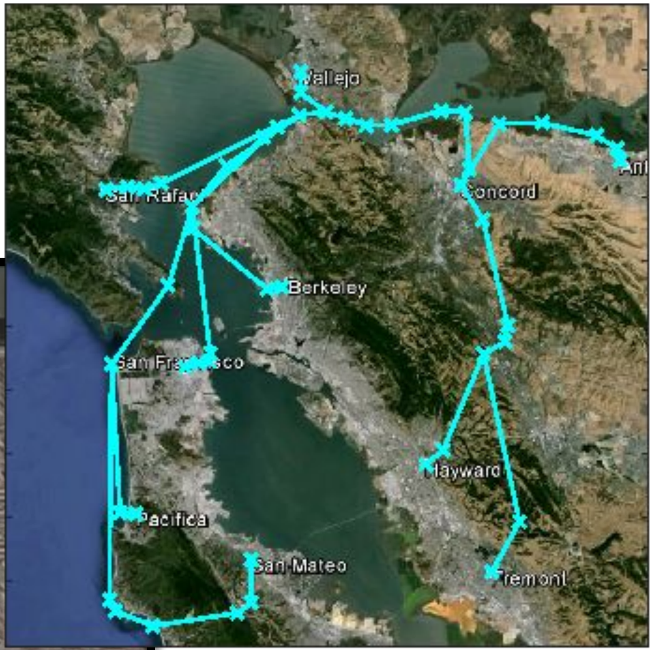




Reachable set:
$\mathcal{R}(t) = \{z: V(z,t) < 0\}$

Intruder Avoidance



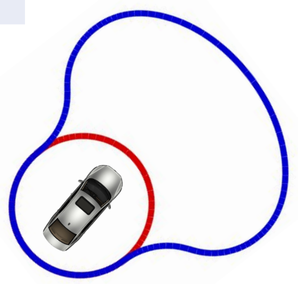


“Flavours” of Reachability

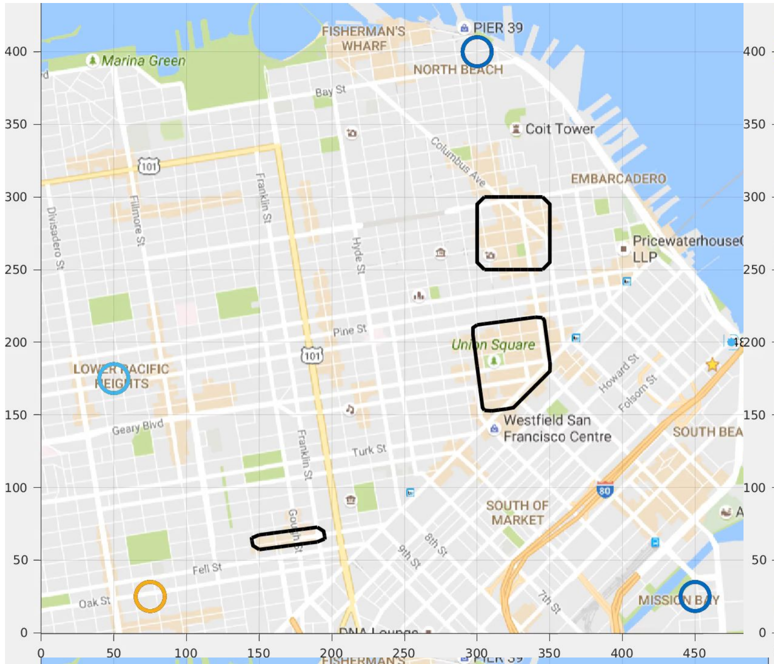
- So far:

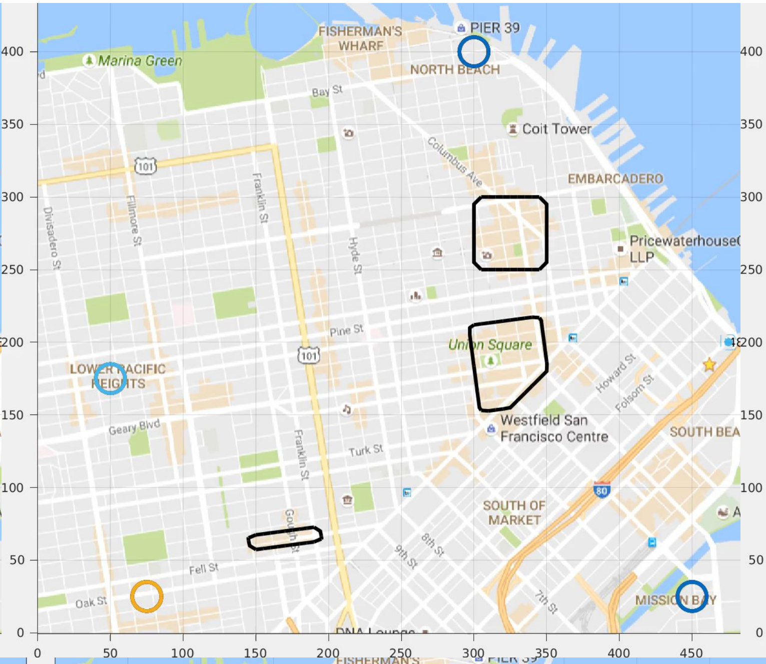
	Backward reachable set	Backward reachable tube
Minimal		
Maximal		

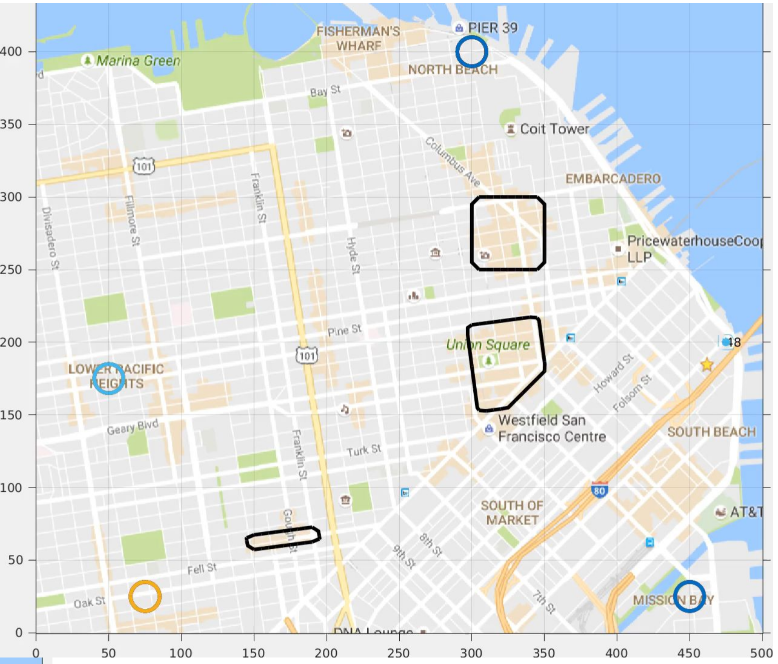
- Other variations:
  - Forward reachable sets and tubes
  - Reach-avoid sets and tubes
    - States from which goal can be reached while avoiding obstacles



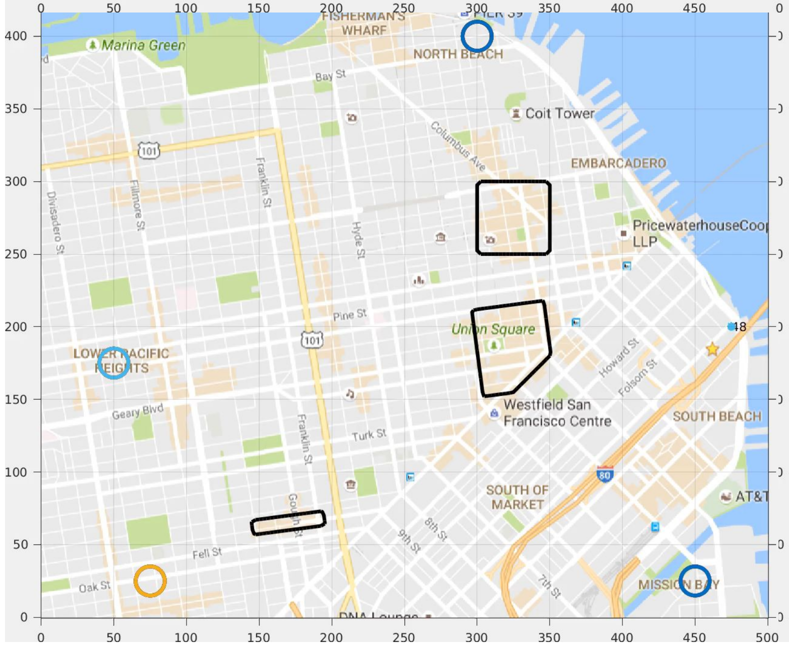
Wind speed: 11 m/s

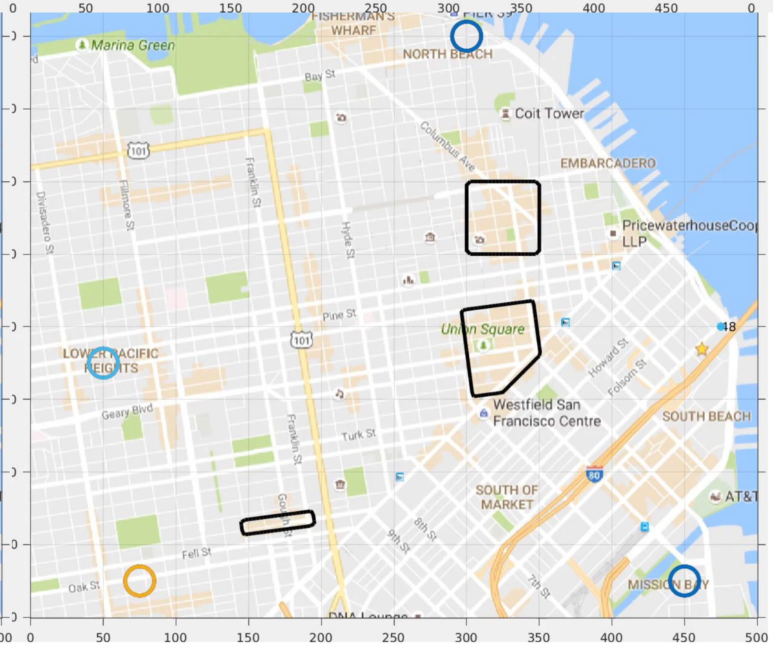






6 m/s





Unmanned Aerial Systems Traffic Management

Scheduled arrival time separation: 0 s

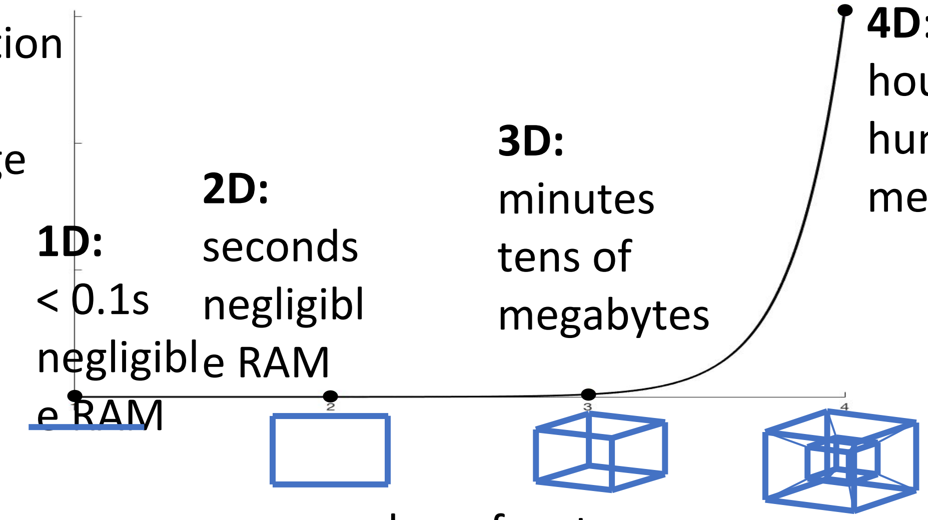
5 s

10 s

Comments

- Computational complexity
  - $V(t, x)$ is computed on an $(n + 1)$-dimensional grid
  - Currently, $n \leq 5$. GPU acceleration under-way
  - Dimensionality reduction methods sometimes help

$O(N^d)$ time and space



- Alternative approaches
  - Sacrifice global optimality
  - Give up guarantees
  - Sampling-based methods
  - Reinforcement learning

Numerical Toolboxes

- helperOC Matlab toolbox
  - https://github.com/HJReachability/helperOC.git
  - Reachability wrapper around the level set toolbox
  - Requires level set toolbox
    - Hamilton-Jacobi PDE solver by Ian Mitchell, UBC
    - https://bitbucket.org/ian_mitchell/toolboxls

- C++ and CUDA version in development, beta also available
  - C++: 5+ times faster than Matlab
  - CUDA: Up to 100 times faster than Matlab
  - https://github.com/HJReachability/beacls

Tutorial Code Overview

```
function tutorial()
% 1. Run Backward Reachable Set (BRS) with a goal
%     uMode = 'min' <-- goal
%     minWith = 'none' <-- Set (not tube)
%     compTraj = false <-- no trajectory
% 2. Run BRS with goal, then optimal trajectory
%     uMode = 'min' <-- goal
%     minWith = 'none' <-- Set (not tube)
%     compTraj = true <-- compute optimal trajectory
% 3. Run Backward Reachable Tube (BRT) with a goal, then optimal trajectory
%     uMode = 'min' <-- goal
%     minWith = 'minVWithTarget' <-- Tube (not set)
%     compTraj = true <-- compute optimal trajectory
% 4. Add disturbance
%     dStep1: define a dMax (dMax = [.25, .25, 0];)
%     dStep2: define a dMode (opposite of uMode)
%     dStep3: input dMax when creating your DubinsCar
%     dStep4: add dMode to schemeData
% 5. Change to an avoid BRT rather than a goal BRT
%     uMode = 'max' <-- avoid
%     dMode = 'min' <-- opposite of uMode
%     minWith = 'minVWithTarget' <-- Tube (not set)
%     compTraj = false <-- no trajectory
% 6. Change to a Forward Reachable Tube (FRT)
%     add schemeData.tMode = 'forward'
%     note: now having uMode = 'max' essentially says "see how far I can
%     reach"
% 7. Add obstacles
%     add the following code:
%     obstacles = shapeCylinder(g, 3, [-1.5; 1.5; 0], 0.75);
%     HJIextraArgs.obstacles = obstacles;
% 8. Add random disturbance (white noise)
%     add the following code:
%     HJIextraArgs.addGaussianNoiseStandardDeviation = [0; 0; 0.5];
```

Tutorial Code Overview

Computation domain
- Make sure domain is large enough
- Make sure grid resolution captures smallest features
- Remember periodic state space dimensions (angles)

Target set
- Built-in functions available to create simple shapes
- Arbitrary functions can be defined using the grid
  - g.xs{i} in this context represents ith state

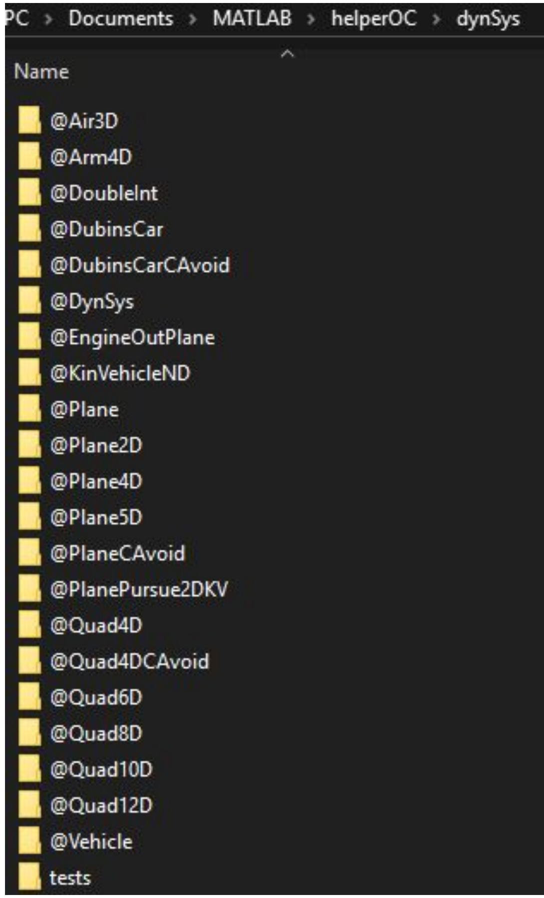
Time horizon
- dt and tau determine what t is stored for $V(t, x)$
- Time discretization for computation is determined automatically to ensure numerical stability

Vehicle parameters (Dubins car's speed and max turn rate)

Reach or avoid? (min for reach, max for avoid)

```
%% Should we compute the trajectory?
compTraj = false;

%% Grid
grid_min = [-5; -5; -pi]; % Lower corner of computation domain
grid_max = [5; 5; pi];    % Upper corner of computation domain
N = [41; 41; 41];         % Number of grid points per dimension
pdDims = 3;               % 3rd dimension is periodic
g = createGrid(grid_min, grid_max, N, pdDims);
% Use "g = createGrid(grid_min, grid_max, N);" if there are no periodic
% state space dimensions

%% target set
R = 1;
% data0 = shapeCylinder(grid,ignoreDims,center,radius)
data0 = shapeCylinder(g, 3, [0; 0; 0], R);
% also try shapeRectangleByCorners, shapeSphere, etc.

%% time vector
t0 = 0;
tMax = 2;
dt = 0.05;
tau = t0:dt:tMax;

%% problem parameters

% input bounds
speed = 1;
wMax = 1;
% do dStep1 here

% control trying to min or max value function?
uMode = 'min';
% do dStep2 here
```

Tutorial Code Overview



ODE model of system
- Implemented as classes, found in the DynSys folder
- Implementing extra models is relatively simple

Pack parameters and solve PDE
- minWith parameter determines whether reachable sets or tubes are computed
- Solution is stored in data, which is a $(n + 1)$-dimensional array

```
%% Pack problem parameters

% Define dynamic system
% obj = DubinsCar(x, wMax, speed, dMax)
dCar = DubinsCar([0, 0, 0], wMax, speed); %do dStep3 here

% Put grid and dynamic systems into schemeData
schemeData.grid = g;
schemeData.dynSys = dCar;
schemeData.accuracy = 'high'; %set accuracy
schemeData.uMode = uMode;
%do dStep4 here

%% additive random noise
%do Step8 here
%HJIextraArgs.addGaussianNoiseStandardDeviation = [0; 0; 0.5];
% Try other noise coefficients, like:
%    [0.2; 0; 0]; % Noise on X state
%    [0.2,0,0;0,0.2,0;0,0,0.5]; % Independent noise on all states
%    [0.2;0.2;0.5]; % Coupled noise on all states
%    {zeros(size(g.xs{1})); zeros(size(g.xs{1})); (g.xs{1}+g.xs{2})/20}; % State-dependent noise

%% If you have obstacles, compute them here

%% Compute value function

HJIextraArgs.visualize = true; %show plot
HJIextraArgs.fig_num = 1; %set figure number
HJIextraArgs.deleteLastPlot = true; %delete previous plot as you update

%[data, tau, extraOuts] = ...
% HJIPDE_solve(data0, tau, schemeData, minWith, extraArgs)
[data, tau2, ~] = ...
  HJIPDE_solve(data0, tau, schemeData, 'none', HJIextraArgs);
```