Intro to Reachability Code

Sylvia Herbert

The toolboxes

- toolboxLS (Ian Mitchell)
  - Used for solving PDEs using level set methods
  - Basic Reachability tools

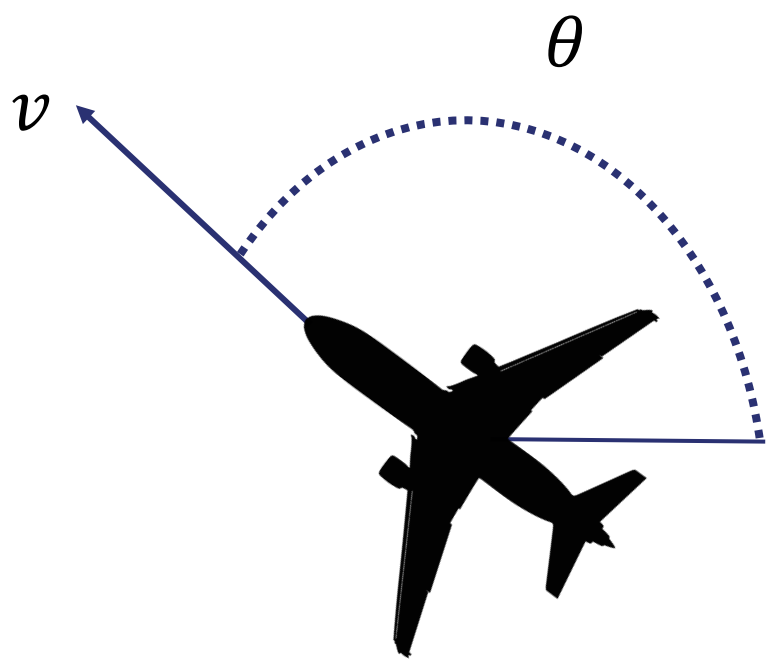
- helperOC (mostly Mo Chen)
  - More refined tools for using toolboxLS
  - Code from research in lab

Notes: We will use two toolboxes. The first is the level set toolbox that solves general PDEs using level set methods. The second is the optimal control repo that our lab has developed to do reachability stuff.

Download/Installation instructions

1. Follow the instructions for downloading toolboxLS here: http://www.cs.ubc.ca/~mitchell/ToolboxLS/
2. Download or clone the helperOC repository from here: https://github.com/HJReachability/helperOC
3. Put both the toolboxLS and helperOC folder/repo and subfolders into your matlab path
4. On matlab in the command line, type the following: tutorial_test()
5. If code doesn't run, you don't see a figure, or you get an error, email sylvia.herbert@berkeley.edu

Motivating Example: Dubins Car



$$\dot{x} = v\cos\theta + d_x$$
$$\dot{y} = v\sin\theta + d_y$$
$$\dot{\theta} = \omega + d_\theta$$

$$\omega \in [\omega_{min}, \omega_{max}]$$
$$\mathrm{d} \in [d_{min}, d_{max}]$$

Notes: For our tutorial example we will be using a standard dubins car with the dynamics shown

Finding the optimal control & disturbance

$$\dot{x} = v\cos\theta + d_1$$
$$\dot{y} = v\sin\theta + d_2$$
$$\dot{\theta} = \omega + d_3$$

$$\mathrm{H}^* = \min_{\omega}\max_{d}\{\nabla V(z(t),t)\cdot f(z,\omega,d,t)\}$$

$$\nabla \mathrm{V} = \left[\mathrm{p_x}, \mathrm{p_y}, \mathrm{p}_\theta\right];$$

$$\mathrm{H}^* = \min_{\omega}\max_{d}(\mathrm{p_x}\dot{x} + \mathrm{p_y}\dot{y} + \mathrm{p}_\theta\dot{\theta})$$

$$\mathrm{H}^* = \min_{\omega}\max_{d}\{\mathrm{p_x}(v\cos\theta + d_x) + \mathrm{p_y}\left(v\sin\theta + d_y\right) + \mathrm{p}_\theta(\omega + d_\theta)\}$$

$$\mathrm{H}^* = \min_{\omega}\max_{d}\left\{\mathrm{p_x}d_x + \mathrm{p_y}d_y + \mathrm{p}_\theta(\omega + d_\theta)\right\} + \text{extra terms}$$

$$\mathrm{H}^* = \min_{\omega}\{\mathrm{p}_\theta\omega\} + \max_{d}\{\mathrm{p_x}d_x + \mathrm{p_y}d_y + \mathrm{p}_\theta d_\theta\} + \text{extra terms}$$

Notes: To find the optimal control and disturbance we must find the argmin/argmax of the hamiltonian, as described in the intro to reachability theory tutorial

Finding the optimal control & disturbance

$$\mathrm{H}^* = \min_{\omega}\{\mathrm{p}_\theta \omega\} + \max_{d}\{\mathrm{p}_\mathrm{x} d_x + \mathrm{p}_\mathrm{y} d_y + \mathrm{p}_\theta d_\theta\} + \text{whatever}$$

$$\omega^* = \operatorname*{argmin}_{\omega}\{\mathrm{p}_\theta \omega\} \rightarrow \text{If } \mathrm{p}_\theta \geq 0,\ \omega^* = \omega_{min}, \text{ else } \omega^* = \omega_{max}$$

$$d_x^* = \operatorname*{argmax}_{d_x}\{\mathrm{p}_\mathrm{x} d_x\}$$
$$d_y^* = \operatorname*{argmax}_{d_y}\{\mathrm{p}_\mathrm{y} d_y\}$$
$$d_z^* = \operatorname*{argmax}_{d_z}\{\mathrm{p}_\mathrm{y} d_z\}$$

$$\rightarrow \text{If } \mathrm{p_i} \geq 0,\ \mathrm{d_i^*} = d_{max}, \text{ else } \mathrm{d_i^*} = d_{min}$$

Notes: Here we show what the optimal control and disturbance will be, and how we will input them into the control. If you have a more complicated control space or non-affine control inputs, you could use a solver instead and plug that into the optimal control and disturbance files.

$$\omega \in [\omega_{min}, \omega_{max}]$$
$$\mathrm{d} \in [d_{min}, d_{max}]$$

Coding up the Dubins Car

- Dubins Car will be a subset of the dynSys class
  - helperOC → dynSys → DubinsCar
- DubinsCar.m defines the dubins car class and its related properties and methods/functions
- dynamics.m defines the dynamics. Note that we define it twice depending on how the function is called (with a cell of states or a single state)
- optCtrl.m defines the optimal control. We simply plug in the control we derived in the previous slide
- optDstb.m defines the optimal disturbance in the same way

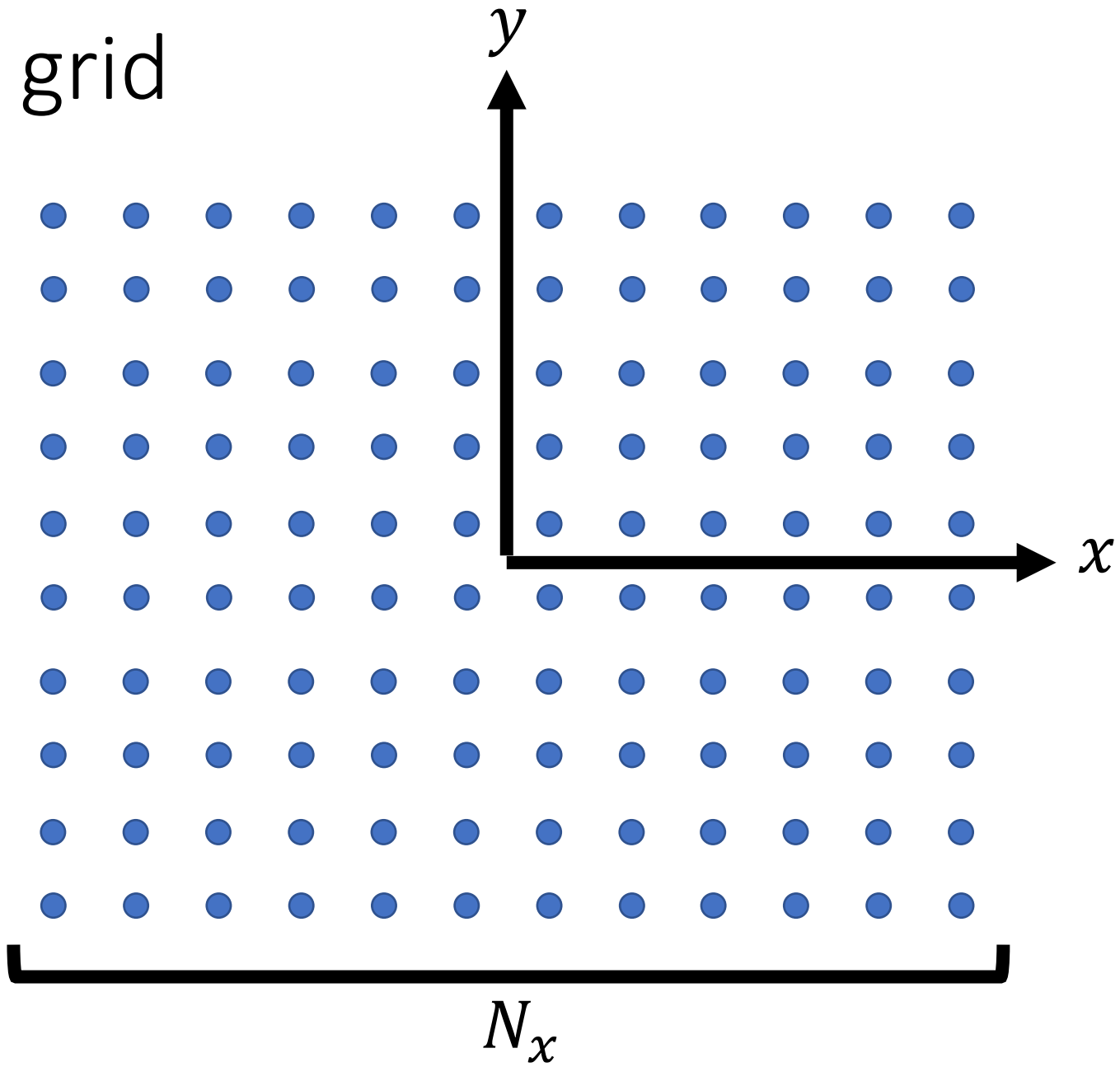
Notes: Here we define how to create your vehicle class with its dynamics and how to find its optimal control/disturbance.

tutorial_test.m

1. Run Backward Reachable Set (BRS) with a goal
2. Same thing but add in computing an optimal trajectory after
3. Run Backward Reachable Tube (BRT) with a goal, optimal trajectory
4. Add disturbance
5. Change to a BRT with an avoid set (instead of a goal), remove trajectory
6. Change to a Forward Reachable Tube (FRT)
7. Add obstacle

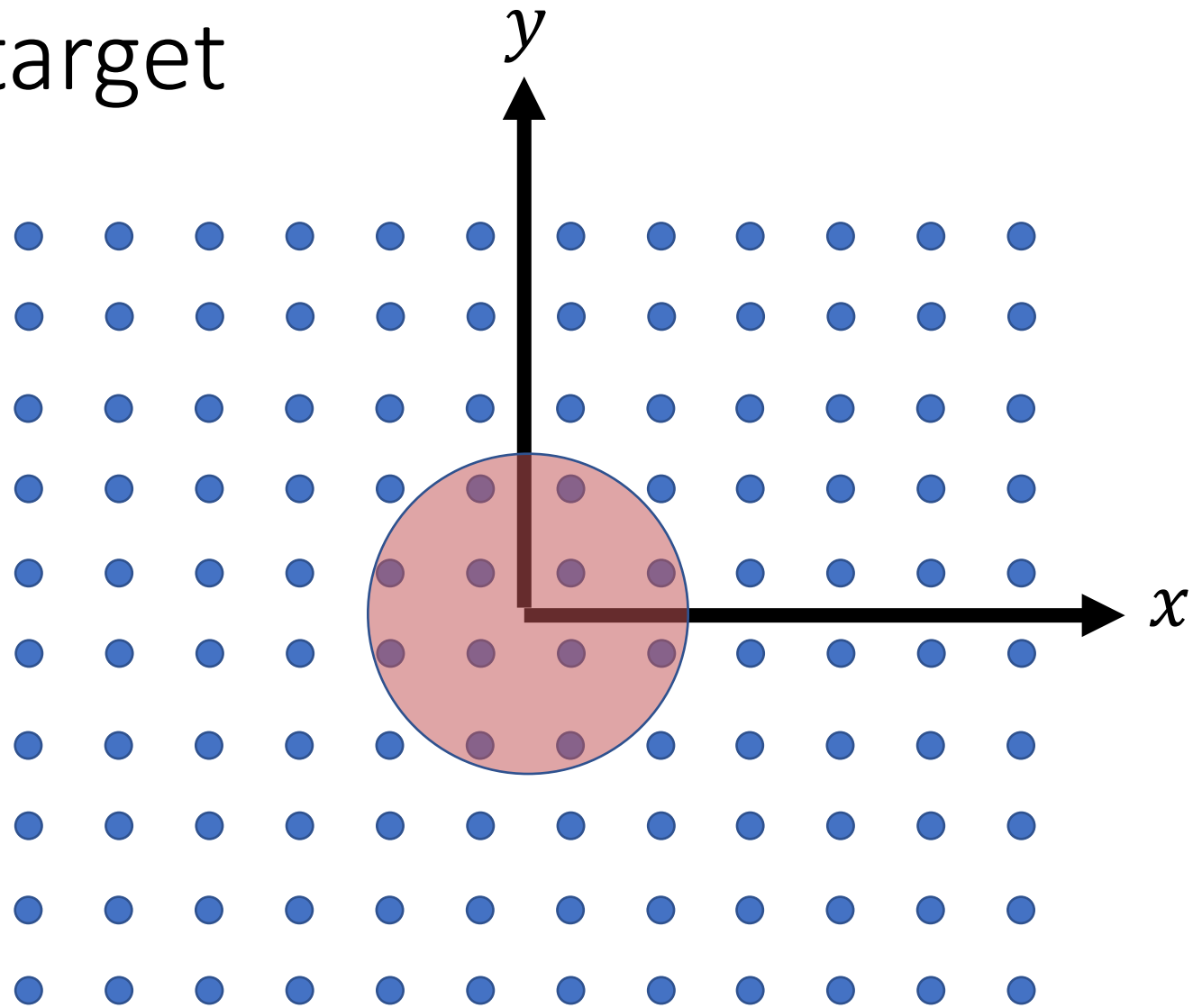
Notes: Open helperOC/ReachabilityWorkshop/tutorial_test.m
Right now it is set to run #1 on this slide.
The comments describe how to change the code depending on what you want to solve.

Making the grid



Notes: When making the grid, this is what you are essentially doing. N is the number of grid points in each dimension

Setting the target



Notes: The target in this case is a circle at the origin in position space with radius R. This target can be reached at any angle, so we make it a cylinder in 3D

Practice!

- Try running the different scenarios, raise your hand if you have any questions