Spring 2021

Stanford
AA 203: Optimal and Learning-based Control
Problem Set 3, due May 26 by 5:00 pm

Problem 1: HJ Reachability

Consider the goal of developing a self-righting quadrotor, i.e., a flying drone that you can chuck into the air at a range of poses/velocities which will autonomously regulate to level flight while obeying dynamics/controls/operational-envelope constraints. For this problem, we'll consider the 6D dynamics of a planar quadrotor:

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x \\ \frac{-(T_1+T_2)\sin\phi - C_D^v v_x}{m} \\ v_y \\ \frac{(T_1+T_2)\cos\phi - C_D^v v_y}{m} - g \\ \omega \\ \frac{(T_2-T_1)\ell - C_D^\phi \omega}{I_{yy}} \end{bmatrix}, \quad T_1, T_2 \in [0, T_{\max}], \tag{1}$$

where the state is given by the position in the vertical plane (x, y), translational velocity (v_x, v_y), pitch ϕ, and pitch rate ω; the controls are the thrusts (T_1, T_2) for the left and right prop respectively. Additional constants appearing in the dynamics above are gravitational acceleration g, the quadrotor's mass m, moment of inertia (about the out-of-plane axis) I_{yy}, half-length ℓ, and translational/rotational drag coefficients C_D^v and C_D^ϕ (see `problem_1/reachability.py` for precise values for these constants).

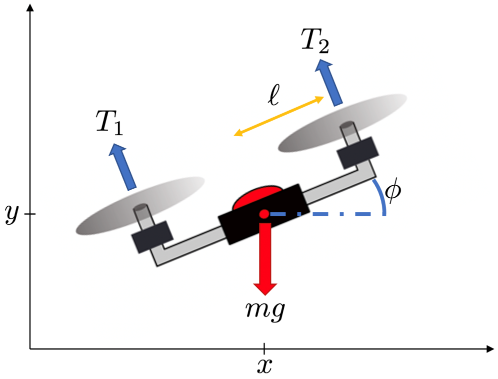

Figure 1: A planar quadrotor.

We will approach the problem of self-righting through continuous-time dynamic programming, in particular employing a Hamilton-Jacobi-Bellman (HJB) formulation.[1]

[1] One might also consider an HJI-based extension to handle worst-case disturbances (e.g., wind), but for simplicity in this exercise we'll consider just the undisturbed dynamics.

To help mitigate the curse of dimensionality, for the goal of achieving level flight we will consider reduced 4D dynamics with state vector $\mathbf{x} = \begin{bmatrix} y & v_y & \phi & \omega \end{bmatrix}^T$. In the context of these reduced dynamics, we define the target set

$$\mathcal{T} = [3, 7] \times [-1, 1] \times [-\pi/12, \pi/12] \times [-1, 1].$$

We assume that if at any time the planar quadrotor can reach this set, then, e.g., an LQR controller (linearized around hover) can take over and maintain level flight.

To bound the domain of our dynamic programming problem (and also to ensure that our quadrotor doesn't plow into the ground), in addition to the dynamics and control constraints given in Eq. (1) we would also like to constrain our planar quadrotor to stay within the operational envelope

$$\mathcal{E} = [1, 9] \times [-6, 6] \times [-\infty, \infty] \times [-8, 8].$$

Reaching the target set $\mathcal{T}$ while *avoiding* the obstacle set $\mathcal{E}^{\complement}$ (i.e., the set complement of $\mathcal{E}$) is referred to as a *reach-avoid* problem. If we can construct two real-valued, Lipschitz continuous functions $h(\mathbf{x}), e(\mathbf{x})$ defined over the state domain such that

$$\mathbf{x} \in \mathcal{T} \iff h(\mathbf{x}) \leq 0, \quad \mathbf{x} \in \mathcal{E} \iff e(\mathbf{x}) \leq 0,$$

i.e., $\mathcal{T}, \mathcal{E}$ are the zero-sublevel sets of h, e respectively, then it may be shown (see, e.g., [FCTS15], Theorem 1) that the value function $V(\mathbf{x}, t)$ defined as

$$\begin{aligned} V(\mathbf{x}_0, t_0) = \min_{\mathbf{u}(\cdot)} \; &\min_{\tau \in [t_0, 0]} h(\mathbf{x}(\tau)) \\ \text{s.t. } \; &\dot{\mathbf{x}}(\tau) = f(\mathbf{x}(\tau), \mathbf{u}(\tau)) \quad \forall \tau \in [t_0, 0] \\ &\mathbf{x}(\tau) \in \mathcal{E} \quad \forall \tau \in [t_0, 0] \\ &\mathbf{x}(t_0) = \mathbf{x}_0 \end{aligned}$$

(where f is the relevant portion of the full dynamics Eq. (1)) satisfies the HJB PDE[2]

$$\begin{aligned} \max \left\{ \frac{\partial V}{\partial t}(\mathbf{x}, t) + \min\{0, H(\mathbf{x}, \nabla_{\mathbf{x}} V(\mathbf{x}, t))\}, e(\mathbf{x}) - V(\mathbf{x}, t) \right\} = 0 \\ \text{where } H(\mathbf{x}, \mathbf{p}) = \min_{\mathbf{u}} \mathbf{p}^T f(\mathbf{x}, \mathbf{u}), \\ V(\mathbf{x}, 0) = \max\{h(\mathbf{x}), e(\mathbf{x})\}. \end{aligned}$$

Implementing an appropriate solver for this type of PDE is somewhat nontrivial (see, e.g., [Mit02] for details); for this exercise we will use an existing solver – you will be responsible for setting the problem up and interpreting the results.

[2] This is similar to the backward reachable tube HJI PDE mentioned in class (omitting the disturbance), where as before the inner min ensures the value function is nondecreasing in time (so that as BRT computation proceeds backward in time, the value function is nonincreasing at successive iterations, i.e., you get to "lock in" the lowest value you ever achieve). The outer max is the new addition in this formulation compared to what we saw in class, and may be interpreted as always making sure $V(x, t) \geq e(x)$ so that if $e(x) > 0$ (i.e., the state is outside of the operating envelope) then also $V(x, t) > 0$ (i.e., the state is outside the BRT of states that can reach the target collision-free).

If running `problem_1/reachability.py` directly or interacting with `problem_1/reachability.ipynb` using a local `jupyter` kernel, install the solver at the command line using `pip`:

```
> pip install --upgrade git+https://github.com/StanfordASL/hj_reachability
```

Otherwise, if using Google Colab to interact with `problem_1/reachability.ipynb`, run a cell containing

```
!pip install --upgrade git+https://github.com/StanfordASL/hj_reachability
```

We provide equivalent starter code in two formats: `problem_1/reachability.py` and `problem_1/reachability.ipynb`. When submitting code for parts (a)–(c) below, only provide the methods/functions that you've been asked to modify.

a) Subject to the control constraints $T_1, T_2 \in [0, T_{\max}]$, derive the locally optimal action that minimizes the Hamiltonian, i.e., for arbitrary $\mathbf{x}, \mathbf{p}$ compute

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \mathbf{p}^T f(\mathbf{x}, \mathbf{u}),$$

where f denotes the last four rows of the dynamics defined by Eq. (1). Use this knowledge to implement the method `PlanarQuadrotor.optimal_control`.

b) Write down a functional form for $h(\mathbf{x})$ such that $\mathbf{x} \in \mathcal{T} \iff h(\mathbf{x}) \leq 0$. Implement the function `target_set`.
Hint: Note that $a(\mathbf{x}) \leq 0 \ \wedge \ b(\mathbf{x}) \leq 0 \iff \max\{a(\mathbf{x}), b(\mathbf{x})\} \leq 0$. This means that if you have multiple constraints represented as the zero-sublevel sets of multiple functions, then the conjunction of the constraints may be represented as a pointwise maximum of the functions.

c) Write down a functional form for $e(\mathbf{x})$ such that $\mathbf{x} \in \mathcal{E} \iff e(\mathbf{x}) \leq 0$. Implement the function `envelope_set`.

d) Run the rest of the script/cells to compute $V(\mathbf{x}, -5)$ and take a look at some of the controlled trajectories; hopefully they look reasonable (see the note below if you're picky/have extra time, though if the quad rights itself and gets to the target set $\mathcal{T}$ that's sufficient for our purposes). Do not submit any trajectory plots; instead include a 3D plot of the zero isosurface (equivalent of a contour/isoline, but in 3D) for a slice of the value function at some fixed y value (e.g., $y = 7.5$ as pre-selected in the starter code). Explain why one of the bumps/ridges has the shape that it does.
Note: If the behavior of your control policy isn't as nice as you'd like (e.g., height/pitch oscillations), consider modifying your target set function $h(\mathbf{x})$ (e.g., by scaling how you account for each dimension in your construction). For the purpose of reachable set computation, at least theoretically[3] the zero-sublevel

[3] With a relatively coarse grid discretization and not-particularly-high-accuracy finite difference schemes/time integrators for PDE solving (sacrifices made so you don't have to wait for hours to see results), for numerical reasons the BRT may have some dependence on your formulation of $h(\mathbf{x})$.

set of the value function V (corresponding to the set of feasible initial states) is unaffected by the details of h as long as $h(\mathbf{x}) \leq 0 \iff \mathbf{x} \in \mathcal{T}$. In the context of dynamic programming to compute an optimal control policy, however, $h(\mathbf{x})$ also defines the terminal cost in a way that materially affects the policy once the set is reached (though in practice, this is where we'd have some other stabilizing controller take over).

e) In a few sentences, write down some pros/cons of this approach (i.e., computing a policy using dynamic programming) for a self-righting quadrotor vs. alternatives, e.g., applying model-predictive control. Potential things to think/write about: computational resources (time, memory) required for online operation, local/global optimality, flexibility to accommodate additional obstacles in the environment, bang-bang controls, etc.

Problem 2: MPC Feasibility

Consider the second-order, discrete-time LTI system

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{u}_k.$$

We want to compute a receding horizon controller for the case where the cost is quadratic, i.e., $h(\mathbf{x}_N) = \frac{1}{2}\mathbf{x}_N^T P \mathbf{x}_N$, $g(\mathbf{x}_k, \mathbf{u}_k) = \frac{1}{2}\mathbf{x}_k^T Q \mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T R \mathbf{u}_k$. Assume (if not otherwise stated) that $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $R = 0.01$, and that the system is subject to the input constraints

$$\begin{bmatrix} -\bar{u} \end{bmatrix} \leq \mathbf{u}_k \leq \begin{bmatrix} \bar{u} \end{bmatrix}, \quad k = 0, \ldots, N-1,$$

and to the state constraints

$$\begin{bmatrix} -\bar{x} \\ -\bar{x} \end{bmatrix} \leq \mathbf{x}_k \leq \begin{bmatrix} \bar{x} \\ \bar{x} \end{bmatrix}, \quad k = 0, \ldots, N.$$

(Note: this setup is similar to the setup in Example 12.1 (pg. 247) in the BBM book[4]). Let P_∞ be the solution to the algebraic Riccati equation:

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B (B^T P_\infty B + R)^{-1} B^T P_\infty A.$$

a) Implement the receding horizon control strategy for this system using CVXPY.

b) Let $\bar{x} = 5$, $\bar{u} = 0.5$, $N = 3$, $P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 10$, and $X_f = \mathbb{R}^2$. Simulate the closed-loop trajectories with initial states $\mathbf{x}_0 = [-4.5, 2]$ and $\mathbf{x}_0 = [-4.5, 3]$. You should be able to reproduce Figure 12.2 (pg. 248) in the BBM book.

[4] F. Borrelli, A. Bemporad, M. Morari. Predictive Control for Linear and Hybrid Systems, 2017. Available online at https://drive.google.com/file/d/1zaaZZjoXm73klAWfC62YlrUzujJOXUMt/view.

c) Let $\bar{x} = 10$, $\bar{u} = 1$, $N = 2$, $P = P_\infty$, and $X_f = 0$. Discretize the state space (pick a reasonable discretization step) and find the set of initial points leading to feasible closed-loop trajectories converging to the origin (i.e., the domain of attraction for the RHC policy).
Hint: To compute P_∞, either iterate the Riccati recursion until convergence (as in HW1) or take a look at scipy.linalg.solve_discrete_are.

d) Let $\bar{x} = 10$, $\bar{u} = 1$, $N = 6$, $P = P_\infty$, and $X_f = 0$. Discretize the state space and find the domain of attraction for the RHC policy.

e) Let $\bar{x} = 10$, $\bar{u} = 1$, $N = 2$, $P = P_\infty$, and $X_f = \mathbb{R}^2$. Discretize the state space and find the domain of attraction for the RHC policy.

f) Let $\bar{x} = 10$, $\bar{u} = 1$, $N = 6$, $P = P_\infty$, and $X_f = \mathbb{R}^2$. Discretize the state space and find the domain of attraction for the RHC policy.

g) Discuss and compare the results in parts c), d), e), and f).

h) Consider the case $\bar{x} = 10$, $\bar{u} = 1$, $P = P_\infty$, and $X_f = 0$. Consider several different values of N and discuss, by presenting simulation experiments, how the trajectory and its cost are affected by the choice of N (you should use an appropriate initial condition).

Problem 3: MPC Terminal Invariant and Stability

Consider the discrete-time LTI system $\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k$, with

$$A = \begin{bmatrix} 0.95 & 0.5 \\ 0 & 0.95 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and state and control constraints

$$\|\mathbf{x}_k\|_2 \leq 5, \qquad \|\mathbf{u}_k\|_2 \leq 1.$$

We wish to synthesize a controller to stabilize the system to the origin while minimizing a quadratic cost function

$$J = \mathbf{x}_N^T P \mathbf{x}_N + \sum_{k=0}^{N-1} \left(\mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k \right).$$

a) Consider the weights

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 1.$$

Propose a final condition X_f and a final cost P that, together, guarantee asymptotic stability and persistent feasibility of the closed-loop system.

b) While maximal positive invariant sets may be computed via iterative methods using tools from computational geometry, we will propose a large set and verify that it satisfies the definition of a positive invariant set. In particular, consider the ellipsoid[5] given by

$$X_f = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x}^T M \mathbf{x} \leq 1\} \tag{2}$$

with

$$M = \begin{bmatrix} 0.04 & 0 \\ 0 & 1.06 \end{bmatrix}. \tag{3}$$

Verify that this set is open-loop invariant.

Hint: We wish to verify the open-loop invariance of the set X_f. This equates to proving:

$$AX_f \subseteq X_f \tag{4}$$

where

$$AX_f = \{z \in \mathbb{R}^n : z = Ax, x \in X_f\} \tag{5}$$

is the set X_f transformed by the open-loop dynamics. This claim is equivalent to saying

$$x^T M x \leq 1 \implies x^T A^T M A x \leq 1. \tag{6}$$

A sufficient condition for the above holding is that

$$x^T A^T M A x \leq x^T M x \quad \forall x. \tag{7}$$

This holds iff

$$A^T M A \preceq M, \tag{8}$$

i.e., $M - A^T M A$ is positive semidefinite.

c) Solve the MPC problem from $\mathbf{x}_0^T = [-3, -2.5]$ and with planning horizon $N = 4$ in the following cases:

   i. With both the terminal constraint computed in the previous part, as well as the associated terminal cost.
   ii. With only the terminal cost.
   iii. With only the terminal constraint.
   iv. With neither the terminal cost of the terminal constraint.

   For all cases, submit two plots. The first plot should show the trajectory taken, all planned trajectories at all time, the terminal invariant set constraint, and the state constraints. The second plot should show the control actions versus iteration, satisfying the control constraints.

[5] Since we plan to use this invariant set as a terminal constraint in a convex program for our MPC implementation in this problem, we would like our invariant set to be convex, and an ellipsoid is a convenient choice of form. Computing the ellipsoid with maximum volume that is (i) contained in the state constraint set $\{\mathbf{x} : \|\mathbf{x}\| \leq 5\}$, and (ii) an invariant set with respect to the open-loop dynamics $\mathbf{x}_{k+1} = A\mathbf{x}_k$ can be posed as a Semidefinite Program (SDP); see Recitation 1.

Problem 4: Introduction to Reinforcement Learning

In this problem, you will investigate three different approaches to solving the LQR problem with unknown dynamics and reward. In particular, we will assume dynamics of the form

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t \tag{9}$$

with $\mathbf{x}_0 \sim \mathcal{N}(0, \Sigma_0)$ and $\mathbf{w}_t \sim \mathcal{N}(0, \Sigma_w)$, $\mathbf{x}_t \in \mathbb{R}^n$, $\mathbf{u}_t \in \mathbb{R}^m$ for all t. The running cost function is

$$c(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} \tag{10}$$

with Q, R positive definite. We will investigate three approaches to solving this problem: a model-based method; a model-free, value-based method, and a model-free, policy-based method. We will solve the discounted problem for a time-invariant policy,

$$\min_{\pi} \; \mathbb{E}[\sum_{t=0}^{N} \gamma^t c(\mathbf{x}_t, \pi(\mathbf{x}_t))] \tag{11}$$

All code is provided in the starter code repository.

For all problems, run the method for both the stochastic dynamics and the deterministic dynamics. This can be changed via a boolean variable in the starter code.

a) First, implement the optimal controller for the known cost and dynamics via solving the Riccati equation. Although we are considering a finite horizon problem, it is typical in reinforcement learning to use a stationary (non-time-varying) policy to reduce the complexity of the learning problem. Simulate this controller from many random initial states to get an estimate of the average cost. This will be the optimal baseline to which we will compare the performance of other controllers.

b) We will now implement a model-based approach to the presented problem. This approach will consist of three steps, iteratively performed:

- Fit a linear dynamics model using least squares regression.
- Fit a model for the cost function using least squares. Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{12}$$

To use linear regression for cost function estimation, rewrite the reward as

$$c(\mathbf{x}, \mathbf{u}) = \mathbf{q}^T \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ \vdots \\ x_1 x_n \\ x_2^2 \\ \vdots \\ x_n^2 \end{bmatrix} + \mathbf{r}^T \begin{bmatrix} u_1^2 \\ u_1 u_2 \\ \vdots \\ u_1 u_m \\ u_2^2 \\ \vdots \\ u_m^2 \end{bmatrix} \tag{13}$$

for some vectors $\mathbf{q}, \mathbf{r}$.

- Compute the optimal policy for the estimated dynamics and reward model via Riccati recursion.

Let L^* denote the optimal policy, computed in part (a) with exact model knowledge, and let L_i denote the policy computed at iteration i. Plot $\|L^* - L_t\|_2$ versus time. Also, plot the achieved cost in the episode versus episode number.

Hint: It is convenient to use the iterative formulation of least squares to update the model at every iteration, which can then be interleaved after every timestep with policy optimization. For a system for which we receive input-output pairs $(\mathbf{z}_t, \mathbf{y}_t)$ at each timestep, which obey the relationship

$$\mathbf{y}_t = C^T \mathbf{z}_t + \mathbf{w}_t \tag{14}$$

for zero mean normal $\mathbf{w}_t$, the iterative update can be written

$$\hat{P}_t = \hat{P}_{t-1} - \frac{\hat{P}_{t-1}\mathbf{z}_t\mathbf{z}_t^T\hat{P}_{t-1}}{1 + \mathbf{z}_t^T\hat{P}_{t-1}\mathbf{z}_t} \tag{15}$$

$$\hat{C}_t = \hat{C}_{t-1} + \frac{(\hat{P}_{t-1}\mathbf{z}_t)(\mathbf{y}_t^T - \mathbf{z}_t^T\hat{C}_{t-1})}{1 + \mathbf{z}_t^T\hat{P}_{t-1}\mathbf{z}_t} \tag{16}$$

with $\hat{P}_0 = I$. For $\hat{C}_0$, you may choose any initialization that satisfies positive definiteness constraints. We recommend initializing $\hat{A}_0$ and $\hat{B}_0$ to random matrices, and $\hat{Q}_0, \hat{R}_0$ to the identity matrix to satisfy the constraints.

c) Implement the policy iteration scheme for LQR presented in [BYB94]. Note that the iterative least square algorithm used in that work is the same as the above. Plot $\|L^* - L_i\|_2$ versus time, where L_i denotes the policy after improvement after every episode. Also, plot the achieved cost in the episode versus episode number.

Hint: This approach can be quite sensitive to various factors. Make sure your Q function estimate is symmetric by setting

$$\hat{Q}_t = \frac{1}{2}(\hat{Q}_t + \hat{Q}_t^T). \tag{17}$$

It is necessary to add exploration noise to the action selection to ensure convergence. Set the action to

$$\mathbf{u}_t = -\hat{L}_t\mathbf{x}_t + \boldsymbol{\epsilon}_t \tag{18}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$.

d) Implement Monte Carlo policy gradient (REINFORCE) for a Gaussian policy of the form

$$\pi_W(\mathbf{u} \mid \mathbf{x}) = \mathcal{N}(W\mathbf{x}, \Sigma) \tag{19}$$

and a choice of Σ. Specifically, find $\nabla_W \log \pi_W(\mathbf{u} \mid \mathbf{x})$ and then implement the Monte Carlo policy gradient algorithm. Plot $\|L^* - L_i\|_2$ versus time, where L_i denotes the policy at the end of each episode. Also, plot the achieved cost in the episode versus episode number.

Hint: Naïve policy gradient is quite unstable on this problem, so we will provide some hyperparameters that should enable you to get a basic version working. First, initialize your policy as the zero matrix, and set $\Sigma = 0.1I$. As a step size parameter α, we recommend values in the neighborhood of 10^{-13}. These hyperparameters should yield a functional if unimpressive policy gradient algorithm, if used on top of the standard REINFORCE policy gradient. Expect this approach to take at least 5000 episodes to yield visible performance improvement. This approach will perform one to two orders of magnitude worse than the other methods; just report the performance, do not try to optimize this method to compete with the others.

There are numerous ways to improve the performance of the policy gradient method. One simple approach is to *whiten* the reward signal. This can be done by maintaining the mean of the received rewards, as well as the empirical standard deviation, and subtracting off this mean and dividing this difference by the standard deviation. This approach shifts and rescales the reward to have mean zero and unit covariance, and should result in substantially improved performance. With this modification, choose a step size in the neighborhood of $\alpha = 10^{-8}$. This modification is not required.

e) Each of the previous methods required access to some amount of problem-specific information. For each of the four methods above, describe the problem information that was used in the design of the algorithm. How does the amount of assumptions for each of the methods compare to the performance of the methods?

Learning goals for this problem set:

Problem 1: To gain familiarity with tools for computing HJ reachability; develop an understanding of sublevel sets in the context of backward reachability.

Problem 2: To understand the basics of feasibility in model predictive control.

Problem 3: To gain an intuitive understanding of the role of terminal costs and invariant constraints in model predictive control.

Problem 4: To understand the most fundamental reinforcement learning algorithms and their strengths and weaknesses.

References

[BYB94] S. J. Bradtke, B. E. Ydstie, and A. G. Barto, *Adaptive linear quadratic control using policy iteration*, Proc. IEEE Conf. on Decision and Control, 1994.

[FCTS15] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, *Reach-avoid problems with time-varying dynamics, targets and constraints*, Hybrid Systems: Computation and Control, 2015.

[Mit02] I. M. Mitchell, *Application of level set methods to control and reachability problems in continuous and hybrid systems*, Ph.D. thesis, Stanford University, 2002.