How to run the experiments

There are 3 notebooks in this folder: [DataCollection], [PlottingCore], and [Plotting Constraint Violation.]

Run the [DataCollection] notebook to carry out the simulations, and store the simulation results as Python pickle files(.pkl) in the data/ folder.

Run [Plotting Core] to plot the regret and estimation error figures. Run [Plotting Constraint Violation] to plot the simulation trajectory figures showing constraint violation/satisfaction. The figures are stored in the figures/ folder.

Background

This folder contains experiments for the SafeLearning algorithm of systems with non-zero equilibrium points. The non-zero equilibrium is achieved by setting the base controller in the form of $K_{stab}x + b$, so that the system dynamics is

$$x_{t+1} = (A - BK_{stab})x_t + B(-b + u_t) + w_t \tag{1}$$

The relationship between the equilibrium position($\bar{x}$) and b is

$$\bar{x} = (A - BK_{stab} - I)^{-1}Bb \tag{2}$$

assuming $A - BK_{stab} - I$ is invertible.

The b value needs to be passed into the method $env.step(x, b)$ for such offset in equilibrium point to take effect.

Dynamically changing b in Safe Learning

We know that since there are e_x, e_u corrections in the DAP algorithm, choosing a b such that $\bar{x}$ becomes very close to the $x_{max/min}$ boundaries often results in infeasibility of the algorithm. Therefore we developed the following method to alleviate such infeasibility issue.

Recall the DAP algorithm solves an optimization problem in the form of

$$\begin{aligned} &\min_{M} \text{LQR Cost about } M \\ &s.t.\ M \in \Omega(A, B, H, e_x, e_u, K_{stab}, b) \end{aligned} \tag{3}$$

The implementation of Ω can be found in the method controllers.SafeDAP.omega_phi. The SafeDAP class can be found in the file controllers.py

We first specify a b_{target} which results in an equilibrium point x_{target} according to equation (2). See the variable b_target in the [DataCollection] notebook.

Typically, b_{target} is chosen so that x_{target} is very close to the boundary. Plugging in such a b_{target} into Ω often results in infeasiblity$(\Omega = \emptyset)$. Therefore, the Safe Learning algorithm will use a b^* that makes Ω nonempty and is as close to b_{target} as possible. That is

$$\begin{aligned} &b^* = \arg\min_{b,M} ||b - b_{target}|| \\ &s.t.\ M \in \Omega(A, B, H, e_x, e_u, K_{stab}, b) \end{aligned} \tag{4}$$

$$s.t.\ M \subset \Omega(A, B, H, e_x, e_u, K_{stab}, b)$$

The decision variable M only decides the feasibility of equation (4), but does not contributes to the objective. The implementation of (4) is in controllers.SafeDAP.solve_b_star()

During Safe Learning, every time after the $\hat{\theta}$ is updated, we first solve (4) to obtain b^*(replacing A, B with $\hat{A}, \hat{B}$), then plug b^* in (3) in place of b to get a surely feasible solution to M. See the first two lines of the $phase()$ local function within the $SafeAdaptiveSim()$ function in the last cell of [DataCollection] notebook.

Meanwhile, e_x, e_u becomes smaller after each episode according to a fixed decay rate(see the variable $decay$ in the [DataCollection] notebook). It can be observed that as e_x, e_u get smaller, b^* will get closer to b_{target}.