3/31 Recitation

This is how we take an L1 cost (nonsmoth) and convert into a linear cost by adding a slack variable and some linear inequalities. Zac went over one way to do it in class, here is another.

L1 trick

this will encourage sparsity in x

$$\min_x \quad f(x) + |x|_1$$
$$\text{s.t.} \quad c(x) = 0$$

ex:

$$\min_x \quad |x|_1$$
$$\text{s.t.} \quad Ax = b$$

add slack t

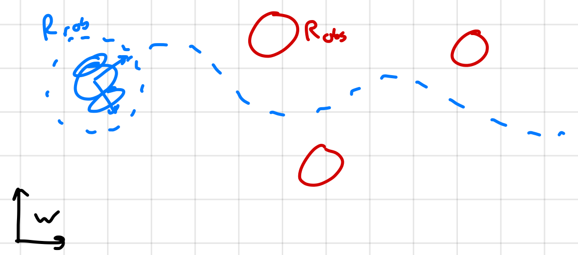
$$\min_{x,t} \quad f(x) + \overbrace{1^T t}^{\Sigma t}$$

$$c(x) = 0$$
$$t \geq x$$
$$t \geq -x$$

$$x = \begin{bmatrix} 1 \\ -2 \\ 3 \\ -4 \end{bmatrix} \quad t = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Useful for
- encouraging sparsity in x
- identifying important features
- min fuel/actuator use
- "LASSO"

Avoiding Collisions



Search/sample based planners are the only option for really hard problems (finding your way through a maze), but can be slow/inefficient/not the right tool for easier problems.

① Search based, sample based planners

- A^* if grid exists (search)
- RRT, RRT^* (sample)

Control Barrier Functions are an easy and cheap way to modify your control inputs to make them "safe". This is not optimal though, since the original control didn't consider the obstacles.

② CBF

$$u_{safe} = f_{CBF}(u)$$

③ traj opt + collision constraints

Doing trajectory optimization while directly considering the collision avoidance constraints is the best way to find locally optimal solutions to this problem, but it also can be slow/unreliable.

CBF

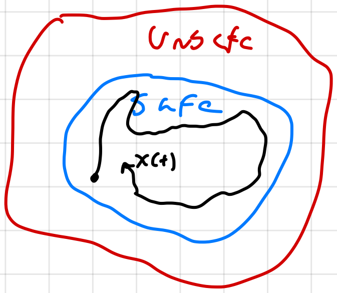
CBF's are based on a safety index phi

$\phi(x) \leq 0$ for safe

$$\text{norm}(r_{rob} - r_{obs})^2 \geq (R_{rob} + R_{obs})^2$$

$$0 \geq (R_{rob} + R_{obs})^2 - (\text{norm}(r_{rob} - r_{obs}))^2 = \phi(x)$$

"forward invariant"



In the CBF literature you will see "forward invariance" talked about a lot. This just means once your state is within the set of safe states, we want to design a controller that is "forward invariant", and always stays within this set.

$$f_{CBF}(u_0) = \underset{u}{\operatorname{argmin}} \quad \|u-u_0\|_2^2$$

$$\text{s.t.} \quad \dot{\phi}_1 \le \lambda_1 \phi_1$$
$$\phi_2 \le \lambda_2 \phi_2$$
$$\vdots$$
$$\underline{u} \le u \le \bar{u}$$

CBF is solving an optimization problem (QP usually) that looks for the closest control input to the nominal control, but makes sure there is a derivative condition satisfied to maintain forward invariance.

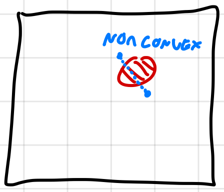
① $\lambda < 0$

② $\dot{\phi} = (\nabla_x \phi(x))^T \dot{x}$, if $\dot{x} = f(x,u)$, $\dot{\phi} = \nabla_x \phi^T f(x,u)$

③ $\dot{x} \approx \dfrac{(f_{discrete}(x,u) - x)}{dt}$, Linearize $f_{discrete}$ if NL

Sometimes our control cannot directly influence the safety index time derivative (phi dot), in this case, we can approximate xdot as (x2 -x1)/dt. This expression just has to be linear in u. See the code with CBF for a double integrator.

Motion planning w/ Collision Avoidance



$$c(x) = \operatorname{norm}(r_{rob} - r_{obs})^2 - (R_{rob} + R_{obs})^2 \ge 0$$

Collision avoidance constraints will ALWAYS be nonconvex. This is because a convex set requires any interpolation between 2 feasible points is also feasible, which is not true if there is a dead zone in the middle of the state space where the obstacle is.

$$\min_{x_{1:N},\, u_{1:N-1}} \quad \ell_N(x_N) + \sum_{i=1}^{N-1} \ell(x_i, u_i)$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

$$c(x_k) \ge 0 \quad \text{Collision avoidance}$$

This is a normal trajopt problem with an added collision avoidance constraint. This will give us a locally optimal solution that directly reasons about the collision constraint. Unfortunately, since it's a nonconvex problem, there are no guarantees for speed/feasibility/quality of solution, so test your controller according.