Introduction to Numerical Optimization – Project

Real-time flight control

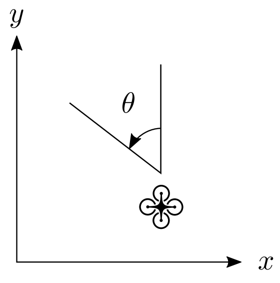
A drone must quickly fly from its starting position s_0 to a known destination s^* in a cluttered two-dimensional environment. The drone velocity $v = 6m/s$ is fixed. The drone can only choose its direction of flight and may change it over time.

Each data instance is stored in a `.jld` file in the `data` folder of the archive available on the web page of the course. The data structure is

```
struct Data
  start::Tuple{Float64,Float64,Float64}    # drone starting point and direction (x,y,theta)
  destination::Tuple{Float64,Float64}      # destination (x,y)
  obstacles::Array{Tuple{Float64,Float64}} # array of (x_i,y_i) for each obstacle i
end
```

The `data.jl` script contains the data structure as well as functions to load, save, and generate data instances.

Although all obstacles are available in the data structure, the drone can only see the obstacles which are within its sight distance r. At any time t, the drone may change its direction of flight. However, in its decision-making process, it may only consider the obstacles whose distance from the current drone location is at most $r = 20m$, and possibly the obstacles which were seen before.



The drone must respect the dynamics of flight. The system state variables are $s = (x, y, \theta)$, representing the drone's current location and direction. The dynamical system is represented by a system of differential equations of motion

$$f(s) = \dot{s} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -v\sin(\theta) + w \\ v\cos(\theta) \\ -K(\theta - u) \end{bmatrix}$$

where w is the crosswind speed along the x axis only, and u is the input control. By assigning an angle value to u, the drone tells the system which direction it wants to take. However, the dynamics prevent the drone from changing its direction of flight instantly. The feedback $-K(\theta - u)$ will make the drone change its actual direction θ over time to eventually match the desired direction u. The larger K, the faster the drone can change its direction. Use $K = 0.2$. The project is not about computing and analysing the solution of this system of differential equations, but rather about choosing a viable direction of flight u by avoiding the obstacles and respecting the system dynamics considered as an internal black box.

Lyapunov stability

The Lyapunov stability criterion demonstrates stability by proving the existence of a Lyapunov function $V(s) : \mathbb{R}^3 \to \mathbb{R}$ such that

$$V(s) \geq m \quad \forall s \in \mathbb{R}^3 \tag{1}$$
$$V(\bar{s}) = m \tag{2}$$
$$\dot{V}(s) = \langle \nabla V(s), \dot{s} \rangle \leq 0 \quad \forall s \in \mathbb{R}^3 \tag{3}$$
$$\dot{V}(\bar{s}) = 0 \tag{4}$$

where $\bar{s}$ is an equilibrium point of the dynamical system, i.e. $f(\bar{s}) = 0$. m is often set to 0 in the literature. However, there is no need for m to be 0. Any value suffices. The purpose of (1) and (2) is to make sure that $\bar{s}$ is also a minimum of V.

The Lyapunov stability criterion will not be used as such in this project because

1. there is no meaningful equilibrium (not even the destination point) to f;
2. had a trajectory of flight towards the destination been obtained, the drone may still collide with obstacles;
3. the Lyapunov stability criterion is for autonomous systems only, with no input control u.

Hence, as such, the Lyapunov stability criterion is not usable. We instead consider the Lyapunov *barrier* variant whose purpose is to prove whether the input control u leads to a trajectory free of obstacles.

Barrier

Consider two semialgebraic sets $\mathcal{S}_{\text{safe}}$ and $\mathcal{S}_{\text{unsafe}}$. The Lyapunov barrier function $V(s) : \mathbb{R}^3 \to \mathbb{R}$ has the following properties:

$$V(s) < b \quad \forall s \in \mathcal{S}_{\text{safe}} \tag{5}$$
$$V(s) > b \quad \forall s \in \mathcal{S}_{\text{unsafe}} \tag{6}$$
$$\dot{V}(s) \leq 0 \quad \forall s \in \mathbb{R}^3 \tag{7}$$

All trajectories starting in $\mathcal{S}_{\text{safe}}$ are guaranteed to remain in $\mathcal{S}_{\text{safe}}$ since V is non-increasing along trajectories. By forcing the obstacles to remain in $\mathcal{S}_{\text{unsafe}}$ and forcing the drone to start in $\mathcal{S}_{\text{safe}}$, this will guarantee that the drone will never collide with the obstacles.

In this project, $V(s)$ is assumed to be a degree d polynomial.

A. Relaxations

1. Consider the SDP relaxation of the barrier function V with $d = 2$ and $w = 0$. Write the model as a feasibility problem and solve it for $u = 0$ and $u = \pi$. For the `obs_behind` and `obs_behind_side` datasets,
   (a) plot the evolution of V along the segment between the drone starting point and the destination;
   (b) plot the value of $V(x, y, 0)$ in the two-dimensional region of interest.
2. Consider the SOCP relaxation with $d = 2$, $u = 0$ and $w = 0$. An SOCP relaxation of $X \succeq 0$ can be obtained by imposing all 2×2 sub-matrices of X to be semidefinite positive:

$$\begin{pmatrix} X_{ii} X_{ij} \\ X_{ji} X_{jj} \end{pmatrix} \succeq 0 \quad \forall i, j$$

   All 2×2 semidefinite positive matrices can be written as SOC constraints with

$$X_{ii} \geq 0$$
$$X_{jj} \geq 0$$
$$X_{ij} = X_{ji}$$
$$\left\| \begin{pmatrix} 2X_{ij} \\ X_{ii} - X_{jj} \end{pmatrix} \right\| \leq X_{ii} + X_{jj}$$

   Perform the same plots with the same datasets.

3. Consider the LP relaxation with $d = 2$, $u = 0$, and $w = 0$. An LP relaxation of $X \succeq 0$ can be obtained by imposing X to be diagonally dominant:

$$X_{ii} \geq \sum_{j \neq i} |X_{ij}| \quad \forall i$$

   Perform the same plots with the same datasets.

4. Change $d = 4$ and add the following constraint and objective function:

$$\begin{aligned} &\min h \\ &\text{s.t.}\, V(s^*) \leq h \end{aligned}$$

   where s^* is the destination. Discuss these modifications and their effect on the two datasets with the three relaxations. Do solvers still manage to converge? Is the solution reliable?

5. Suppose the drone chooses $u = \frac{3\pi}{10}$ as its input angle. The actual trajectory of flight can be computed with the forward Euler method:

$$s_{t+1} = s_t + \Delta_t f(s_t)$$

   Use the constraint

$$\dot{V}(s_t) \leq 0 \quad \forall t$$

   Plot the trajectory and the evolution of V along that trajectory, for the `obs_behind_side` dataset and $w = 0$.

6. There is an unknown albeit bounded crosswind. Discuss (no need to solve) how to model that a solution remains feasible for all $w \in [-2, 2]$.

B. Full flight

This section involves choosing a viable input control $u \in [-\pi, \pi]$, flying for some time along the trajectory computed with the Euler method, choosing a new u, and repeating the process until the destination is reached.

7. Instead of declaring u as an optimization variable, solve k problems, each one with a fixed specific value of u. What is the advantage of fixing the value of u instead of declaring it as a variable? Is this appropriate for a real-time flight application to base the control decision solely on a few values of u?

8. Out of the k solutions, how would you choose a viable u which also helps the drone getting closer to its destination?

9. Write your iterative optimization process to make the drone reach its destination despite some crosswind w, for several datasets. Choose whichever relaxation you deem best suited between SDP, SOCP and LP.

C. Sensitivity analysis

For this section, use a linear solver's sensitivity analysis.

10. With your model from question 3 and the `obs_one_behind` dataset, give the b interval outside outside of which the optimal basis would change.