Stanford **Spring 2023**

AA 203: Optimal and Learning-based Control
Homework #1
Due April 24 by 11:59 pm

Learning goals for this problem set:

Problem 1: Learn how to construct stabilizing controllers by exploiting structure in the dynamics.

Problem 2: Gain familiarity with the Pontryagin maximum principle (PMP), study the structure of time-optimal trajectories, and learn about singular arcs.

Problem 3: Implement an indirect method for optimal control and gain familiarity with JAX.

1.1 *Backstepping.* Consider the strict-feedback system

$$\begin{aligned}\dot{x} &= f(x) + B(x)z \\ \dot{z} &= u\end{aligned},$$

with $x \in \mathbb{R}^n$ and $z, u \in \mathbb{R}^m$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $B : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are known smooth functions, and $f(0) = 0$.

Suppose the subsystem $\dot{x} = f(x) + B(x)z$ can be stabilized by a smooth feedback law $z = \phi_0(x)$ with $\phi_0(0) = 0$, i.e., the closed-loop system $\dot{x} = f(x) + B(x)\phi_0(x)$ is *globally asymptotically stable* with respect to the origin $x = 0$. Moreover, suppose we know a smooth, positive-definite, radially unbounded Lyapunov function $V_0 : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and positive definite function $\rho : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ satisfying

$$\nabla V_0(x)^\mathsf{T}(f(x) + B(x)\phi_0(x)) \leq -\rho(x),$$

for all $x \in \mathbb{R}^n$.

We now consider the entire (x, z)-system, which we can only control through $u \in \mathbb{R}^m$. We want to use our knowledge of a stabilizing controller for the inner x-dynamics and the strict-feedback form of the (x, z)-dynamics to "back out" a stabilizing controller for the entire system.

Use the Lyapunov candidate function

$$V_1(x, z) = V_0(x) + \frac{1}{2}\|z - \phi_0(x)\|_2^2$$

to find a stabilizing controller $u = \phi_1(x, z)$ for some function $\phi_1 : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$ that ensures $(x, z) \to (0, 0)$. Notice that V_1 comprises the "inner" Lyapunov function V_0 and a penalty term for the difference between z and the value of the "inner" stabilizing control. Explicitly derive the function ϕ_1 and rigorously describe why it stabilizes the (x, z)-system using Lyapunov theory (i.e., prove $V_1(x, z)$ is positive-definite and radially unbounded, and $\dot{V}_1(x, z)$ is negative-definite along trajectories of the (x, z)-subsystem in closed-loop with $u = \phi_1(x, z)$).

1.2 *Singular arc for Dubins' car.* The kinematics of Dubins' car are described by

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= u\end{aligned},$$

where $(x, y) \in \mathbb{R}^2$ is the car's position, $\theta \in \mathbb{R}$ is the car's heading, $v > 0$ is the car's constant known speed, and u is the controlled turn rate. The turn rate is bounded, i.e., $u \in [-\bar{\omega}, \bar{\omega}]$, where $\bar{\omega} > 0$ is a known constant.

The car starts at $(x, y) = (0, 0)$ with a heading of $\theta = 0$ at $t = 0$. We want the car to drive to $(x, y) = (0, c)$ in the least amount of time possible, where $c > 0$ is a given constant.

(a) Use Pontryagin's maximum principle to express the optimal control input $u^*(t)$ as a function of the optimal co-state $p^*(t) := (p_x^*(t), p_y^*(t), p_\theta^*(t)) \in \mathbb{R}^3$.

Hint: You should discover that the maximum condition for $u^*(t)$ is not informative whenever $p_\theta^*(t) \equiv \bar{p}_\theta$ for a particular fixed value $\bar{p}_\theta \in \mathbb{R}$. When such a lack of information persists over a non-trivial time interval, i.e., any time interval $[t_1, t_2]$ with $t_2 > t_1 \geq 0$, this is known as a *singular arc*. To compute $u^*(t)$ in this case, use the fact that $p_\theta^*(t) \equiv \bar{p}_\theta$ is constant in time along this singular arc.

(b) Use boundary conditions to argue why $p^*(t)$ might end in a singular arc. Suppose we know $p^*(t)$ begins on a non-singular arc, then switches once to and ends on a singular arc. For this particular case, argue why $u^*(0) = \bar{\omega}$ and describe the optimal state trajectory $(x^*(t), y^*(t), \theta^*(t))$ and control trajectory $u^*(t)$ in words without explicitly deriving them.

1.3 *Single shooting for a unicycle.* Consider the kinematic model of a unicycle

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \end{aligned} ,$$

where (x, y) is the planar position of the vehicle, θ is its heading angle, v is its forward velocity, and ω is its angular velocity. Overall, the state and control input for this system are $x := (x, y, \theta) \in \mathbb{R}^3$ and $u := (v, \omega) \in \mathbb{R}^2$, respectively. We have overloaded x to denote both horizontal position $x \in \mathbb{R}$ and the full state vector $x \in \mathbb{R}^3$.

Our task is to drive the vehicle from the starting configuration $x(0) = (0, 0, \pi/2)$ to the target configuration $x(T) = (5, 5, \pi/2)$ in minimum time with as little control effort as possible. To this end, we consider the objective

$$J(x, u) = \int_0^T \left(\alpha + v(t)^2 + \omega(t)^2\right) dt,$$

where $\alpha > 0$ is a chosen constant weighting factor and T is the free final time.

(a) Derive the Hamiltonian and necessary optimality conditions, specifically

  i. the ODE for the state and co-state,

  ii. the optimal control as a function of the state and co-state, and

  iii. the boundary conditions, including the additional condition for free final time T.

Hint: Since the control set is unbounded, use the weak maximum condition.

In practice, you might use a boundary value problem (BVP) solver from an existing computing library (e.g., `scipy.integrate.solve_bvp`), but in this problem we will use a bit of nonlinear optimization theory and JAX to write our own!

(b) In the file `starter_single_shooting_unicycle.py`, complete the implementations of `dynamics`, `hamiltonian`, `optimal_control`, and `pmp_ode`. Use $\alpha = 0.25$.

In the single shooting method, we need to initialize estimates of the initial co-state $p(0)$ and final time T. We then integrate the state and co-state dynamics forward in time from $t = 0$ to $t = \hat{T}$, at which point we check whether the terminal boundary conditions are satisfied.

(c) Use the ODE integration from `pmp_trajectories` to complete `boundary_residual`, which should compute a measure of how far off each of your terminal boundary conditions is from satisfaction, given guesses for the initial co-state $p(0)$ and final time T.

(d) Finally, in `newton_step` and `single_shooting`, implement the Newton-Raphson root-finding method for `boundary_residual`. Now, if you provide an appropriate guess for the initial costate and final time, you can run `python3 starter_single_shooting_unicycle.py` and see a plot of the optimal solution. You may find that whether or not your BVP solver converges to a solution is highly dependent on the quality of your initial guess – indeed, initialization is a major challenge when applying indirect methods for optimal control!

Hint: For finding roots of a function $f : \mathbb{R}^n \to \mathbb{R}^n$, each iteration of the Newton-Raphson method entails improving a current best guess $x^{(k)}$ at iteration k using the update rule

$$x^{(k+1)} = x^{(k)} - \frac{\partial f}{\partial x}(x^{(k)})^{-1} f(x^{(k)}).$$

Submit your completed version of `starter_single_shooting_unicycle.py` and the generated plot.