Professor Fearing EECS192/Assignment #2 Car Steering v 1.02 Spring 2021
Due by 6 pm Fri. April 2 on BCourses in .pdf. One assignment per team.

For this assignment you will be using `V-rep` for dynamic simulation of a car. (V-rep Pro EDU V3.6.2 can be downloaded for Mac/Ubuntu/Windows at `http://www.coppeliarobotics.com/previousVersions`).

The car model `cory-track-fastcar.ttt`, and python code can be found in the repository: `https://github.com/ucb-ee192/simulator-pub`.

V-rep is the server, and you will modify a Python function `control_loop()` (in `controller.py`) to drive the simulated car. The simulator estimates `lat_err` = `yDist` = y_a, the lateral error from the track at a distance approximately 2 car lengths in front, and takes 2 inputs: 1) commanded steering angle `steerAngle` = δ, `car.set_steering()` and 2) longitudinal velocity set using `car.set_speed(3.0)`. `controller.py` will run the car but it is not tuned. Also the line detector is quite basic. You will modify and extend the code to get a good car controller. The simulator runs ok at 10 ms time step (dynamics are updated at 10X).

Note: the ODE physics engine starts with a random seed, so every run can be slightly different.

For all plots, time axes should be in seconds, and lateral errors in m or cm. A Python plotting tool `log-visualizer.py` is provided in the repo. Combine all plots into a single .pdf file to upload.

(25 pts) 1. Comparison of CheckPoint 7 Telemetry and Simulation for Step
Note, the car can be placed in the simulator on the long step section using the V-Rep object/item shift tool.
(5 pts) a. What parameters were used in your car for C7: velocity, control gains, etc?
For next parts, use, as much as possible, line finding algorithm and controller used in real car in `controller.py`.
(8 pts) b. Show "waterfall plot" and line tracking for real and simulated car (you are welcome to use or modify `log-visualizer.py`.
(8 pts) c. Show plots for error, steering angle, and velocity for real and simulated car on one page (6 subplots).
(4 pts) d. Compare track error for simulation and real data. How do they compare? Are peak overshoot and damped frequency similar for simulation and real car?

For each part below, plot on 1 page a) actual x vs y position of car b) lateral error y_a as function of time, c) steering angle δ as function of time. For part 3, also plot d) $\dot{y}_a$ as a function of time. Plots should be in physical units (e.g. y_a in meters). For each part, list constants used. (k_p should have units of radians/meter, etc.)

(25 pts) 2. Steering Simulation- proportional control
(22) a. Using pure position control, $\delta = k_p y_a$, choose a fixed speed V and k_p that allows the car to successfully complete the track without hitting any cone(s). Report k_p and V.
(3) b. Specify worst-case overshoot (cm), and note on plots where this occurs.

(25 pts) 3. Steering Simulation- proportional + derivative control
(20) a. Using PD control $\delta = k_p y_a + k_d \dot{y}_a$, find the highest fixed speed that allows the car to successfully complete the track without hitting any cone(s). Report k_p, k_d, and V. (Estimate $\dot{y}_a \approx \frac{y_a[n]-y_a[n-1]}{\cancel{20}\ 10 \text{ ms}}$.)
(3) b. Specify worst-case overshoot (cm), and note on plots where this occurs.
(2) c. Briefly comment on any differences in performance observed between P and PD type control, such as overshoot or maximum speed.

(25 pts) 4. Steering steering servo speed limit
(23) a. The default steering servo is set unrealistically fast (line 33 in ~~`controller.py`~~ `carInterface.py`): `steering_slew_rate = 600/.16`, (600 degrees in 160 ms).
Change to a more realistic rate of 60 degrees in 160 ms, and repeat question 3.
(2) b. Briefly comment on any differences in performance observed between fast and slow steering servo response. What track features cause the most problems?