Advanced Controls

In the controls lab, we used feedback control to minimize the difference in distance traveled between the two wheels ($\delta[n] = d_L[n] - d_R[n]$). When perturbations cause one wheel to get ahead of the other, the feedback controller adjusts the left and right inputs to correct the error. So, when δ is nonzero, the controller turns the car to send δ back to zero. We can exploit this to make the car turn in a controlled fashion.

Recall that our inputs are:

$$u_L[n] = \frac{v^* + \beta_L}{\theta_L} - \frac{k_L}{\theta_L}\delta[n]$$

$$u_R[n] = \frac{v^* + \beta_R}{\theta_R} + \frac{k_R}{\theta_R}\delta[n]$$

Note that the first term in each input is always constant, while the second changes over time due to feedback.

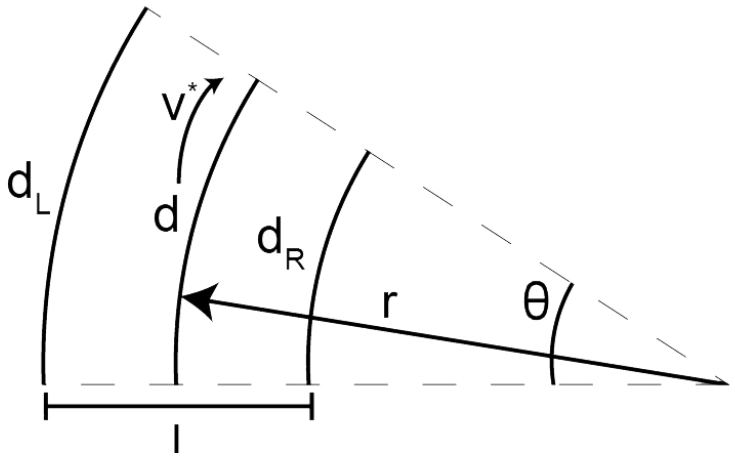
Let's say the right wheel has moved more than the left ($d_R[n] > d_L[n]$), so δ is negative. This causes $u_R[n]$ to decrease and $u_L[n]$ to increase, thereby correcting the disturbance. However, if we artificially increase δ's magnitude by some factor, this would cause $u_R[n]$ to decrease and $u_L[n]$ to increase past the point of straightness: put differently, it causes the car to turn right (if the car's left wheel has travelled further than the right wheel, the car is turning right). Let's explore this idea more formally.

Turning via Reference Tracking

We would like the car to turn with a specific radius r and velocity v^*. To turn smoothly, we want δ to change at a particular rate: rather than just adding a constant offset to δ, we will include a time-dependent reference factor, making δ a function of turn radius r, velocity $v*$, time n, and distance bw the car wheels in centimeters l. Each tick corresponds to approximately 1cm of wheel circumference, so $l = d$.

1. What radius would we use if we want the car to go straight?
2. Inspect the figure below. Write $\delta[n]$ as a function of r, v^*, l, n using the following variables:
   - n - timestep
   - r [cm] - turning radius
   - d [ticks] - distance traveled by the center of the car
   - l [cm] - distance between the centers of the car's wheels
   - ω [radians/tick] - angular velocity
   - θ - angle traveled

Hint: you will find the formula for arc length useful.



Implementing Turns

Before actually implementing turning on the Launchpad, we need to take into account the fact that the sampling periods of our encoders (aka data collection) and our control loop might be different. We'll denote the data collection period by T_d, and the control period by T_c. To see how this fits into your calculated turning reference, note that the units of v^* are ticks/T_d and the units of n are seconds/T_c, or equivalently, ticks: T_c is in seconds, so since we know n is in ticks, we can restate it as seconds/T_c for easier conversion.

T_d is an integer multiple of T_c, and we'll define another variable m as $\frac{T_d}{T_c}$. In our case, $T_c = 100$ms, and $T_d = 500$ms, so $m = 5$. Therefore, **we replace** v^* **with** $\frac{v^*}{m}$, which is in units of ticks/T_c.

Now you are ready to finish the lab! Go to the Jupyter notebook and follow the instructions to do so.

Notes written by Mia Mirkovic (2019)