Drift Parking

In this lab, you will design and implement an open-loop drift maneuver on your BARC. Your goal will be to get your car to slide into a parking spot, similar to the expert drift parking maneuver shown **here**.

Task 1 Dynamic Bicycle Model

In this section of the lab, we are going to analyze the dynamics of drift under steady state conditions. We start by using the dynamic bicycle model and Pacejka tire model. The dynamic bicycle model (depicted in Fig. 1) is a single rigid body model with lumped rear wheel and front wheels. We assume the vehicle is rear-wheel driven (i.e. only the rear wheels apply an input force). The vehicle model is derived by writing out the balance of linear momentum and balance of angular momentum equations.
Note, this model is different from ones that we have studies in class. Instead of modeling the lateral dynamics, we instead model the slip angle dynamics by using the approximation $\beta \approx \frac{v_y}{v_x}$. The state space representation for the system dynamics is given below:

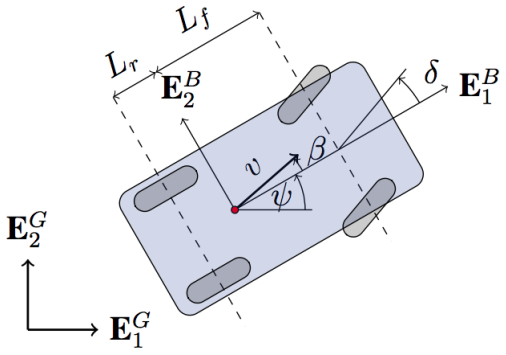


Figure 1: Dynamic Bicycle Model

$$\dot{\beta} = \frac{1}{mv_x}(F_{yF} + F_{yR}) - r \tag{1a}$$

$$\dot{r} = \frac{1}{I_z}(l_f F_{yF} - l_r F_{yR}) \tag{1b}$$

$$\dot{v}_x = \frac{1}{m}(F_{xR} - F_{yF}\sin\delta) + v_x r\beta \tag{1c}$$

with states $z = \begin{bmatrix}\beta, & v_x, & r\end{bmatrix}^T$ and inputs $u = \begin{bmatrix}\delta, & F_{xR}\end{bmatrix}^T$, where

$$
\begin{aligned}
m &= \text{vehicle mass} \\
\beta &= \text{sideslip angle} \\
v_x &= \text{longitudinal velocity} \\
r = \dot{\psi} &= \text{yaw rate} \\
\delta &= \text{steering angle} \\
F_{xR} &= \text{rear force} \\
F_{yR} &= \text{lateral force at lumped rear tire} \\
l_r &= \text{distance from the CoM to the rear axle} \\
l_f &= \text{distance from the CoM to the front axle}
\end{aligned}
$$

We use the Pacejka tire model to describe the lateral force of both the front and rear tires:

$$F_y(\alpha) = D \sin\left(C \tan^{-1}\left(B\alpha_f\right)\right) \tag{2}$$

where the front and rear tire side slip angles α are given by

$$\alpha_F = \tan^{-1}\left(\beta + \frac{l_f r}{v_x}\right) - \delta \tag{3}$$

$$\alpha_R = \tan^{-1}\left(\beta - \frac{l_r r}{v_x}\right) \tag{4}$$

Drift is a steady-state maneuver characterized by saturation of the rear tire forces, high side slip angle β, and often counter-steering. When a vehicle drifts, it looks like it is moving sideways or laterally, which corresponds to high side-slip angles β.

To analyze drift, we analyze the value of the states and inputs at steady state. Steady state means that the system states are not changing:

$$f(x^{\mathsf{eq}}, u^{\mathsf{eq}}) = \mathbf{0} \tag{5}$$

The system state and input at steady state are the equilibrium values of the system. The steady state definition in (5) does not mean that our physical system is at rest, but that the state and input do not change, (i.e. the slip angle, velocity and yaw rate may be non-zero, but they do not change over time). From the steady-state definition, we to set each of the dynamics equations in (1) to zero, and then determine the state and input, x^{eq} and u^{eq} respectively, that makes the equality hold.

Note that we have five unknowns, three from the state and two from the input, but we only have three equations from our system model in (1). Our problem is undetermined, so we need to fix exactly two values from our set of unknowns so that the number of unknown variables and number of equations are equal. We will fix the values of the longitudinal velocity v_x and steering angle δ, and grid the steering angle to observe how the steady state values change with steering angle.

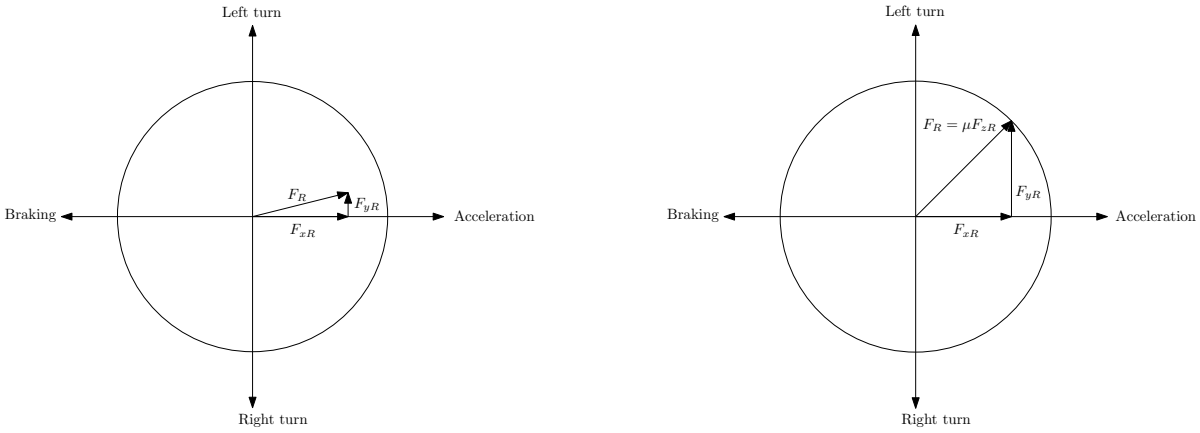
We define the following parameter values for our system, which are based on the first-generation BARC

vehicle:

$$
\begin{aligned}
m &= 1.98\text{kg} \\
I_z &= 0.24 \text{ [kg m}^2\text{]} \\
l_r &= 0.125 \text{ [m]} \\
l_f &= 0.125 \text{ [m]} \\
v_x &= 2 \text{ [m/s}^2\text{]} \\
B &= 7 \\
C &= 1.2 \\
D &= -\mu F_{zR} = -\frac{\mu m g}{2} = -2.2726 \\
\mu &= 0.234 \\
g &= 9.81 \text{ [m/s}^2\text{]}
\end{aligned}
$$

where F_{zR} is the normal force on the rear tire, μ is the coefficient of friction, and g is the acceleration due to gravity. For convenience, we lump the terms into the parameter D. As a note, the moment of inertia was estimated doing a torsional pendulum experiment and the coefficients for the Pajecka model were estimated by conducting cornering tests and then using nonlinear least squares optimization to fit the parameters to the data[1].

Recall that drift is characterized by rear tire saturation. This means that the longitudinal and lateral components of the rear tire use all the force available from the friction circle. Look at Fig. 2 to compare the rear tire force during typical cornering and during drifting.



(a) Cornering (b) Drifting

Figure 2: During typical cornering, the total rear tire force F_R uses only a portion of the available friction from the friction circle, while during drifting, the rear tire force uses all the available friction

During drifting, the vehicle uses all the available lateral force from the friction circle, which means we impose the following constraint on the to model drift

$$F_{yR} = \sqrt{(\mu F_{zR})^2 - F_{xR}^2} \tag{6}$$

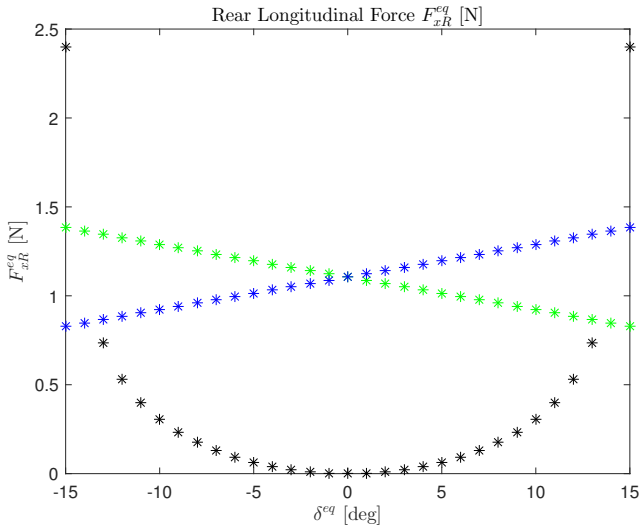
[1]For more details on inertia estimation, refer to https://www.mathworks.com/company/newsletters/articles/improving-mass-moment-of-inertia-measurements.html, and for details on the tire model estimation, refer to the paper "Autonomous Drifting with Onboard Sensors" In 13th International Symposium on Advanced Vehicle Control, AVEC16., September, 2016.

Note that during tire saturation, the lateral force can be applied to either the left or the the right of the vehicle.

The task now is to generate plots of how the equilibrium side slip angle β^{eq}, yaw rate r^{eq}, and longitudinal force F_{xR}^{eq} vary given a fixed longitudinal velocity v_x^{eq}, and a grid ed steering angle δ^{eq}.

We have provided a file called `eq_analysis_3s_barc_hw.m` to get you started. You can download it **here**. The program uses symbolic variables with the *syms* command to construct algebraic expressions for each of the quantities in our system model, and then feeds the dynamic equations into the function *vpasolve*, which numerically solves a set of nonlinear equations. The solution is stored and later plotted.

*****Deliverable** Submit all three plots for the equilibrium values of β^{eq}, r^{eq}, and F_{xR}^{eq} vary given a fixed longitudinal velocity v_x^{eq}. Your graph for the equilibrium longitudinal force against equilibrium steering angle should look approximately as follows:



Task 2 Sampling-based Parameter Estimation - No Deliverables

Drift parking consists of a series of precisely timed, but simple input commands: turn the steering wheel, apply the handbrake, then apply the pedal brake. We can generalize the inputs as step commands characterized by a set of parameters for timing and magnitude. Figure 3 shows the step input commands for the sliding maneuver for a full-scale vehicle.

In this lab, you will apply a sampling-based method to achieve the drift parking maneuver. You will sample steering and motor input commands from a uniform probability distribution and apply those commands directly to the vehicle. Rather than sampling from the entire spectrum of possible input commands, we can guide the sampling process by observing what the expert does, devising some general guidelines about how he performs the maneuver, and then using that information to limit our sample space.

The BARC only has one braking function (i.e. no hand brake), but it is not a hard-brake, meaning the wheels do not lock. For this experiment, we will use the input profile in Figure 4 to get the BARC to slide.

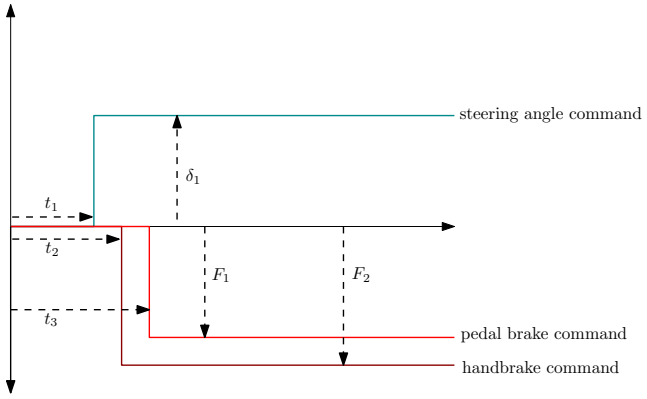


Figure 3: The input commands for steering angle, handbrake and pedal brake are parameterized by timing parameters t_1, t_2, t_3 and magnitudes δ_1, F_1, F_2.

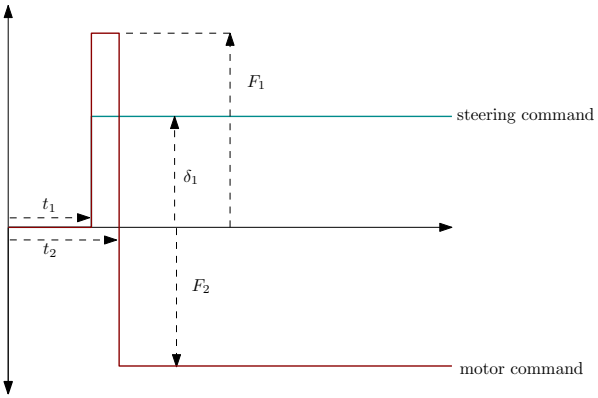


Figure 4: The input commands for steering angle, handbrake and pedal brake are parameterized by timing parameters t_1, t_2 and magnitudes δ_1, F_1, F_2.

More explicitly, the function profile for the force and steering commands we will use for the BARC are as follows:

$$\delta(t) = \begin{cases} 0 & t \leq t_1 \quad \text{go straight} \\ \delta_1 & t > t_1 \quad \text{turn} \end{cases} \tag{7a}$$

$$F_{xR}(t) = \begin{cases} 0 & t \leq t_1 \quad \text{go straight} \\ F_1 & t_1 < t \leq t_2 \quad \text{accelerate} \\ F_2 & t > t_2 \quad \text{brake} \end{cases} \tag{7b}$$

Before time t_1, we want the car to go straight. At time t_1, the car will turn and accelerate, and then at time t_2 the car will brake.
For this lab, you will sample three parameters : the timing command $\Delta t = t_2 - t_1$, the steering angle δ_1 and the acceleration command F_1. Each of the parameters Δt, δ_1 and F_1 will come from a uniform distribution $U(a, b)$, where a is the lower bound and b is the upper bound. We provide the braking command F_2. The parameter t_1 represents the moment the sliding maneuver starts, which we will not sample. We will specify t_1 as the moment we have travelled a certain, predefined distance.

For the drift maneuver, we break down the motion into two segments. In the first segment, you will command

your BARC to drive straight from rest to a target velocity using feedback from the encoders and the IMU. In the second segment, you will apply the sampled drift maneuver from equation 7 in open loop (no feedback control). Figure 5 illustrates how the maneuver should look in from a top view. The data in Figure 5 was generated from a drift experiment with the RC and an indoor gps kit[2]. The velocity measurements and slip angle estimate from the experiment are shown in Figure 6a and 6b, respectively. Note the the slip angle $\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$ becomes very high toward the end of the maneuver since the RC travels almost entirely laterally.

The colored blocks in Figure 5 show the trajectory of the vehicle over time, with the initial position around $(x_0, y_0) = (0, -1.5)$. The purple blocks indicate when the BARC is driving straight, and the yellow blocks indicate when the BARC is applying the sampled drift maneuver input.

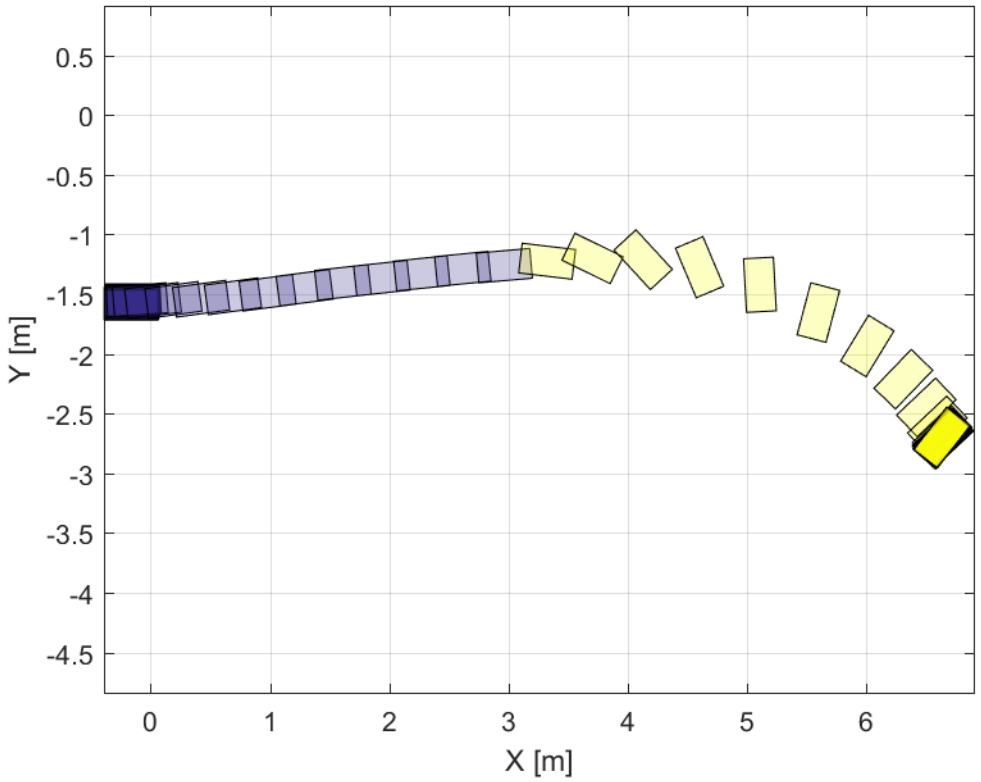
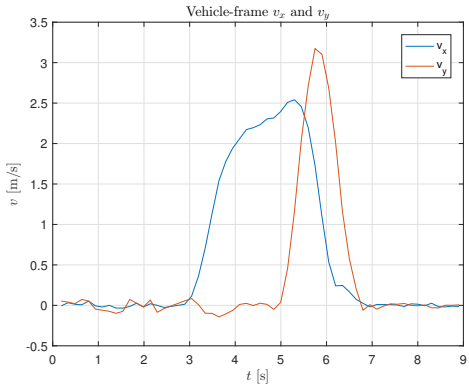


Figure 5: Purple blocks indicate the RC is driving straight, and the yellow blocks indicate the RC is applying the sampled input.

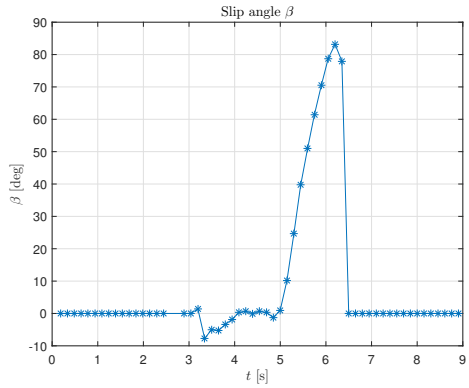
Task 3 Controller

For this task, you will implement a drift manuever on your BARC. Note that you will need at least 8 meters by 4 meters of open space to run this experiment.

[2]Marvelmind Robotics Indoor GPS kit : https://marvelmind.com/



(a) The longitudinal velocity v_x and lateral velocity v_y



(b) The side slip angle $\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right)$ is positive since the RC moves in the positive y direction in its frame (i.e. $+y$-axis in the vehicle frame)

Figure 6: The side slip angle v_y and β rise as the RC enters the drift maneuver, and then drops sharply when the RC comes to a stop

You will work with two files for this lab. You will use the launch file `lab8.launch` to tune PI parameters to drive straight until you have reached a target longitudinal velocity, and the file `driftController.py` to implement the sampled-based drift maneuver from Equation 7.

1. Ensure you have the most up-to-date files by updating your `lab8` package with the following steps:
   - On the Odroid, move into your `barc` directory with `cd barc`
   - `git fetch upstream`
   - `git checkout upstream/master -- workspace/src/labs/src/lab8`
   - `git checkout upstream/master -- workspace/src/labs/launch`
   - `git checkout upstream/master -- scripts`
   - `git checkout upstream/master -- arduino/arduino_nano328_node`
2. We need to update the software running on the Arduino. In a new terminal, run the command `flash_nano`
   This compiles the latest code and flashes the software onto the Arduino.

3. Now, you will tune two different PI controllers. One PI controller will control the motor (i.e. longitudinal force) to get the BARC to reach a target velocity of $v_{\text{ref}} = 4$[m/s] using feedback from the encoders, and the other PI controller will control the servo (i.e. steering angle) to get the BARC to drive straight using feedback from the IMU (more specifically, the yaw angle). We have provided a PID implementation in the file `pid.py` and have already given the usage of the controller inside `driftController.py`. We have also already provided simple estimators for the speed and the yaw angle, defined in the file `observers.py`, which use measurements from the encoders and IMU, respectively. You can either use the PI implementation and estimators we provide or write your own. Our implementation assumes all four encoders are working and that your IMU is orientated correctly.

   You will tune the PID parameters from the file *lab8.launch*. In `driftController.py`, the function *get_param* gets the parameters you set in the launch file. The file `driftController.py` implements both PI controllers for you, so you only need to edit the PI parameters inside the file `lab8.launch`.

You can use the cruise control PI values you tuned in Lab 5 to control the BARC to the reference velocity $v_{\text{ref}} = 4$[m/s]. You want to tune the PID to reach the target velocity within the first 3 meters. The file `driftController.py` sets the motor to neutral after traveling a distance $s = 3$ meters, where the distance is estimated by integrating speed over time. We also provide the estimate of distance for you. If you have more space for testing, you can increase the value of s in `driftController.py`.

To run the experiment, run the following command
`roslaunch labs lab8.launch id:="unique_experiment_id_here"`

4. ***Deliverable**: Tune both PI controllers to get BARC to drive in a straight line. Submit your tuned PI values for both controllers.

5. Write a sample-based drift maneuver for your vehicle, using the following strategy:

   - In the file `driftController.py`, sample the parameters Δt, δ_1 and F_1 from a uniform distribution $U(a, b)$, where a is the lower bound and b is the upper bound. You will use the function *uniform* to generate a sample for each parameter. We have shown usage of the function *uniform* for the case of the parameter Δt. We suggest the following bounds for each parameter, but you can expand or narrow the range as you observe how the BARC slides with each experiment
     $a_{\Delta t} = 0$, $b_{\Delta t} = 0.8$ (duration of acceleration)
     $a_{\delta_1} = 1850$, $b_{\delta_1} = 1900$ (aggressive turn range)
     $a_{F_1} = 1850$, $b_{F_1} = 1900$ (aggressive acceleration range)
   - Implement the sampled control input profile. We have provided commented code in the main loop of the `driftPlanner.py` file to help you implement the sampled input. You can un-comment the code and add the remaining logic, or you can write your own implementation. Now drive the BARC forward from rest using the velocity and steering PI controllers. After travelling a distance $s = 3$ [m], the RC should apply the sampled input profile in open loop.
   - Run the experiment multiple times and observe the motion of vehicle, noting the values of the sampled parameters Δt, δ_1 and F_1 each time. If the vehicle slides in a manner similar to the trajectory in Figure 5, stop and record the values of the sampled parameters. We have also uploaded a short video clip to show how the vehicle should slide. You can view it **here**

6. ***Deliverable** Fix the parameters Δt, δ_1 and F_1 (don't sample) to the ones from the successful trial, re-run the experiment and use an external camera to record your BARC's drift maneuver. Also submit the values of Δt, δ_1 and F_1 that you used.
   Note: you should observe that the sliding motion is repeatable given fixed parameters and the same initial state (position and orientation). The BARC may not slide into exactly the same spot because we are relying on integration of a noisy speed estimate for distance, but for highly repeatable trajectories, we would need to accurate position feedback from a GPS or a camera sensor.

HINTS:

- For this lab, it is very important that all your encoders are working and that the IMU is oriented correctly.
- It will be easier for your BARC to enter a slide if you decrease the coefficient of friction between the tires and the floor. You can try conducting the experiment on a smooth surface or applying tape to the wheels in order to make sliding easier.