Low Level Controller Documentation

MPC Lab
Enrico Cruvinel
12/06/2021

1 Overview

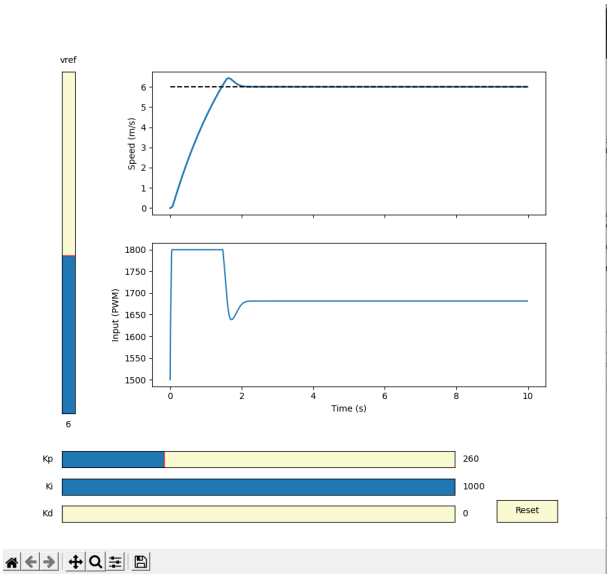
This document is a guide to implementing BARC's lower level controller. First, the data from the car needs to be collected. That data is then processed in a system identification script for lateral and longitudinal motion, which will output specific system parameters. Those parameters should be changed in the model dynamics portion of the controller tuning file. Once that is done, the controller tuning program can be run from which the controller gains will be obtained. The controller gains are then modified in the firmware file, which should be flashed to the Arduino.

2 System Identification

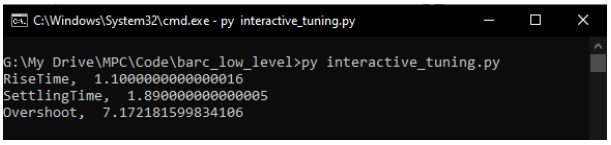
First, we need to obtain the system parameters of our model. Files pertaining to system identification are in the non-planar repository. Data should be collected for longitudinal and lateral motion with different acceleration and steering commands. In the end, the user will have files in .npz format. The files corresponding to longitudinal motion should be added to `/nonplanar_vehicle_navigation/sys_id/longitudinal`; likewise, files corresponding to lateral data should be added to `/nonplanar_vehicle_navigation/sys_id/lateral`. The "files" variable in the first lines of the process_id_data.py file should be modified accordingly to match the data files names. After that variable is modified, process_id_data.py should be run. The program will correlate PWM input and acceleration. The output of the longitudinal process id file should be, respectively, the delay, offset, gain, rolling resistance (roll_res), damping, drag, and two saturation coefficients (sat_poly_3 and sat_poly_5). The output of the lateral process id file should be, respectively, delay, offset, gain, outer gain, lr and lf.

3 Controller Tuning

The files for the controller tuning are in the BARC Low Level repository. There are two options for controller tuning, one is the automated (auto_tuning.py), the other one is manual(interactive_tuning.py). In either case, the vehicle_config variable needs to be changed to reflect the system parameters obtained in the previous step. Once that is changed run either one of the tuning scripts. The automatic tuning script will iterate over different controller gains and output the gains that minimize a cost function composed of rise time, settling time and overshoot. The interactive tuning script will display the system response and controller output for given controller gains and reference value. System characteristics such as overshoot, rise time, and settling time are printed on the terminal. The sliders can be used to change the gain values until the desired response is achieved.



(a) System Response and Controller Output



(b) Output System Characteristics

Figure 1: Interactive tuning

4 Hardware Implementation

The hardware files can be found at the BARC Hardware Interface repository. The file in `/barc_hardware_interface/arduino/BARC_NODE_V*/BARC_NODE_V*.ino` should be modified. Specifically, the PID controller variable declaration gains should correspond to the tuned gains from the previous step. After that is done, verify compile the code and flash it to the Arduino. Once that is done, the controller implementation is complete.

5 Summary

1. Relevant repositories: Non-planar, BARC Low Level, and BARC Hardware Interface
2. Run experiments to obtain lateral and longitudinal data files
3. Add those files to the respective sys_id folder and modify the "files" variable in process_id_data.py
4. Obtain system parameters and assign them to the vehicle_config variable in either the auto_tuning.py or interactive_tuning.py files
5. Run either tuning file and obtain the controller gains
6. Open the `firmware/BARC.ino` file and change the gain values in the PID declaration
7. Flash the code into the Arduino