Project 3: Optimal Control

24-677 Special Topics: Linear Control Systems

Prof. M. Bedillion

Due: April 29, 2021, 11:59 pm.

- We will use Gradescope to grade. The link is on the panel of CANVAS.
- Submit **your_controller.py** to Gradescope under **Programming-P3** and your solutions in **.pdf** format to **Project-P3**. Insert the performance plot image in the **.pdf**. We will test **your_controller.py** and manually check all answers.
- We will make extensive use of Webots, an open-source robotics simulation software, for this project. Webots is available here for Windows, Mac, and Linux.
- For Python usage with Webots, please see the Webots page on Python. Note that you may have to reinstall libraries like `numpy, matplotlib, scipy,` etc. for the environment you use Webots in.
- Please familiarize yourself with Webots documentation, specifically their User Guide and their Webots for Automobiles section, if you encounter difficulties in setup or use. It will help to have a good understanding of the underlying tools that will be used in this assignment. To that end, completing at least Tutorial 1 in the user guide is highly recommended.
- If you have issues with Webots that are beyond the scope of the documentation (e.g. the software runs too slow, crashes, or has other odd behavior), please let us know via Piazza. We will do our best to help.
- We advise you to start with the assignment early.

1 Introduction

In this project, you will complete the following goals:

1. Design an lateral optimal controller
2. Consider the relative merits of static LQR control vs. model predictive control for controlling the nonlinear system

[Remember to submit the write-up, plots, and codes on Gradescope.]

2 Model

We will use the same model from Project 2, which is repeated here for clarity. The error-based linearized state-space for the lateral dynamics is as follows.
e_1 is the distance to the center of gravity of the vehicle from the reference trajectory.
e_2 is the orientation error of the vehicle with respect to the reference trajectory.

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4C_\alpha}{m\dot{x}} & \frac{4C_\alpha}{m} & -\frac{2C_\alpha(l_f-l_r)}{m\dot{x}} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_\alpha(l_f-l_r)}{I_z\dot{x}} & \frac{2C_\alpha(l_f-l_r)}{I_z} & -\frac{2C_\alpha(l_f^2+l_r^2)}{I_z\dot{x}} \end{bmatrix}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{2C_\alpha}{m} & 0 \\ 0 & 0 \\ \frac{2C_\alpha l_f}{I_z} & 0 \end{bmatrix}\begin{bmatrix} \delta \\ F \end{bmatrix} + \begin{bmatrix} o \\ -\frac{2C_\alpha(l_f-l_r)}{m\dot{x}} - \dot{x} \\ 0 \\ -\frac{2C_\alpha(l_f^2+l_r^2)}{I_z\dot{x}} \end{bmatrix}\dot{\psi}_{des}$$

In lateral vehicle dynamics, $\dot{\psi}_{des}$ is a time-varying disturbance in the state space equation. Its value is proportional to the longitudinal speed when the radius of the road is constant. When deriving the error-based state space model for controller design, $\dot{\psi}_{des}$ can be safely assumed to be zero.

$$\frac{d}{dt}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4C_\alpha}{m\dot{x}} & \frac{4C_\alpha}{m} & -\frac{2C_\alpha(l_f-l_r)}{m\dot{x}} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2C_\alpha(l_f-l_r)}{I_z\dot{x}} & \frac{2C_\alpha(l_f-l_r)}{I_z} & -\frac{2C_\alpha(l_f^2+l_r^2)}{I_z\dot{x}} \end{bmatrix}\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{2C_\alpha}{m} & 0 \\ 0 & 0 \\ \frac{2C_\alpha l_f}{I_z} & 0 \end{bmatrix}\begin{bmatrix} \delta \\ F \end{bmatrix}$$

For the longitudinal control:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}\begin{bmatrix} \delta \\ F \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\psi}\dot{y} - fg \end{bmatrix}$$

Assuming $\dot{\psi} = 0$:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}\begin{bmatrix} \delta \\ F \end{bmatrix}$$

3 Controller Design Approaches

We will look at two different approaches to study the system (of course in addition to these you can come up with your own to maximize performance!)

- Static LQR control. Linearize the system about the nominal operating speed and design an infinite horizon LQR controller for the system. Apply this LQR controller throughout the trajectory.
- Model predictive control. We know that the system will likely slow down around curves. We can re-linearize the system in terms of the actual operating speed at each time step and solve a discrete time LQR controller n steps into the future, then output only the 1st control learned in the sequence.

3.1 Static LQR Control

Start by discretizing the continuous error dynamics at the given time step (the ZOH discretization is probably best). Now we will design the infinite horizon LQR controller. Using LQR requires manually creating two matrices Q and R. Q works to penalize state variables, while R penalizes control input. Try to think about what form your Q and R matrices should take for good performance.

- For Q, large values will heavily restrict changes to the respective states, while small values will allow the states to easily change.
- Similarly, in R, large values will heavily restrict control input, while small values will allow the control input to vary widely.
- One idea for tuning is to set the relevant indices of Q and R to

$$\frac{1}{(\textbf{max value of the corresponding state/input})^2}$$

  in order to normalize the value. Make sure to experiment outside of this guideline to determine the best performance.
- There is a relationship between Q and R, though it is subtle. For example, if you increase weights in Q, you are more heavily penalizing changes in the states, which will require more control input. This would imply that you should decrease weights in R to see an effect. Due to this, it may be helpful to keep either Q or R fixed while varying the other during tuning.
- Diagonal versions of Q and R are often used because they are trivially positive definite and easy to interpret in terms of the state variables.

3.2 Model Predictive Control

The LQR controller assumes that the speed is constant - can we remove this constraint? Consider the following procedure at each time step.

- Using the measured car speed, compute a new linearized model of the (discrete time) system.
- Compute a finite horizon LQR controller N steps into the future.
- Output the first element of the control sequence.

As an alternative, of course, we could re-linearize the system and recompute an infinite horizon LQR. This recomputation could be triggered this when the speed has moved sufficiently far away from the operating point. However, there might be advantages to having an approach that is consistent between all time steps (avoids abrupt changes in control / computation time).

4 Project 3: Problems [Due April 29th, 2021]

Exercise 1. Like Project 2 we will practice LQR design on an unrelated system before jumping into the control of the Sprinter. You should solve all problems computationally using Matlab.

1. For the system
$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -10 & -7 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$
find the control $u(t)$ that minimizes the performance measure
$$J = 10x_1^2(5) + \frac{1}{2}\int_0^5 5x_1^2 + x_2^2 + 0.25u^2 dt.$$
Plot the control sequence starting from $x(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

2. Discretize the above system using a step size of $T = 0.005$ and find a control that minimizes the performance measure
$$J = 10x_1^2(1000) + \frac{1}{2}\sum_{k=1}^{999} 5x_1(k)^2 + x_2^2(k) + 0.25u^2(k).$$
Plot the control sequence starting from $x(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and compare to the continuous time result.

3. Design an infinite time LQR controller for the continuous time system using the cost function
$$J = \frac{1}{2}\int_0^\infty 5x_1^2 + x_2^2 + 0.25u^2 dt.$$
Plot the control sequence up to 5 seconds starting from $x(0) = \begin{bmatrix}1\\1\end{bmatrix}$ and compare to the above results.

Exercise 2. Show your approach for designing the two controllers discussed in the prior section. Specifically, show the following.

1. Show your approach to tuning the Q and R matrices along with the Matlab code you used to design your controllers. Provide your implementation code in Python as well, along with a graph that shows your best performance.
2. Show the approach you used to determine N for the model predictive controller. This should include simulation results as a function of N while holding Q and R constant. Performance should converge as N increases. Provide your implementation code in Python, along with a graph that shows your best performance.

Exercise 3. Optimize your Webots performance for both controller types..

You can reuse your longitudinal PID controller from part 1 of this project, or even improve upon it. However, it may require retuning based on observed performance.

Design the two controllers in `your_controller.py`. You can make use of Webots' built-in code editor, or use your own.

When you complete the track, the scripts will generate a performance plot via `matplotlib`. This plot contains a visualization of the car's trajectory, and also shows the variation of states with respect to time.

Submit `your_controller_lqr.py` and `your_controller_mpc.py` and the final completion plots as described on the title page. Both controllers are **required** to achieve the following performance criteria to receive full points:

1. Time to complete the loop = 200 s
2. Maximum deviation from the reference trajectory = 6.5 m
3. Average deviation from the reference trajectory = 2.5 m
4. **(NEW!)** Average rate of change in steering angle = 0.025 rad/timestep

[10% Bonus]: Complete the loop within 130 s. The maximum deviation and the average deviation should be within in the allowable performance criteria mentioned above.

5 Appendix

(Already covered in P1)

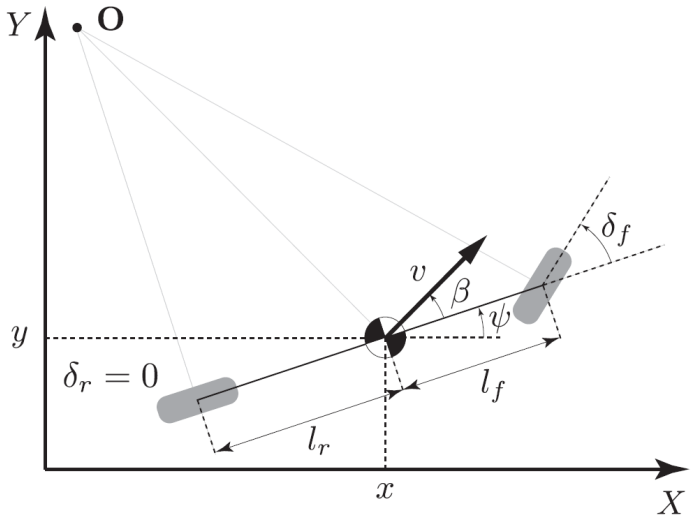

Figure 1: Bicycle model[2]

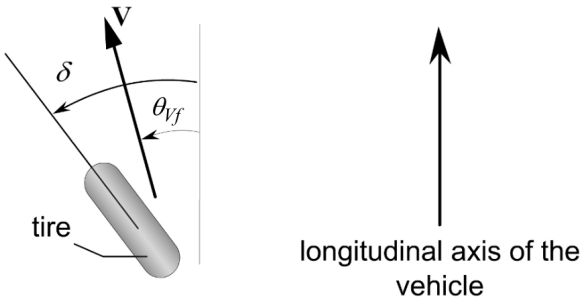

Figure 2: Tire slip-angle[2]

We will make use of a bicycle model for the vehicle, which is a popular model in the study of vehicle dynamics. Shown in Figure 1, the car is modeled as a two-wheel vehicle with two degrees of freedom, described separately in longitudinal and lateral dynamics. The model parameters are defined in Table 2.

5.1 Lateral dynamics

Ignoring road bank angle and applying Newton's second law of motion along the y-axis:

$$ma_y = F_{yf} \cos \delta_f + F_{yr}$$

where $a_y = \left(\frac{d^2y}{dt^2}\right)_{inertial}$ is the inertial acceleration of the vehicle at the center of geometry in the direction of the y axis, F_{yf} and F_{yr} are the lateral tire forces of the front and rear

wheels, respectively, and δ_f is the front wheel angle, which will be denoted as δ later. Two terms contribute to a_y: the acceleration $\ddot{y}$, which is due to motion along the y-axis, and the centripetal acceleration. Hence:

$$a_y = \ddot{y} + \dot{\psi}\dot{x}$$

Combining the two equations, the equation for the lateral translational motion of the vehicle is obtained as:

$$\ddot{y} = -\dot{\psi}\dot{x} + \frac{1}{m}(F_{yf}\cos\delta + F_{yr})$$

Moment balance about the axis yields the equation for the yaw dynamics as

$$\ddot{\psi}I_z = l_f F_{yf} - l_r F_{yr}$$

The next step is to model the lateral tire forces F_{yf} and F_{yr}. Experimental results show that the lateral tire force of a tire is proportional to the "slip-angle" for small slip-angles when vehicle's speed is large enough - i.e. when $\dot{x} \geq 0.5$ m/s. The slip angle of a tire is defined as the angle between the orientation of the tire and the orientation of the velocity vector of the vehicle. The slip angle of the front and rear wheel is

$$\alpha_f = \delta - \theta_{Vf}$$
$$\alpha_r = -\theta_{Vr}$$

where θ_{Vp} is the angle between the velocity vector and the longitudinal axis of the vehicle, for $p \in \{f, r\}$. A linear approximation of the tire forces are given by

$$F_{yf} = 2C_\alpha \left(\delta - \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right)$$
$$F_{yr} = 2C_\alpha \left(-\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}\right)$$

where C_α is called the cornering stiffness of the tires. If $\dot{x} < 0.5$ m/s, we just set F_{yf} and F_{yr} both to zeros.

5.2 Longitudinal dynamics

Similarly, a force balance along the vehicle longitudinal axis yields:

$$\ddot{x} = \dot{\psi}\dot{y} + a_x$$
$$ma_x = F - F_f$$
$$F_f = fmg$$

where F is the total tire force along the x-axis, and F_f is the force due to rolling resistance at the tires, and f is the friction coefficient.

5.3 Global coordinates

In the global frame we have:

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi$$
$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi$$

5.4 System equation

Gathering all of the equations, if $\dot{x} \geq 0.5$ m/s, we have:

$$\ddot{y} = -\dot{\psi}\dot{x} + \frac{2C_\alpha}{m}(\cos\delta\left(\delta - \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right) - \frac{\dot{y} - l_r\dot{\psi}}{\dot{x}})$$
$$\ddot{x} = \dot{\psi}\dot{y} + \frac{1}{m}(F - fmg)$$
$$\ddot{\psi} = \frac{2l_fC_\alpha}{I_z}\left(\delta - \frac{\dot{y} + l_f\dot{\psi}}{\dot{x}}\right) - \frac{2l_rC_\alpha}{I_z}\left(-\frac{\dot{y} - l_r\dot{\psi}}{\dot{x}}\right)$$
$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi$$
$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi$$

otherwise, since the lateral tire forces are zeros, we only consider the longitudinal model.

5.5 Measurements

The observable states are:

$$y = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ X \\ Y \\ \psi \end{bmatrix}$$

5.6 Physical constraints

The system satisfies the constraints that:

$$|\delta| \leqslant \tfrac{\pi}{6}\ rad$$
$$F \geqslant 0 \text{ and } F \leqslant 16000\ N$$
$$\dot{x} \geqslant 10^{-5}\ m/s$$

Table 1: Model parameters.

Name	Description	Unit	Value
$(\dot{x}, \dot{y})$	Vehicle's velocity along the direction of vehicle frame	m/s	State
(X, Y)	Vehicle's coordinates in the world frame	m	State
ψ, $\dot{\psi}$	Body yaw angle, angular speed	rad, rad/s	State
δ or δ_f	Front wheel angle	rad	State
F	Total input force	N	Input
m	Vehicle mass	kg	4500
l_r	Length from rear tire to the center of mass	m	3.32
l_f	Length from front tire to the center of mass	m	1.01
C_α	Cornering stiffness of each tire	N	20000
I_z	Yaw intertia	kg m^2	29526.2
F_{pq}	Tire force, $p \in \{x, y\}$,$q \in \{f, r\}$	N	Depends on input force
f	Rolling resistance coefficient	N/A	0.028
`delT`	Simulation timestep	sec	0.032

5.7 Simulation

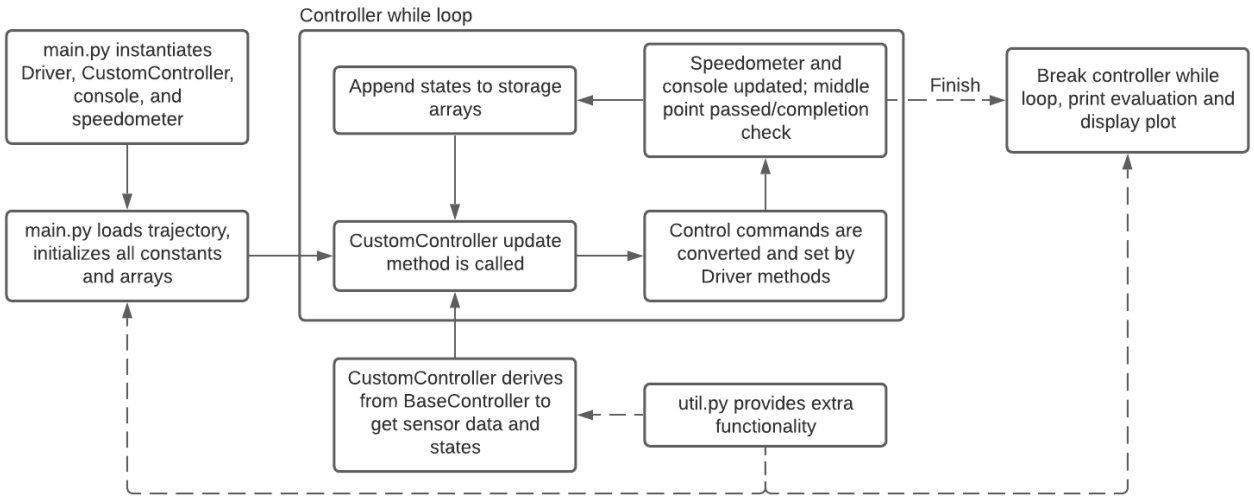

Figure 3: Simulation code flow

Several files are provided to you within the `controllers/main` folder. The `main.py` script initializes and instantiates necessary objects, and also contains the controller loop. This loop runs once each simulation timestep. `main.py` calls `your_controller.py`'s `update` method

on each loop to get new control commands (the desired steering angle, δ, and longitudinal force, F). The longitudinal force is converted to a throttle input, and then both control commands are set by Webots internal functions. The additional script `util.py` contains functions to help you design and execute the controller. The full codeflow is pictured in Figure 3.

Please design your controller in the `your_controller.py` file provided for the project part you're working on. Specifically, you should be writing code in the `update` method. Please **do not** attempt to change code in other functions or files, as we will only grade the relevant `your_controller.py` for the programming portion. However, you are free to add to the `CustomController` class's `__init__` method (which is executed once when the `CustomController` object is instantiated).

5.8 BaseController Background

The `CustomController` class within each `your_controller.py` file derives from the `BaseController` class in the `base_controller.py` file. The vehicle itself is equipped with a Webots-generated GPS, gyroscope, and compass that have no noise or error. These sensors are started in the `BaseController` class, and are used to derive the various states of the vehicle. An explanation on the derivation of each can be found in the table below.

Table 2: State Derivation.

Name	Explanation
(X, Y)	From GPS readings
$(\dot{x}, \dot{y})$	From the derivative of GPS readings
ψ	From the compass readings
$\dot{\psi}$	From the gyroscope readings

5.9 Trajectory Data

The trajectory is given in `buggyTrace.csv`. It contains the coordinates of the trajectory as (x, y). The satellite map of the track is shown in Figure 4.

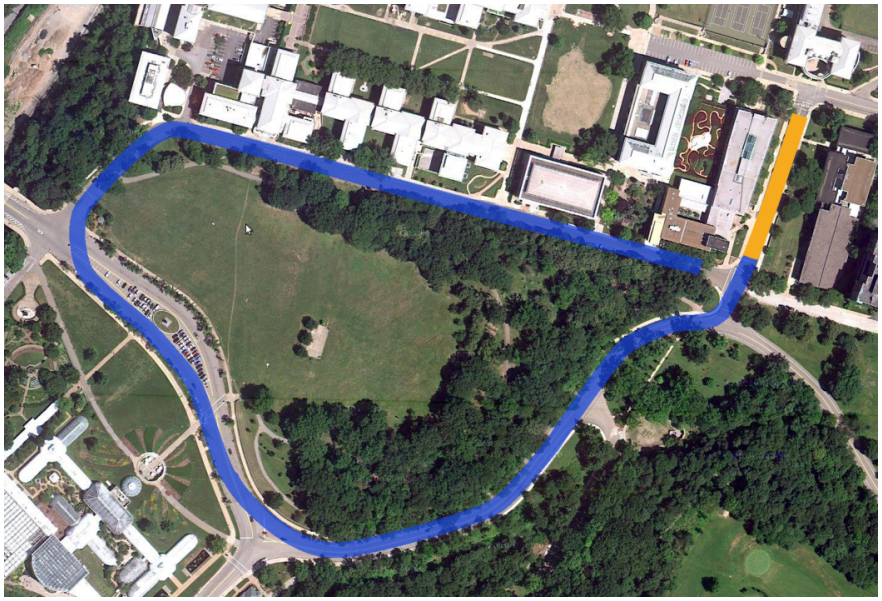

Figure 4: Buggy track[3]

6 Reference

1. Rajamani Rajesh. Vehicle Dynamics and Control. Springer Science & Business Media, 2011.

2. Kong Jason, et al. "Kinematic and dynamic vehicle models for autonomous driving control design." Intelligent Vehicles Symposium, 2015.

3. cmubuggy.org, https://cmubuggy.org/reference/File:Course_hill1.png

4. "PID Controller - Manual Tuning." *Wikipedia*, Wikimedia Foundation, August 30th, 2020. https://en.wikipedia.org/wiki/PID_controller#Manual_tuning