Project: Part 2

24-677 Special Topics: Modern Control - Theory and Design

Prof. D. Zhao

Due: Nov 9, 2021, 11:59 pm. Submit within the deadline.

- Your online version and its timestamp will be used for assessment.
- We will use Gradescope to grade. The link is on the panel of CANVAS. If you are confused about the tool, post your questions on Campuswire.
- Submit **your_controller.py** to Gradescope under **P2-code** and your solutions in **.pdf** format to **P2-writeup**. 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 the TAs know via Campuswire. We will do our best to help.
- We advise you to start with the assignment early. All the submissions are to be done before the respective deadlines of each assignment. For information about the late days and scale of your Final Grade, refer to the Syllabus in Canvas.

1 Introduction

In this part of the project, you will complete the following two assignments:

1. Check the controllability and stabilizability of the linearized system
2. Design a lateral full-state feedback controller

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

2 Model

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. psi error

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

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 P2: Problems

Exercise 1. Considering the linearized, error-based state space system for the vehicle in the Model section above:

1. Check the controllability and observability of the system at the following longitudinal velocities: 2 m/s, 5 m/s and 8 m/s.

2. For longitudinal velocities v from 1 m/s to 40 m/s, plot the following:

   (a) $log_{10}(\frac{\sigma_1}{\sigma_n})$ versus v (m/s), where σ_i is the ith singular value of the controllability matrix P $(i = 1, 2, ..., n)$. (In other words, what is the logarithm of the greatest singular value divided by the smallest?)

   (b) $Re(p_i)$ versus v (m/s), where Re is real part and p_i is the ith pole of the continuous state space system. [Use 4 subplots, one for each of the 4 poles]

   What conclusions can you draw about the overall controllability and stability of the system in observing these two plots?

[Submit you answers in the **.pdf** file and also submit the Python script. The Python script should be named **Q1.py**]

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

$$\begin{bmatrix} e_1 \\ \dot{e}_1 \\ e_2 \\ \dot{e}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{4\times 20000}{1888.6\times 2} & \frac{4\times 20000}{1888.6} & \frac{-2\times 20000(1.55-1.39)}{1888.6\times 2} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2\times 20000(1.55-1.39)}{25845\times 2} & \frac{2\times 20000(1.55-1.39)}{25845} & -\frac{2\times 20000(1.55^2-1.39^2)}{25854\times 2} \end{bmatrix}$$

1. $\dot{x} = 2m/s$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -21.2 & 42.36 & -1.69 \\ 0 & 0 & 0 & 1 \\ 0 & -0.124 & 0.248 & -0.364 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 21.18 & 0 \\ 0 & 0 \\ 2.40 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P = [B \ AB \ A^2B \ A^3B]$$

$$P = \begin{bmatrix} 0 & 0 & 21.18 & 0 & -453.07 & 0 & 9712.7 & 0 \\ 21.18 & 0 & -453.07 & 0 & 9712.7 & 0 & -206155 & 0 \\ 0 & 0 & 2.4 & 0 & -3.50 & 0 & 58 & 0 \\ 2.40 & 0 & -3.50 & 0 & 58.05 & 0 & -122.6 & 0 \end{bmatrix}$$

$r(P) = 4$ Controllable

$Q = \begin{bmatrix} C \\ CA \\ CA^2 \\ CA^3 \end{bmatrix}$ will be printed in python console.

$r(Q) = 4$ Observable

$\dot{x} = 5m/s$

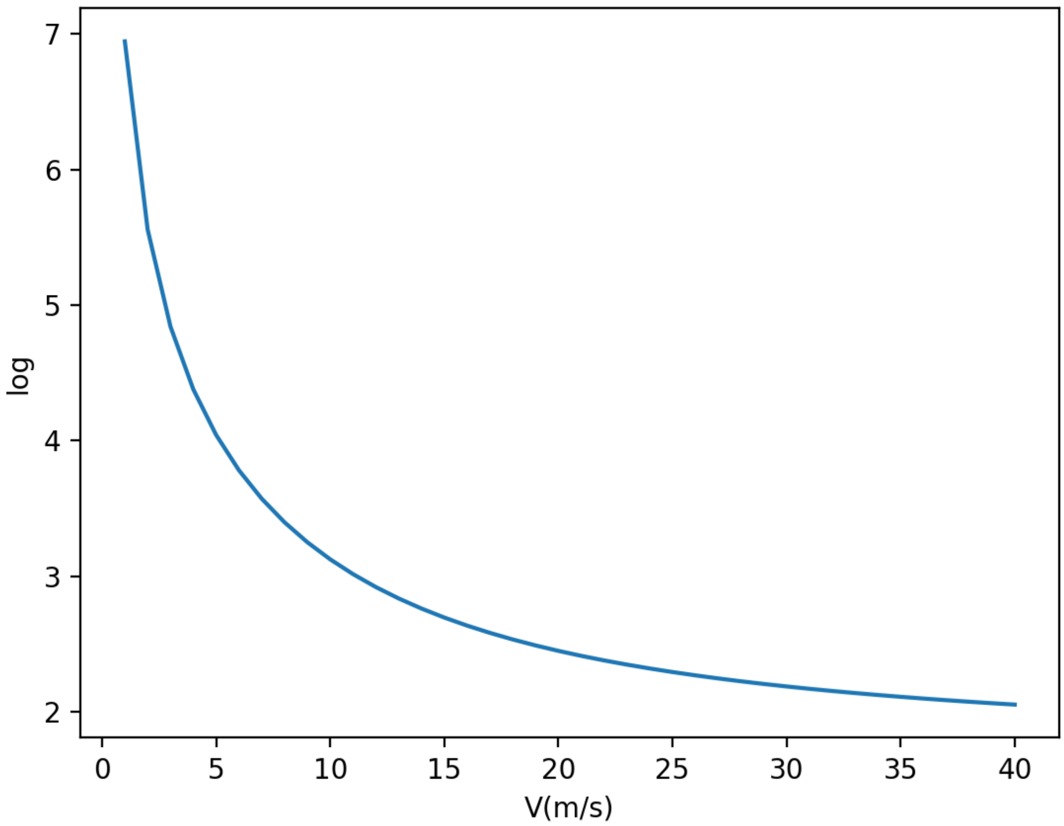
$r(P) = 4$ Controllable

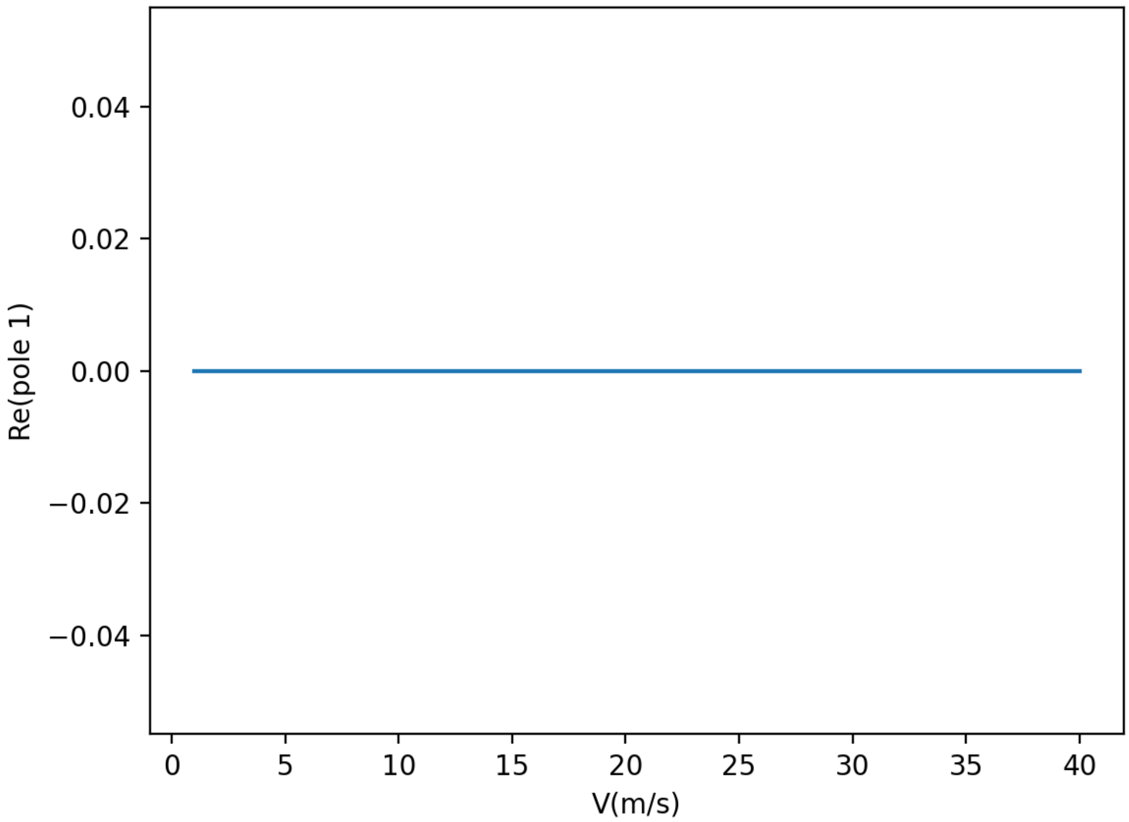
$r(Q) = 4$ Observable

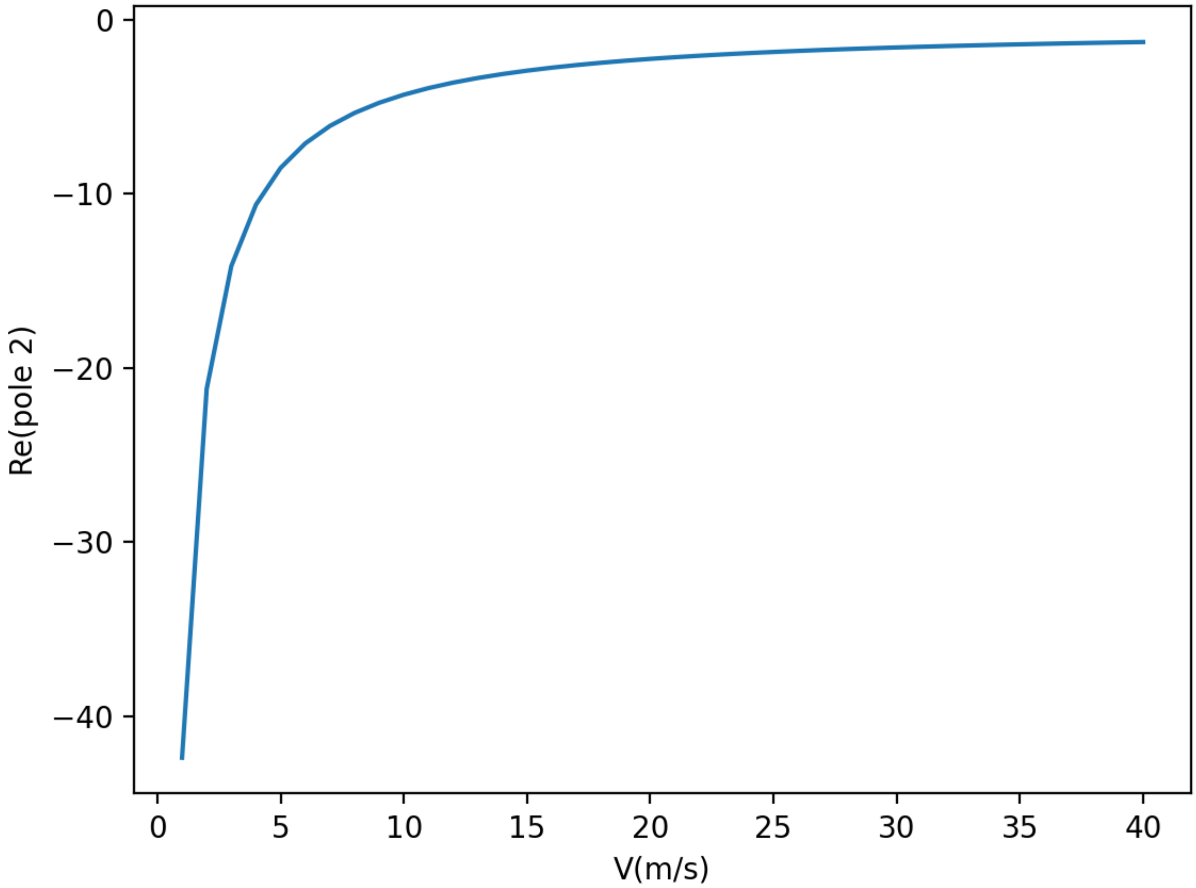
$\dot{x} = 8m/s$ $r(P) = 4$ Controllable

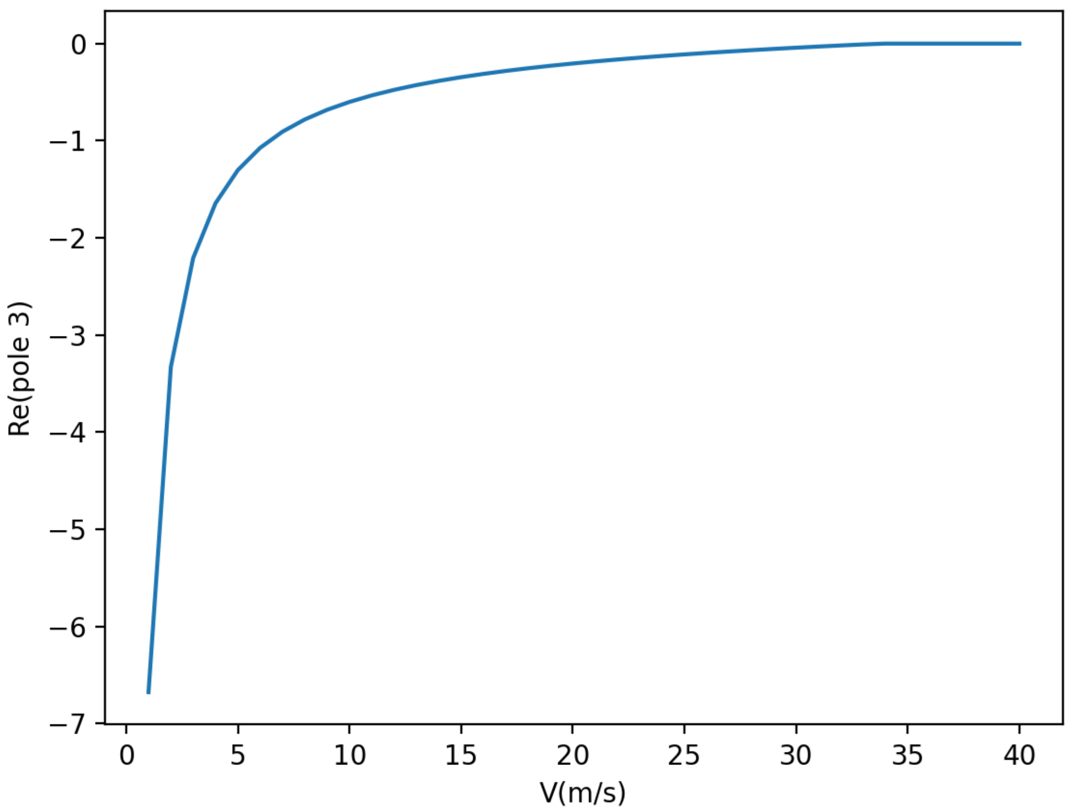
$r(Q) = 4$ Observable

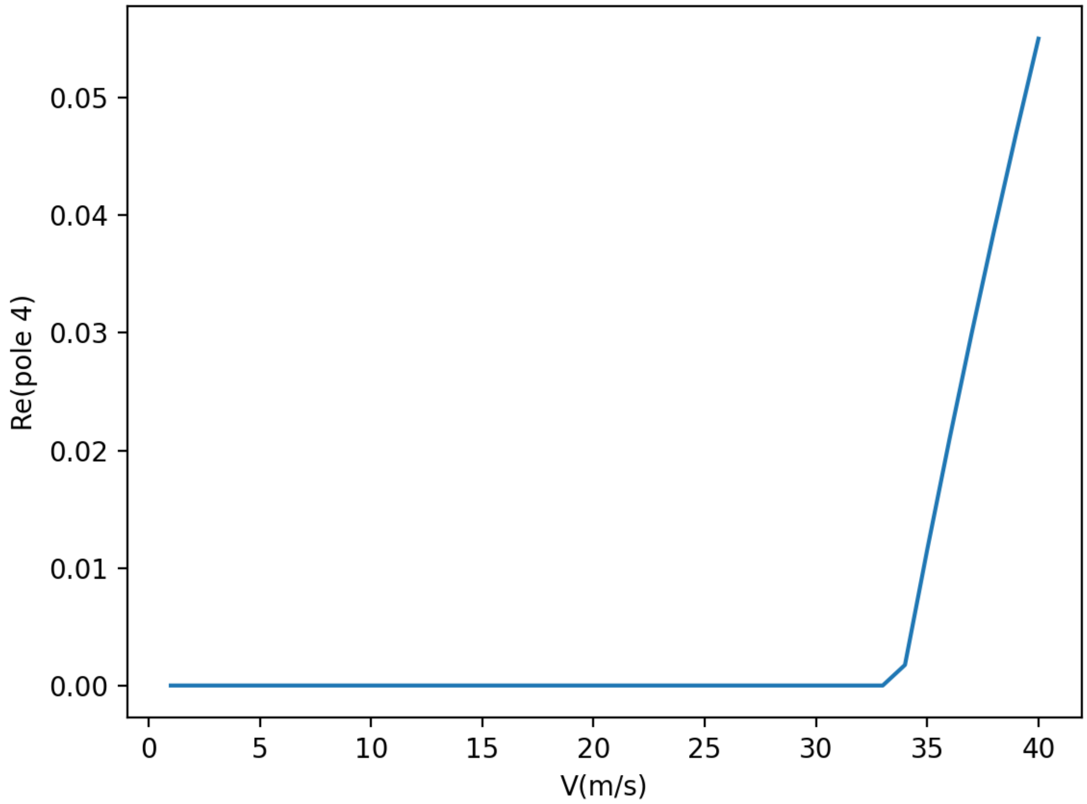
2.(b) From plot 1, the logarithm of maximum singular value over minimum singular value of controllibility matrix is approaching zero, which indicates that system is becoming more and more controllable.
For the 4 poles plot, the real part of poles are approaching zero which means the system is getting more and more unstable.


7
6
5
4
3
2
log
0
5
10
15
20
25
30
35
40
V(m/s)

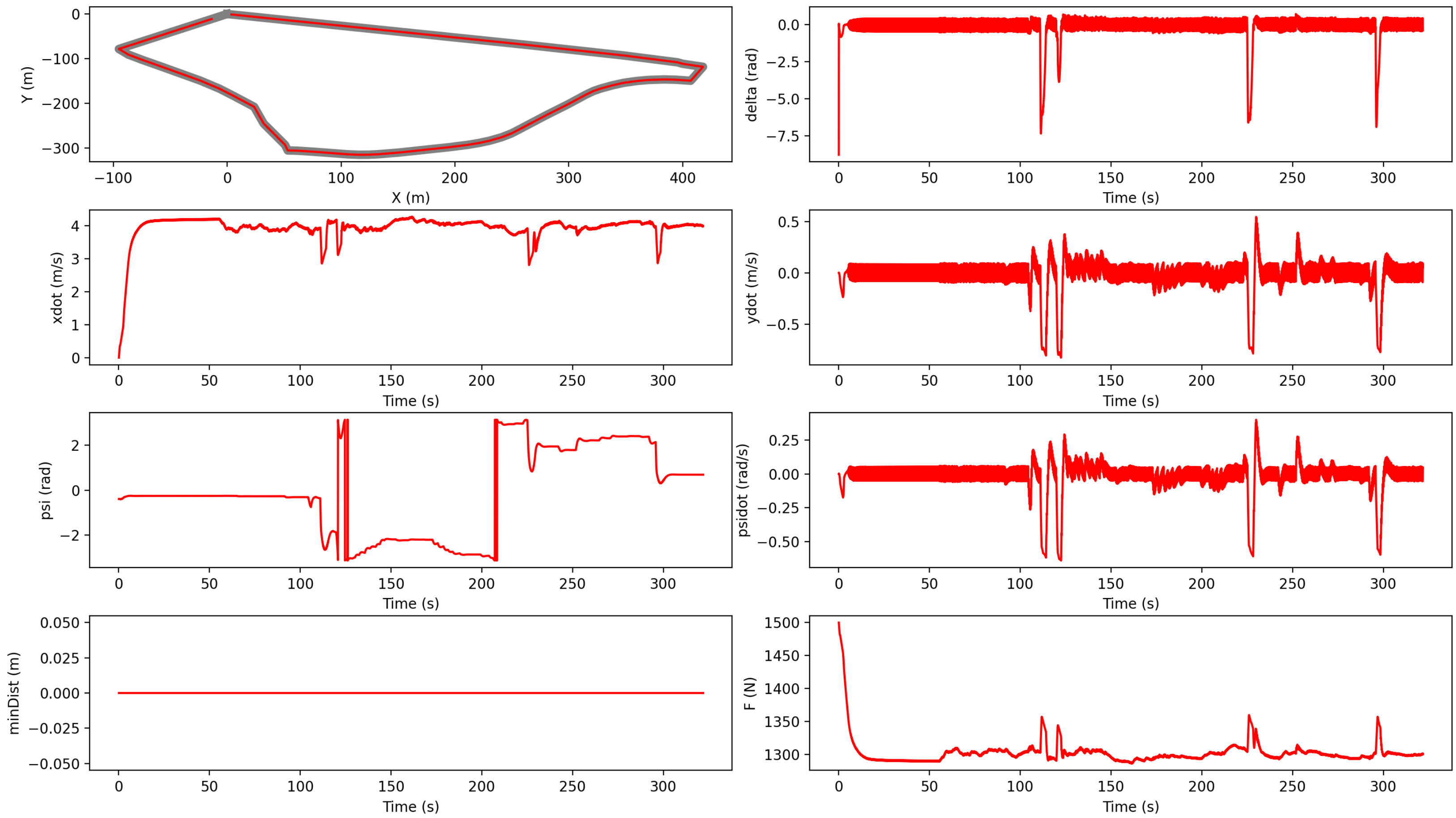

0.04
0.02
0.00
−0.02
−0.04
Re(pole 1)
0
5
10
15
20
25
30
35
40
V(m/s)


0
−10
−20
−30
−40
Re(pole 2)
0
5
10
15
20
25
30
35
40
V(m/s)


0
−1
−2
−3
−4
−5
−6
−7
Re(pole 3)
0
5
10
15
20
25
30
35
40
V(m/s)



Re(pole 4)
V(m/s)



Y (m)
X (m)
delta (rad)
xdot (m/s)
ydot (m/s)
psi (rad)
psidot (rad/s)
minDist (m)
F (N)
Time (s)

Exercise 2. For the lateral control of the vehicle, design a state feedback controller using pole placement. Tune the poles of the closed loop system such that it can achieve the performance criteria mentioned below.

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.

Check the performance of your controller by running the Webots simulation. You can press the play button in the top menu to start the simulation in real-time, the fast-forward button to run the simulation as quickly as possible, and the triple fast-forward to run the simulation without rendering (any of these options is acceptable, and the faster options may be better for quick tests). If 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.py` and the final completion plot as described on the title page. Your controller is **required** to achieve the following performance criteria to receive full points:

1. Time to complete the loop = 350 s
2. Maximum deviation from the reference trajectory = 9.0 m
3. Average deviation from the reference trajectory = 4.5 m

Some hints that may be useful:

- The `signal` subpackage within scipy is required for this part. Please investigate which functions you will need to use. The main goal is to calculate a gain matrix K such that $-Kx = u$, where x is the states and u is the control input.
- It is somewhat difficult to tune pole-placement controllers. Learning optimal control in the next submodule will fortunately make this task much easier. Some tips to help for this assignment follow.
  - Poles must be negative if the system is stable.
  - Poles can be complex, where an imaginary number is denoted with `j`, e.g. `-3+1j`. If you use a complex pole, you must also include its complex conjugate.
  - Don't use the poles from Exercise 1 as a starting point - these are the system's open-loop poles. Your goal is to select new positions for the closed-loop poles.
  - Poles placed closer to the imaginary axis (in other words, closer to 0 on the real axis) will dominate the system response. These poles allow the system to converge quickly.

  - The further poles are placed from the imaginary axis, the less influence they have. The same is true for poles which are further from the real axis. If all poles are fairly distant, the system will have a slow response.
  - Having at least one dominant pole to help the system to converge is recommended. The placement of your other poles is up to you based on your performance. Alternatively, you can also place a pair of conjugate poles close to the imaginary axis, and keep the other two away from it.

- The controller itself can be continuous or discrete - it is your choice whether to discretize the system or not.

4 Appendix

(Already covered in P1)

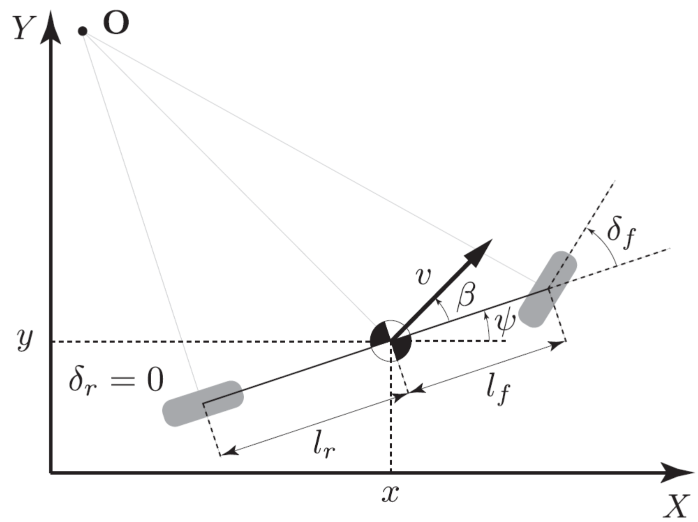


Figure 1: Bicycle model[2]

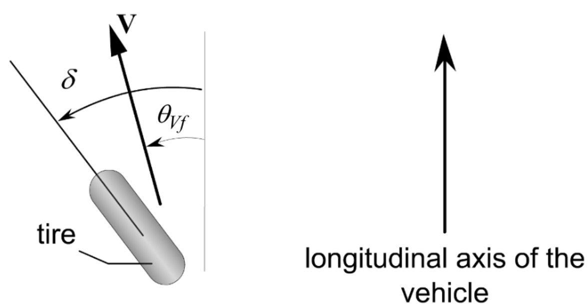


Figure 2: Tire slip-angle[2]

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.

4.1 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

$$\begin{aligned}\alpha_f &= \delta - \theta_{Vf}\\ \alpha_r &= -\theta_{Vr}\end{aligned}$$

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

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

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.

4.2 Longitudinal dynamics

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

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

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.

4.3 Global coordinates

In the global frame we have:

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

4.4 System equation

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

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

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

4.5 Measurements

The observable states are:

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

4.6 Physical constraints

The system satisfies the constraints that:

$$|\delta| \leqslant \tfrac{\pi}{6}\ rad$$
$$F \geqslant 0 \text{ and } F \leqslant 15736\ N$$
$$\dot{x} \geqslant 10^{-5}\ m/s$$

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	Input
F	Total input force	N	Input
m	Vehicle mass	kg	1888.6
l_r	Length from rear tire to the center of mass	m	1.39
l_f	Length from front tire to the center of mass	m	1.55
C_α	Cornering stiffness of each tire	N	20000
I_z	Yaw intertia	kg m^2	25854
F_{pq}	Tire force, $p \in \{x, y\}$,$q \in \{f, r\}$	N	Depends on input force
f	Rolling resistance coefficient	N/A	0.019
`delT`	Simulation timestep	sec	0.032

4.7 Simulation

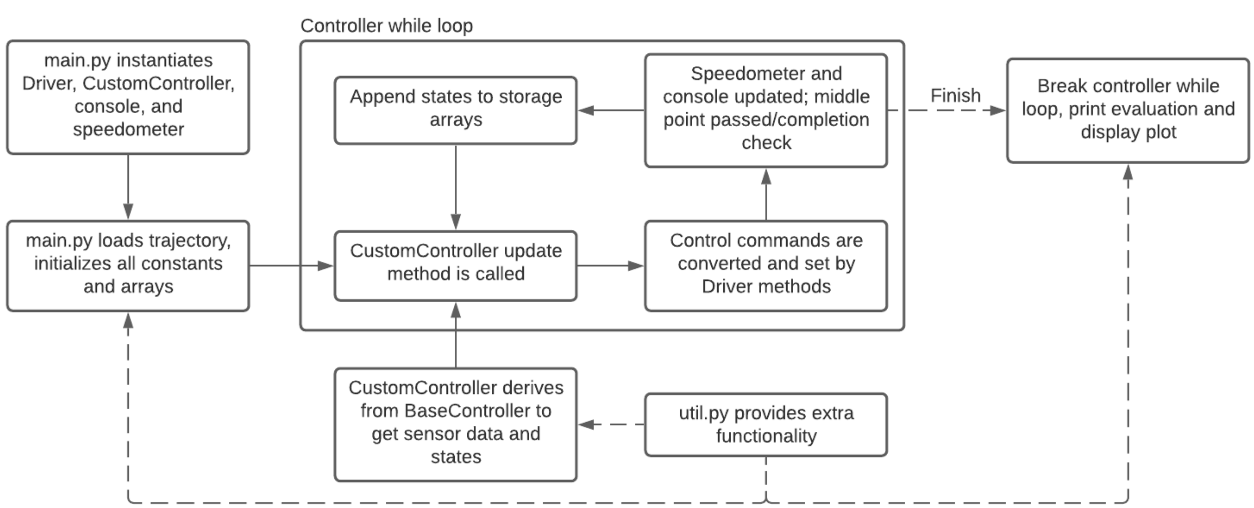


Figure 3: Simulation code flow

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).

4.8 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

4.9 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.

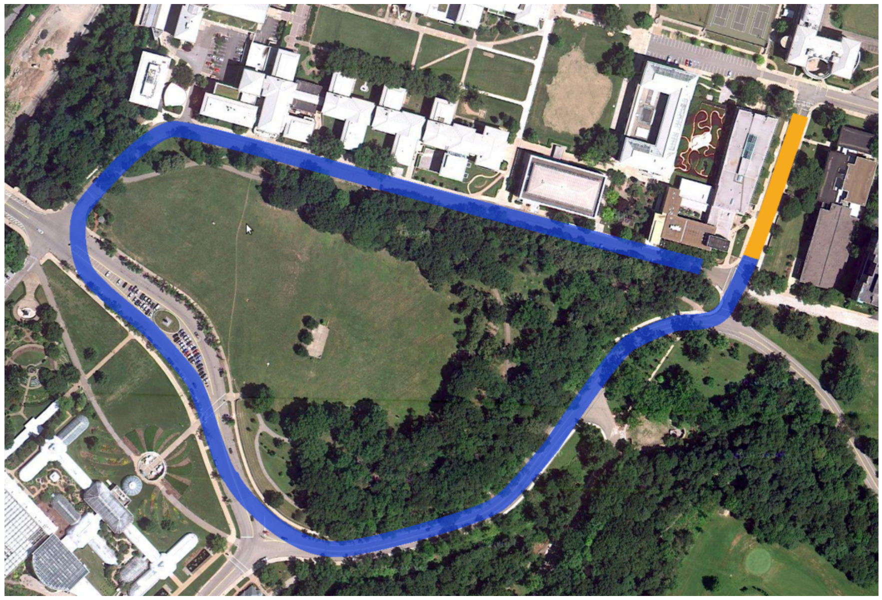


Figure 4: Buggy track[3]

5 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