ME227: Vehicle Dynamics and Control **Spring 2021**

Assignment 4: Lanekeeping

Due Thursday, April 29 at 5:00pm PST ©Chris Gerdes, 2017

Purpose

In this assignment, we will look at the characteristics of the simple lookahead controller we developed in class. We will also put together a simulation that will serve as our model for testing controllers prior to implementing them on Niki in your project teams.

Instructions

This and following homework assignments will be submitted using two different tools, Gradescope and MATLAB Grader. Throughout the assignment, each prompt will be marked with either **Gradescope** or **MATLAB Grader** to make it clear where you should be submitting the answer to that problem. MATLAB Grader questions will also be shown in **Blue** to make them easy to see.

All written portions must be turned in through Gradescope. See the Piazza post on homework guidelines for more instructions on the different homework resources available to you. Whatever format you decide to use, please $\boxed{\textbf{BOX}}$ all of your final answers.

Some problems will make use of the MATLAB Grader suite discussed in class. These problems are available directly from Canvas by clicking on each individual question. You are allowed to submit code to MATLAB Grader as many times as you want before the due date without penalty. In this way you can be sure each function or script passes all of the assesments that go along with it before moving on to the next problem. We encourage you to write your own test cases as well to ensure your code is working as expected.

When writing functions and running simulations, use the set of parameters given to you for Niki. These will be available in each MATLAB Grader problem where they are appropriate. Note that some problems will use different parameters than Niki's true values to demonstrate a different vehicle dynamics concept. We'll try to point out where that happens. The real values for Niki are given in Appendix A and B at the end of this document.

This week we will additionally ask you to use a new tool **MATLAB Online**. This tool will be useful for simulating your project code without some of the limitations of MATLAB Grader. If you have a normal desktop license for MATLAB, you are welcome to use that as well, it will effectively be the same tool. See our Piazza post for more information about how to setup MATLAB Online.

Problem 0 – Forming Project Teams (Canvas)

For the course project, we ask that you form groups of four (we currently have 52 enrolled students in the class). We have made 13 groups on Canvas for you to sign up. You must sign up for a group by the due date of this homework. You are free to sign up immediately, starting now. We will open up a functionality to search for groups on Piazza to help students coordinate to create groups. You can post whether you are looking for more group members or looking for a group.

You can find the groups in Canvas under "People". There will be a "Project Groups" tab there. Please form a group by the due date of this homework, April 29th at 5:00PM PST.

The full project description will be included in a separate document, but there are a few things we'd like you to keep in mind when choosing groups. We will ask you to implement both a lookahead controller as well as a controller of your choosing (PID, LQR, MPC, etc). When picking teammates, you should discuss potential ideas for this second controller. While we encourage you to use this second controller as a means to explore new or interesting control techniques, we are not grading you on its complexity (e.g. a non-linear model predictive controller will not necessarily be graded higher than a PID controller, simply because it's more complex). It might also help to post what availability you have to work, as some people are located in different timezones or are working part/full time.

Problem 1 – Understanding the Lookahead Controller

In this problem, we will analyze the lookahead controller and look at how the system poles change as we change parameters in the controller. We will use the form of the lookahead controller derived in class (which is slightly different than the version used in the kinematic model since we now consider tire stiffness)

$$\delta = \frac{-K_{la}\left(e + x_{la}\Delta\Psi\right)}{C_{af}}$$

With this controller, the system becomes

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \\ \Delta\Psi \\ \Delta\dot{\Psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K_{la}}{m} & -\frac{(C_{\alpha f}+C_{\alpha r})}{mU_x} & \frac{(C_{\alpha f}+C_{\alpha r})}{m} - \frac{K_{la}x_{la}}{m} & \frac{(-aC_{\alpha f}+bC_{\alpha r})}{mU_x} \\ 0 & 0 & 0 & 1 \\ -\frac{K_{la}a}{I_z} & \frac{(bC_{\alpha r}-aC_{\alpha f})}{I_zU_x} & \frac{(aC_{\alpha f}-bC_{\alpha r})}{I_z} - \frac{K_{la}ax_{la}}{I_z} & -\frac{\left(a^2C_{\alpha f}+b^2C_{\alpha r}\right)}{I_zU_x} \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \\ \Delta\Psi \\ \Delta\dot{\Psi} \end{bmatrix}$$

This system describes the closed loop response of the vehicle using the lookahead controller listed above. You should use this system of equations for all of the questions in **Problem 1**.

Question 1.A – Determining an Appropriate Lookahead Gain (MATLAB Grader)

Let's start with the "lookdown" controller ($x_{la} = 0$). Our first step is to determine a gain that seems physically reasonable.

Follow the prompts in MATLAB Grader to create a script to calculate K_{la} such that our controller produces a steer angle of 3 degrees when the lateral error is one meter and the vehicle is pointed straight along the path.

You are to modify the lookahead gain, `K_la`, to the appropriate value.

Question 1.B – Understanding our Controller Gain (Gradescope)

Imagine the controller developed in **Question 1.A** is deployed as a lane-keeping driver assistance system. Would it provide the driver a comfortable nudge to slowly guide them back into the lane, or more overt restoring forces to the lane-center. Explain your reasoning.

Describe how the controller would respond if it was deployed for driver assistance. Explain your reasoning.

Question 1.C – Varying Lookahead Gain (MATLAB Grader)

Let's look at how the lookahead gain impacts the poles of the system. Fix the speed U_x at 15 m/s and set the lookahead distance x_{la} to 10 m.

Follow the prompts in MATLAB Grader to create a script to plot the poles of the system for lookahead gains, K_{la}, from 1,000 N/m to 10,000 N/m, using increments of 1,000 N/m.

Question 1.D – Lookahead Gain Pole Position Analysis (Gradescope)

Examine the plots you created in **Question 1.C**. What are the trends in the natural frequency, ω_n, and damping ratio, ζ, of the **dominant pair of poles** as the lookahead gain is increased from 1,000 N/m to 10,000 N/m?

Describe the behavior of the dominant pair of poles in terms of ω_n and ζ as lookahead gain is increased.

Question 1.E – Simulating Different Lookahead Gains (MATLAB Grader)

Let's see how well the predictions from **Question 1.D** match the system's simulated response.

Follow the prompts in MATLAB Grader to create a script to simulate the system response using lookahead gains of 1,000 N/m and 10,000 N/m. Keep the lookahead distance at 10 m, and the speed at 15 m/s. Simulate the responses for 10 seconds each. Plot the lateral error as a function of time for each of the lookahead gains. Begin with an initial error of 1 m, the vehicle pointing straight along the path, and with $\dot{e} = 0$ and $\Delta\dot{\Psi} = 0$.

Because our system is in a convenient state-space form, we can use the built-in MATLAB function `lsim` to easily simulate our system. To do this we need to create a system object using `ss`. Because we only want to look at the behavior of e we can set the output matrix C to output e alone like this:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$$

You can read more about how to implement these functions in the MATLAB documentation here:

MATLAB lsim Documentation Page and *MATLAB ss Documentation Page.*

NOTE: You can copy your state matrix from ***Question 1.C*** *here and in subsequent problems where the system dynamics don't change. For some reason using Ctrl-C and Ctrl-V seems to work more consistently than right-clicking and selecting copy and paste.*

Question 1.F – Lookahead Gain Simulation Analysis (Gradescope)

Examine the plots you created in **Question 1.E**. Do the time responses differ in the manner the dominant poles would suggest from **Question 1.D**? Explain why or why not.

Describe the system response in the simulation relative to your predictions based on the pole positions. Explain your response.

Problem 2 – Lookahead Control of an Oversteering Car

In the previous problem we used a parameter set corresponding to our Volkswagen GTI, Niki. Niki is understeering in the linear region of handling. The lookahead controller also has some nice properties when used with an oversteering car. We'll explore those more in the problem.

For the following questions we will be using a hypothetical vehicle parameter set. All of the parameters will be the same, with the exception of the weight distribution, which will now be 30% of the weight on the front axle and 70% of the weight on the rear axle. This change has been implemented for you in the vehicle params struct on MATLAB Grader.

Question 2.A – Understeer Gradient and Critical Speed (MATLAB Grader)

Follow the prompts on MATLAB Grader to create a script to calculate the understeer gradient in units of rad/m/s^2 and the critical speed of the car with the parameters stated above.

Question 2.B – Varying Lookahead Distance with an Oversteering Car (MATLAB Grader)

Here we will examine the behavior of the poles of the system with an oversteering car and the lookahead controller relative to different lookahead distances. The state-space form will be the same here as in **Problem 1**, we just need to fill in the proper values for the new vehicle parameters.

Follow the prompts on MATLAB Grader to create a script to plot the pole locations of this system over a range of lookahead distances from 0 m to 30 m in increments of 2 m. Set U_x to 30 m/s and
K_{la} to 3,500 N/m.

Question 2.C – Oversteering Vehicle Pole Position Analysis (Gradescope)

Examine the plots you created in **Question 2.B**. For what range of lookahead distance is the system stable?

Give the range of lookahead distances over which the system is stable.

Question 2.D – Simulating an Oversteering Vehicle (MATLAB Grader)

Let's see how well the predictions from **Question 2.C** match the system's simulated response

Follow the prompts in MATLAB Grader to create a script to simulate the system response of the oversteering vehicle. Keep U_x at 30 m/s and K_{la} at 3,500 N/m. Set the lookahead distance x_{la} to 25 m. Simulate the system response for 10 seconds. Plot the lateral error and heading error on separate plots as a function of time. Begin with an initial error of 1 m, the vehicle pointing straight along the path, and with $\dot{e} = 0$ and $\Delta\dot{\Psi} = 0$.

Because we're interested in both lateral error and heading error in this problem, we need to change the output matrix C slightly, relative to **Problem 1**. To get both of these quantities, we'll use:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

in our state-space object.

Question 2.E – Oversteer Vehicle Simulation Analysis (Gradescope)

Examine the plots you created in **Question 2.D**. Has the lookahead controller sucessfully stabilized the vehicle with respect to the road?

Describe the stability of the system with this controller.

Problem 3 – Building a Nonlinear Simulation

In previous homeworks, we have built up a simulator in MATLAB Grader based on the vehicle and tire models we have covered in class. **This week we're going to expand on this simulator in MATLAB, rather than MATLAB Grader**. For these exercises and for the project, you can either use MATLAB Online (https://matlab.mathworks.com/) or MATLAB. **The teaching team believes there are benefits to downloading MATLAB to your computer, though all assignments and the project can be completed using MATLAB Online.**

Stanford has made MATLAB and related MATHWORKS products available to all students at no charge. To download MATLAB, go to the following site and make an account using your Stanford email address. This will permit you to download and install MATLAB.

`http://www.mathworks.com/`

If you choose to install MATLAB, MATLAB Installer will ask you which toolboxes you'd like to install. In the appendix of this assignment, you will find a list of required toolboxes from the teaching team along with addition useful information about the software. While you may install all toolboxes, you will find that the program files becomes extremely large if you decide to install all toolboxes.

Once you complete this problem, you will have a great testing and development tool for your project in the comings weeks.

To begin, we'll provide you with tested versions of the functions you've already written `Fy_fiala` and `slip_angles`, and `integrate_euler`. We will also provide you with a script to generate the vehicle parameters struct that you've used in previous homeworks, and some templates for the functions you need to implement. You'll find these on Canvas in the zip file `HW4_P3.zip`. If using MATLAB Online, you can upload this zip file directly to MATLAB Online and extract it there.

For the entirety of this problem, we're going to build up the most general simulator we can, meaning we're going to enforce fewer assumptions than we have previously. You should not use small angle assumptions, constant speed assumptions, or straight road/constant radius assumptions in these problems. Implement the full equations of motion. The functions provided to you are written with this standard in mind.

The first three problems will walk you through implementing the new pieces of the simulator. We have created MATLAB Grader prompts to go along with these problems to help you check if you have implemented each function correctly. We have left the answer template blank, so you can copy and paste your code directly from MATLAB Online into MATLAB Grader.

The next three problems will walk you through implementing a lookahead and speed controller and testing it in different simulation environments.

Question 3.A – Implementing A Nonlinear Bicycle Model (MATLAB Online / MATLAB Grader)

Using the code template provided, implement a function `nonlinear_bicycle_model` that takes in vehicle states and parameters and calculates the state derivatives for each of the states. You should implement all of the position and velocity states in the model for a total of 6 states. Do not use small angle assumptions.

Niki is a front wheel drive vehicle. This means any positive longitudinal force (drive force) will be applied only at the front tires. If the longitudinal force is negative (braking force), it will be split evenly between the front and rear axles. You need to implement this behavior in your `nonlinear_bicycle_model` function.

When you are finished writing this function in MATLAB Online, copy and paste your code in the corresponding MATLAB Grader prompt to validate your results. You can use the code in the ”Code to Call Your Function” section of the MATLAB Grader prompt to debug and develop your code in MATLAB Online.

Question 3.B – Simulate Step Function (MATLAB Online / MATLAB Grader)

The next function we need is `simulate_step`. This function will be very similar to the one written in Homework 2, except it will call the `nonlinear_bicycle_model` function, and will propagate all six vehicle states.

Use the provided code template to implement the `simulate_step` function.

When you are finished writing this function in MATLAB Online, copy and paste your code in the corresponding MATLAB Grader prompt to validate your results. You can use the code in the ”Code to Call Your Function” section of the MATLAB Grader prompt to debug and develop your code in MATLAB Online.

Question 3.C – Simulator Script (MATLAB Online / MATLAB Grader)

In this problem you will use your experience over the past few weeks to build your own simulator script from scratch. Start with the simulator template in the `ME227Simulator.m` file. Use the `simulate_step` function from question **Question 3.B** to propogate the vehicle states given your current states and command inputs. In the next few questions, we will implement feedback controllers to generate command inputs. For now, make your commanded lateral force 0 ($F_x = 0$ N) and steer angle -0.5 deg ($\delta = -0.5$ deg).

Run this simulation from 0 to 8 seconds with the straight road by uncommenting the correct road parameters and commenting out the others at the top of the code template. Start with initial conditions $e = 1$ m, $\Delta\Psi = 0$ rad, $s = 0$ m, $r = 0$ rad/s, $U_y = 0$ m/s, and $U_x = 10$ m/s.

Once you have everything running, copy and paste your script into MATLAB Grader to validate your results.

Important: Do not change the names of the vehicle states in the simulation template, we will use these to validate your results in MATLAB Grader.

Important: You must comment out the lines containing ” `clear; clc; close all` ***” and the line calling the ”*** `animate` ***” function or your solution will not pass the MATLAB Grader tests.***

Question 3.D – Implementing Active Lane Assist and Cruise Control (MATLAB Online / Gradescope)

In previous homeworks, we have assumed that the vehicle is moving at a constant speed. Let's implement a simple cruise control function in your simulation that generates longitudinal force to track a desired longitudinal velocity. Let

$$F_{x,total} = K_{long}(U_{x,des} - U_x)$$

so we have a simple proportional controller that will generate a force in response to a difference in the actual and desired longitudinal speeds. Tune the gain K_{long} to produce a 0.05 g longitudinal acceleration in response to a 1 m/s difference in the desired and actual speeds.

Additionally, let's implement an active lane assist that helps the driver remain close to a desired path. We'll use the same lookahead controller from **Problem 1** and **Problem 2** with a gain $K_{la} = 1750$ N/m and lookahead distance $x_{la} = 20$ m. Set your desired speed to $U_{x,des} = 31$ m/s in your simulator script.

Use the code template `me227_controller` to implement both the lateral and longitudinal controller. **Do not change the inputs or outputs of this function!** We will use this function when implementing your controllers both in simulation and on Niki. We have copied the vehicle parameter structure that you're used to into the function, so you can access those variables as you normally would.

Let's simulate Niki with these controllers without a driver correctly intervening to aid the active lane assist. Simulate Niki with these controllers implemented on the straight road from 0 to 20 seconds. Start with initial conditions $e = 1$ m, $\Delta\Psi = 0$ rad, $s = 0$ m, $r = 0$ rad/s, $U_y = 0$ m/s, and $U_x = 27$ m/s.

Create plots of all six of our system states (r, U_x, U_y, $\Delta\Psi$, s, and e), and submit them to gradescope.

NOTE: You can save figures from MATLAB Online as either image files (.png, .jpg, etc.) or as a .pdf. You can then add these directly to your submission if you're using word or Latex, or you can use a program like PDFShuffler or PDFSam to add these pages in after you have a .pdf to submit, or you can use the separate images upload option of Gradescope to upload these images and pictures of your other answers individually. If you are having trouble, please contact the teaching team so we can help out.

Submit plots of all vehicle states.

Question 3.E – Lane Keeping on a Highway (MATLAB Online / Gradescope)

Using the same controller parameters as **Question 3.D**, simulate the vehicle tracking a 980 m radius curve from 0 to 20 seconds, and then with the gently undulating highway for the same amount of time. Start with initial conditions $e = 1$ m, $\Delta\Psi = 0$ rad, $s = 0$ m, $r = 0$ rad/s, $U_y = 0$ m/s, and $U_x = 31 m/s$.

While this road curvature may seem very low, we'll discuss highway design in a future lecture and find that this is a possible value.

Create plots of all six of our system states (r, U_x, U_y, $\Delta\Psi$, s, and e), for each simulation (curved and undulating roads), and submit them to gradescope.

Is this controller able to track the lane effectively without driver intervention in both lane configurations? Explain your conclusion.

Submit plots of all vehicle states for 2 different simulations. Is this controller able to track the lane effectively without driver intervention in both lane configurations? Explain your

conclusion.

Question 3.F – Adding Feedforward Control (MATLAB Online / Gradescope)

Finally add feedforward control to your lookahead controller like we discussed in class. Your controller should now have the form

$$\delta = -\frac{K_{la}}{C_{\alpha f}} \left(e + x_{la} \Delta\Psi \right) + \delta_{ff}$$

where the feedforward term is

$$\delta_{ff} = \frac{K_{la} x_{la}}{C_{\alpha f}} \Delta\Psi_{ss} + \kappa \left(L + K U_x^2 \right)$$

and the steady-state yaw rate is

$$\Delta\Psi_{ss} = \kappa \left(\frac{m a U_x^2}{L C_{\alpha r}} - b \right)$$

Using the same controller parameters as **Question 3.D**, simulate the vehicle tracking the curved and gently undulating roads for 0 to 20 seconds. Start with initial conditions $e = 1$ m, $\Delta\Psi = 0$ rad, $s = 0$ m, $r = 0$ rad/s, $U_y = 0$ m/s, and $U_x = 31$ m/s.

Create plots of all six of our system states (r, U_x, U_y, $\Delta\Psi$, s, and e), for each simulation (curved and undulating roads), and submit them to gradescope.

How large of an impact on tracking performance did the feedforward control have?

Submit plots of all vehicle states for 2 different simulations. Describe the change in controller performance after adding feedforward control.

Appendix A – Vehicle Parameters

Variable Name	**Value**	**Units**	**Description**
`veh.m`	1926.2	kg	Mass (Includes 4 passengers)
`veh.Iz`	2763.49	$\mathrm{kg\,m^2}$	Yaw Moment of Inertia
`veh.a`	1.264	m	Distance from Center of Mass to Front Axle
`veh.b`	1.367	m	Distance from Center of Mass to Rear Axle
`veh.L`	2.631	m	Wheelbase
`veh.Wf`	9817.9	N	Static front axle weight
`veh.Wr`	9078.1	N	Static rear axle weight

Table 4.1: Vehicle Parameters and Values

Appendix B – Tire Parameters

Linear Tire Model

Variable Name	Value	Units	Description
`f_tire.ca_lin`	80,000	N/rad	Front Cornering Stiffness
`r_tire.ca_lin`	120,000	N/rad	Rear Cornering Stiffness

Table 4.2: Linear Tire Model Parameters

Fiala Tire Model

Variable Name	Value	Units	Description
`f_tire.cy`	110,000	N/rad	Front Cornering Stiffness
`f_tire.mu`	0.90	Unitless	Front Peak Coefficient of Friction
`f_tire.mu_s`	0.90	Unitless	Front Sliding Coefficient of Friction
`r_tire.cy`	180,000	N/rad	Rear Cornering Stiffness
`r_tire.mu`	0.94	Unitless	Rear Peak Coefficient of Friction
`r_tire.mu_s`	0.94	Unitless	Rear Sliding Coefficient of Friction

Table 4.3: Fiala Tire Model Parameters

Appendix C – MATLAB

MATLAB Background

MATLAB is widely used in the automotive industry and in engineering more generally. As a result of its broad application, MATHWORKS has developed various toolboxes for industry-specific needs. Given this, it's often not necessary to download all toolboxes that MATHWORKS offers and companies will usually purchase only the toolboxes that are needed by their engineers.

MATLAB Installation

To install MATLAB and any associated toolboxes, go to the MATHWORKS website (`www.mathworks.com`) and create an account using your Stanford email. This will give you access to the complete version of MATLAB. Follow the instructions on the MATHWORKS website to download the software.

MATLAB Version

Mathworks tries to give both backward and forward compatibility to MATLAB. This is not always true and is especially tricky with certain toolboxes. Downloading the latest version of MATLAB (2021a) should work for everything we're doing in this class. If you already have an older version of MATLAB installed on your computer, don't worry about upgrading to the latest version. Should any compatibility issues arise, the teaching team will handle them with you on a case-by-case basis.

MATLAB Toolboxes

We require downloading these toolboxes in order to use the framework we have developed for the project:
MATLAB 2021a
Control System Toolbox
DSP System Toolbox

Simulink and Stateflow will not be used in this class, but they are widely used in research and industry. You may find it helpful to download these now for future use. You can always return later to download additional toolboxes.