ME227: Final Design Report

Julianne Dones, Natalie Ferrante, Nefeli Ioannou, Aubrey Kingston
Team *Tokyo Drift*

May 21st, 2020

Project Goal

To develop a speed controller and two different steering controllers and evaluating the performance on simulators of varying fidelity.

Speed Control

“If everything seems under control, you're not going fast enough.”
— Mario Andretti

Speed Control

We update the simple proportional speed controller to take into account Drag, Rolling Resistance, and Road Grade

Previous Controller:

$$F_x = K_{longitudinal}(U_{x,\ desired} - U_x)$$

Speed Control

We update the simple proportional speed controller to take into account Drag, Rolling Resistance, and Road Grade

Previous Controller:

$$F_x = K_{longitudinal}(U_{x,\ desired} - U_x)$$

New Controller:

$$F_x = mA_{x,\ desired} + F_{rr} + F_{drag} + K_{longitudinal}(U_{x,\ desired} - U_x) + F_{grade}$$

Assume grade (and therefore F_{grade}) = 0 for now

Speed Control

We update the simple proportional speed controller to take into account Drag, Rolling Resistance, and Road Grade

New Controller:

$$F_x = mA_{x,\ desired} + F_{rr} + F_{drag} + K_{longitudinal}(U_{x,\ desired} - U_x) + F_{grade}$$

```
%-----------------------------------------------------------------------
%% Longitudinal Control Law
%-----------------------------------------------------------------------

frr = 0.015;
CdA = 0.594; % m^2
rho = 1.225;
F_drag = CdA * rho * (1/2) * Ux^2; % drag force
F_rr = frr * veh.m*g;

theta_r = 0; % angle of grade (could be an input from path)
F_grade = veh.m * g * sin(theta_r);

Fx = veh.m * Ax_des + F_rr + F_drag +  K_long*(Ux_des - Ux) + F_grade;
%-----------------------------------------------------------------------
```

Steering Controllers

“Speed has never killed anyone. Suddenly becoming stationary, that's what gets you.”

-**Jeremy Clarkson**

Lookahead with Feedforward

Implemented lookahead controller with feedforward term added:

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

with:

$$\Delta\Psi_{ss} = \kappa\left(\frac{maU_x^2}{LC_{\alpha r}} - b\right)$$

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

Lookahead with Feedforward

Process for Determining Kla, xla, and Klong

Kla -- Root Locus Analysis:

Found range of stable gains given *max (Ux) = 14m/s* from path profile (all gains found to be stable)

Kla -- Based on *comfortable g-load* for passengers [1]

Test with simulator to find Kla satisfies these constraints and tracking error guarantees (prioritized lateral acceleration)

Xla -- Root Locus Analysis:

Found range of stable gains given *max (Ux)* from path profile and chosen Kla (found Xla > 2m required)

Xla -- Based on keeping *lateral error <= 0.25m* and *actuator commands* considerations:

Test with simulator to find system satisfies lateral error constraint and minimizes large actuator command changes

Klong -- Based on the *jerk* in the system (change in acceleration):

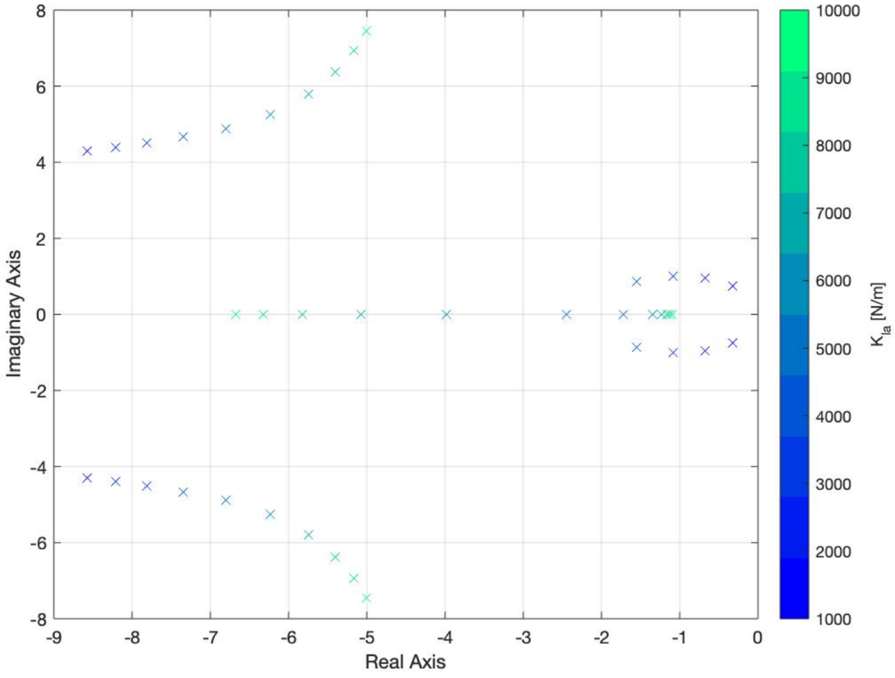
Test with simulator to minimize jerk in the system.

(ran all in sim_mode 3 for highest accuracy)

[1] https://repositories.lib.utexas.edu/bitstream/handle/2152/20856/cats_rr_40.pdf;sequence=2

Lookahead w/ Feedforward

The root locus of the system shown below with Kla = 1,000:10,000 shows that Kla is stable for all values in this range

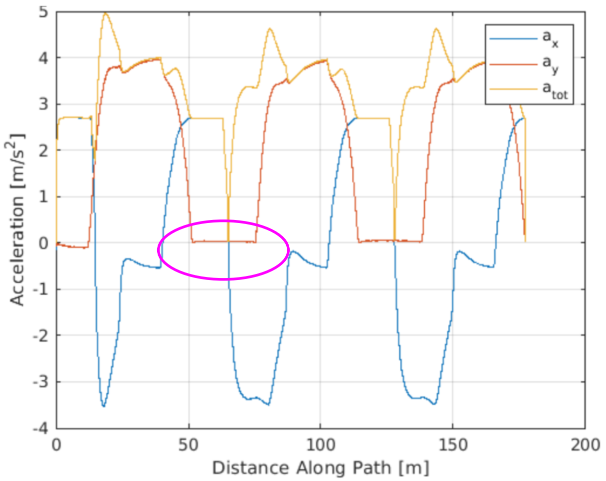


Lookahead w/ Feedforward

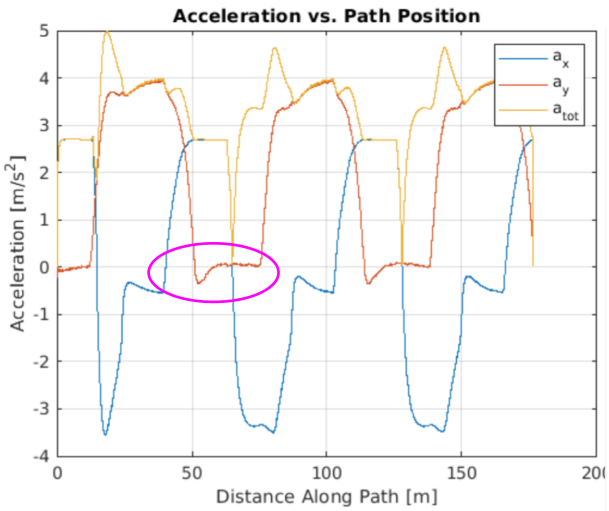
Simulation Results on more detailed environment (*sim mode = 3*) for several stable values of Kla (1000, 3500, 5000) and the values of xla and klong from problem set 4. Acceleration plots shown below.

xla = 15m

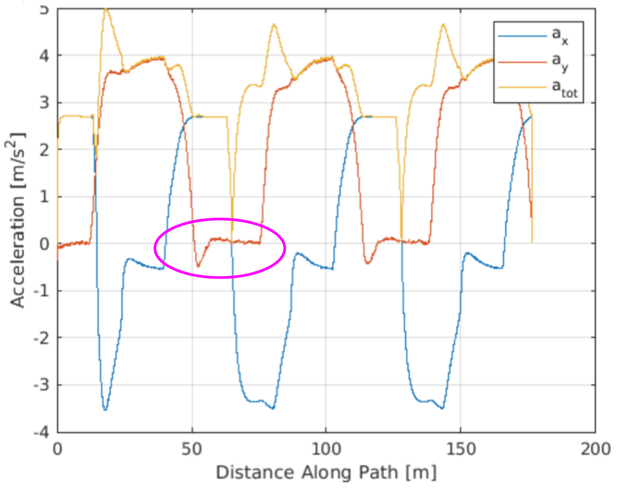
Klong = 0.1



Kla = 1000



Kla = 3500



Kla = 5000

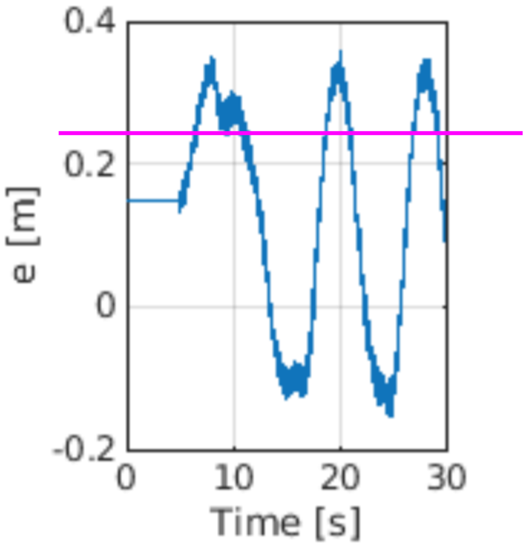
Lookahead w/ Feedforward

Lateral Error

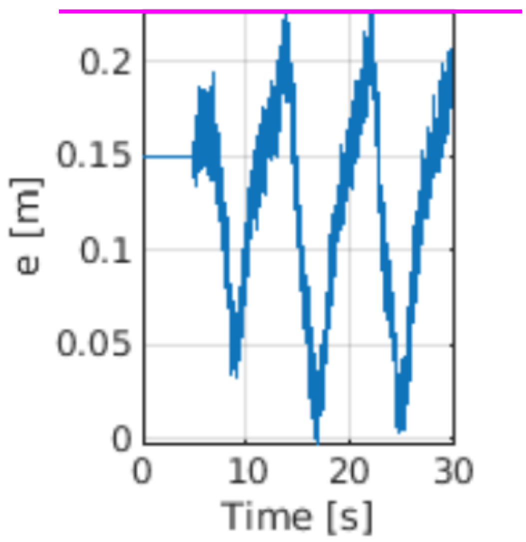
Lateral Error plots shown below (same simulation). Constraint of 25 cm lateral error indicated by pink line.

xla = 20

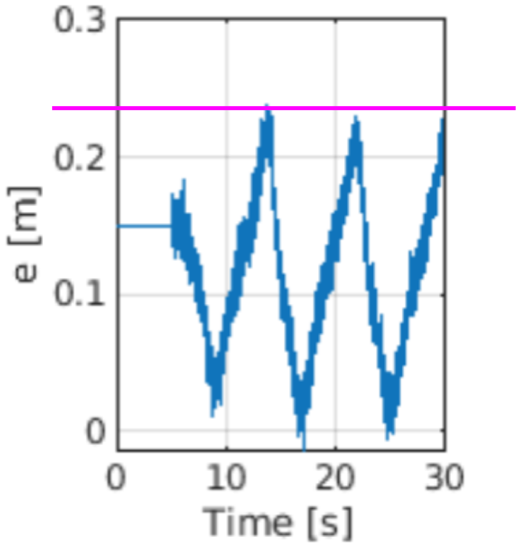
Kla = 0.1



Kla = 1000



Kla = 3500



Kla = 5000

Lookahead w/ Feedforward

Analysis for Kla

From Root Locus Plot:
We can see from our root locus plot that the system is stable for all values of Kla between 1,000 and 10,000, so we know we can use any value in this range.

From Simulation:
From the simulation results, we can see trends in both the acceleration and the lateral error in as Kla increases. Looking at the acceleration plots, we notice that as Kla increases, the peaks in the acceleration curve also increase (indicated by the pink circles in the acceleration plots). Since Kla did not affect the maximum acceleration seen by the vehicle, we chose to instead look at how it affected the jerk, or the change in acceleration. Looking at the lateral error plots, we can see that as Kla increases, the lateral error decreases. In order to both satisfy our constraint of a lateral error below 0.25m and to minimize the abruptness of change in acceleration of the vehicle, we chose a Kla value of 4,000.

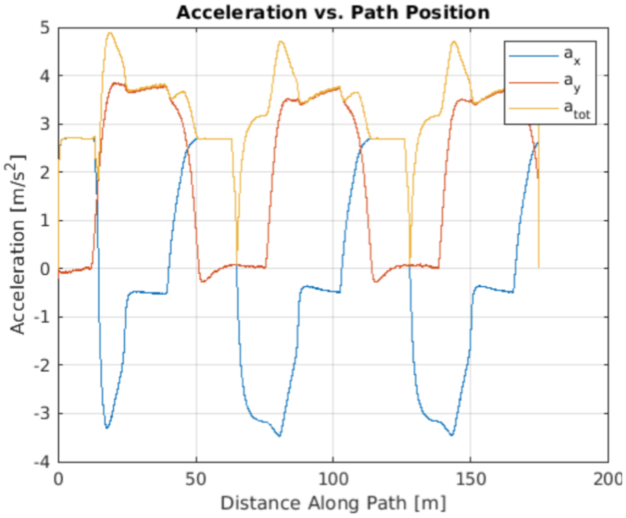
Lookahead w/ Feedforward

Accelerations

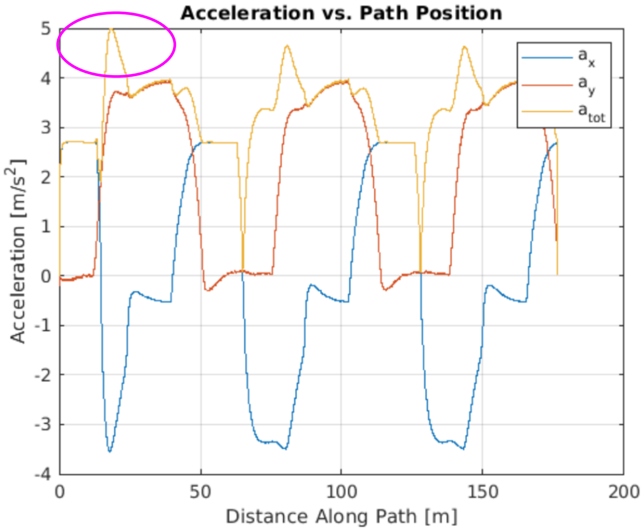
Simulation Results on more detailed environment (*sim mode = 3*) for several stable values of Kong(0.05, 0.1, 0.2) with our chosen value of Kla = 4000 and the value of xla from problem set 4. Acceleration plots shown below.

Kla = 4000

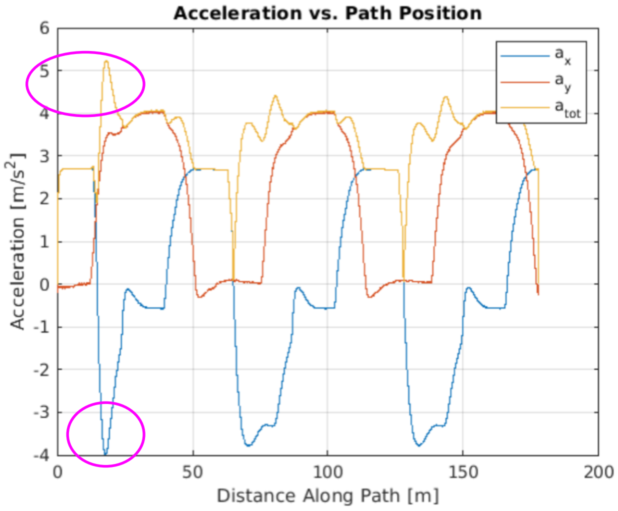
Xla = 15



Klong = 0.05



Klong = 0.1



Klong = 0.2

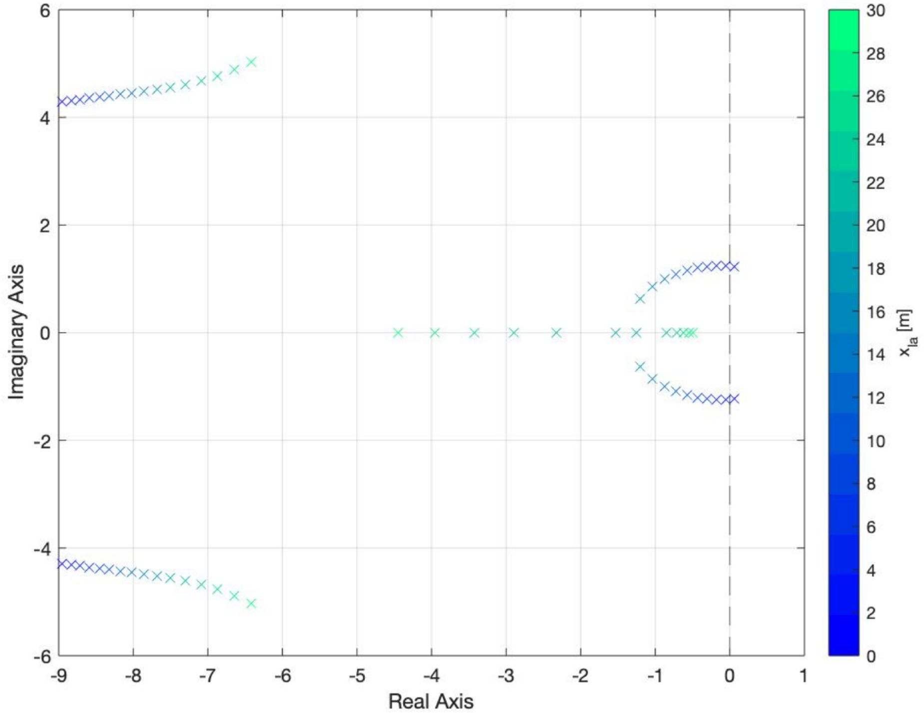
Lookahead w/ Feedforward

Analysis for Klong

From Simulation:
The simulation results for varied values of Klong showed minimal trends in most of the areas of interest (acceleration, actuator commands, and lateral error) but we did notice that the acceleration peaked slightly more when we increased Klong, and had more rounded edges when we decreased it. Keeping in mind that the higher peaks in acceleration will cause more discomfort, and the sharper the edges of the peaks the higher the jerk, we chose our value of Klong to minimize these values, at Klong = 0.05.

Lookahead w/ Feedforward

The root locus of the system with varying xla 0:30m (with Kla = 4000). xla must be greater than 2m for stability.



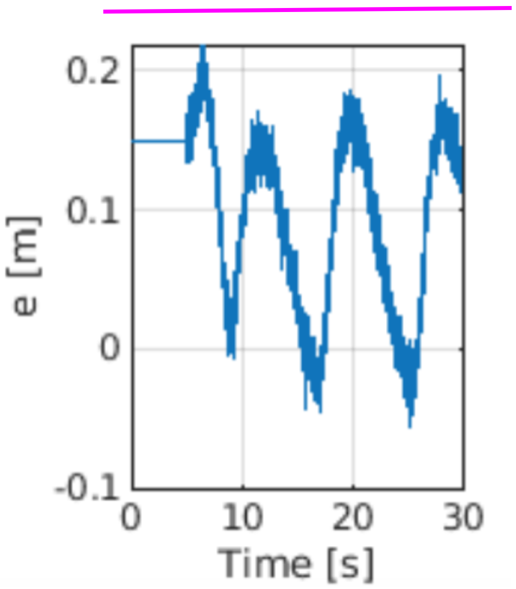
Lookahead w/ Feedforward

Lateral Error

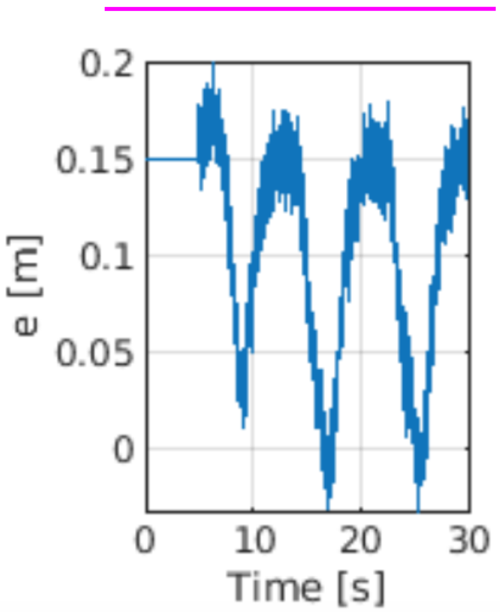
Lateral Error plots shown below for varying values of Xla, with chosen Kla. Constraint of 25 cm lateral error indicated by pink line.

Kla = 4000

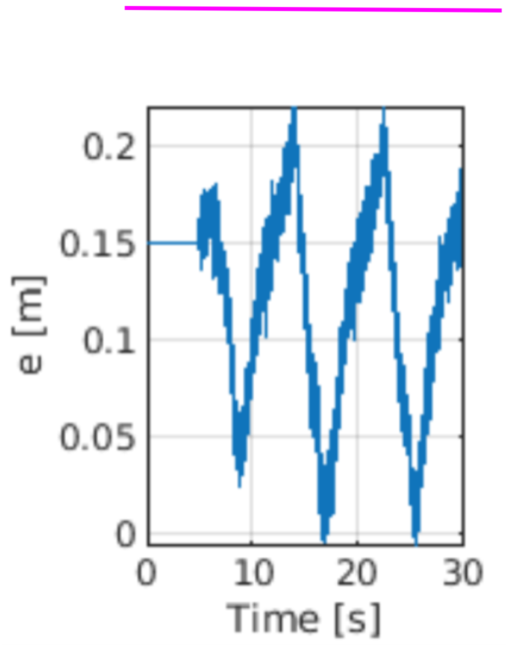
Klong = 0.05



Xla = 10m



Xla = 15m



Xla = 20m

Lookahead w/ Feedforward

Analysis for Xla

From Root Locus Plot:
We can see that for values of Xla varying from 0 to 30m, that Xla is only stable if it is larger than 2m.

From Simulation:
The simulation results for varied values of Xla showed minimal trends in most of the areas of interest (acceleration, actuator commands, and lateral error) but the lateral error was slightly higher (although still below 0.25m) for values both larger and smaller than Xla = 15m so we chose to keep it the same as the value we used in our assignment 4, Xla = 15m.

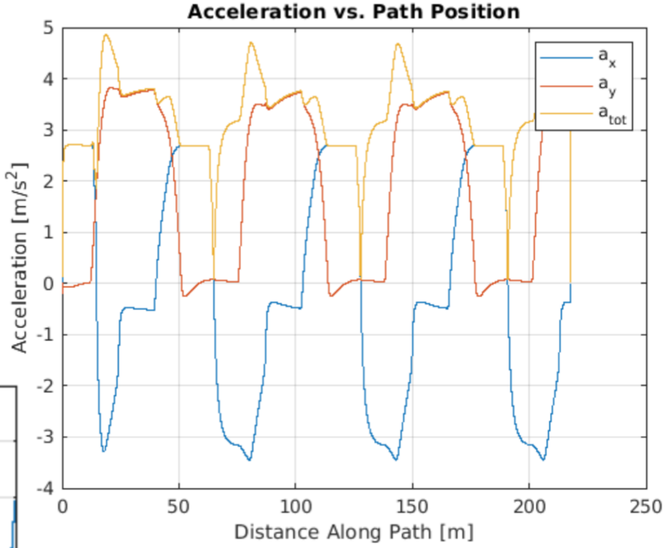
Lookahead w/ Feedforward

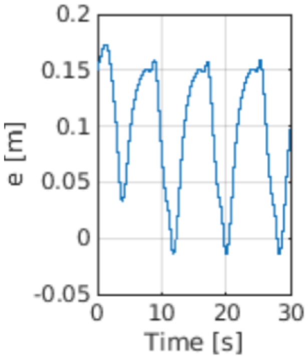
Simple vs Detailed Simulation

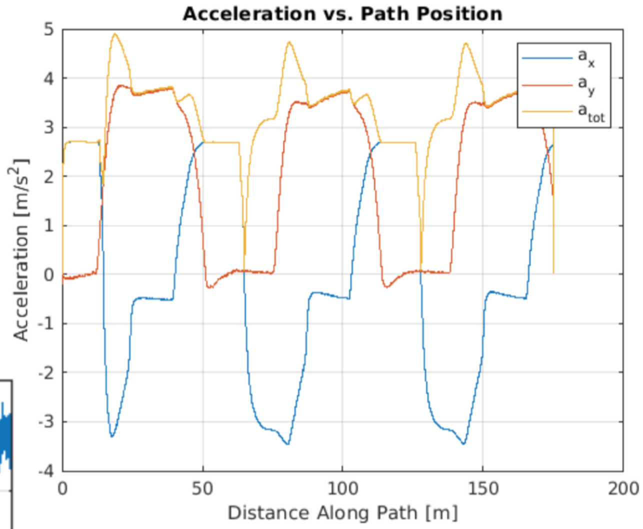
sim mode = 1

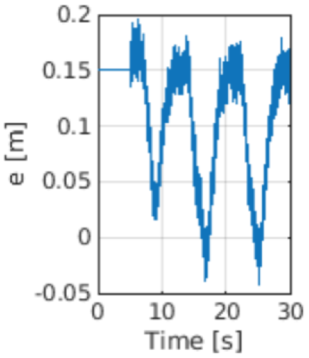
Kla = 4000

Klong = 0.05
xla = 15



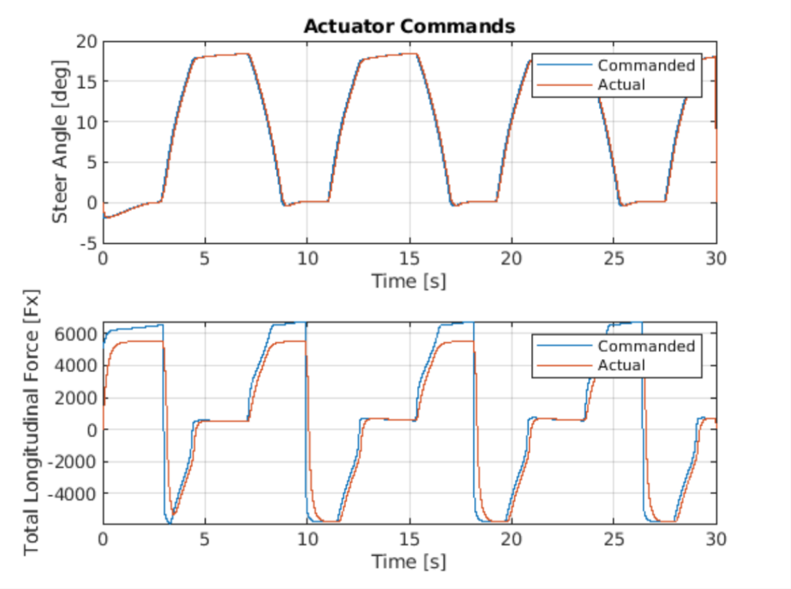






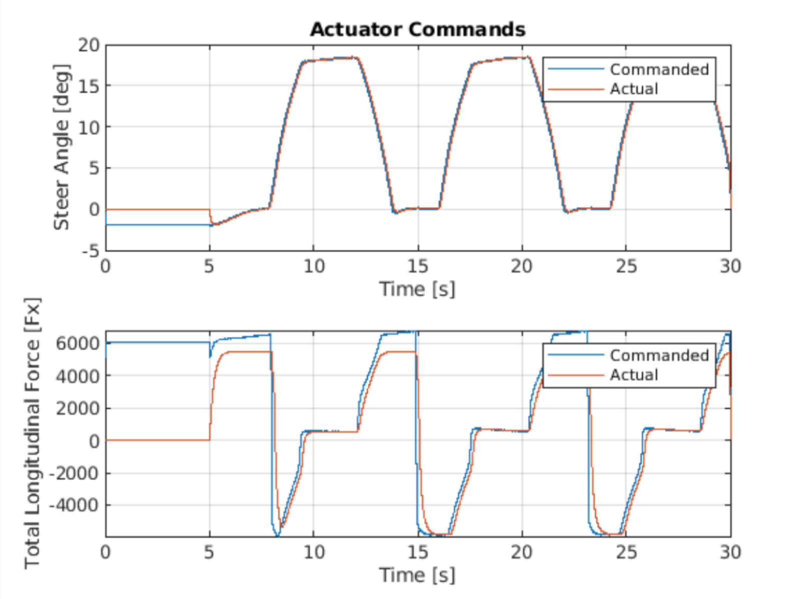
sim mode = 1

Kla = 4000
Klong = 0.05
xla = 15



sim mode = 3

Kla = 4000
Klong = 0.05
xla = 15



Lookahead w/ Feedforward

Discussion of Results

Simple vs. Detailed Simulation

Overall, our simulation plots showed the same general shape and behavior between sim modes 1, 2, and 3.

While the lateral error behavior remained about the same for all sim modes, added noise increased peak values from 0.17 m without noise to 0.2 m in sim modes with noise. We thus prioritized editing our gains to ensure these maximum lateral error values were handled well, and executed our iterative process for determining gains in sim mode 3 for this reason.

The addition of a time delay in part 3 did not have an effect on our outputs besides delaying them by 5 seconds. Once the time delay elapsed, the plots follow the same behavior as in sim modes 1 and 2.

LQR Controller

LQR Tuning

LQR Gain Matrices: Q & R

Q selection based on Bryson's rule: $\frac{1}{(maximum\ allowable\ error)^2}$

-> for **e**: $\frac{1}{0.25^2}$ (given by specs)
-> for **dpsi**: $\frac{1}{0.15^2}$ (corresponds to ~8 .6 degree heading error))
-> for the derivatives (**e_{dot} and $dpsi_{dot}$**): 30 = 1/(.18)^2

$$Q = \begin{pmatrix} 16 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 44.44 & 0 \\ 0 & 0 & 0 & 30 \end{pmatrix}$$

R selection - dictates the control effort weights

Found such that we do not have a large actuator command jump

$$R = 150$$

LQR

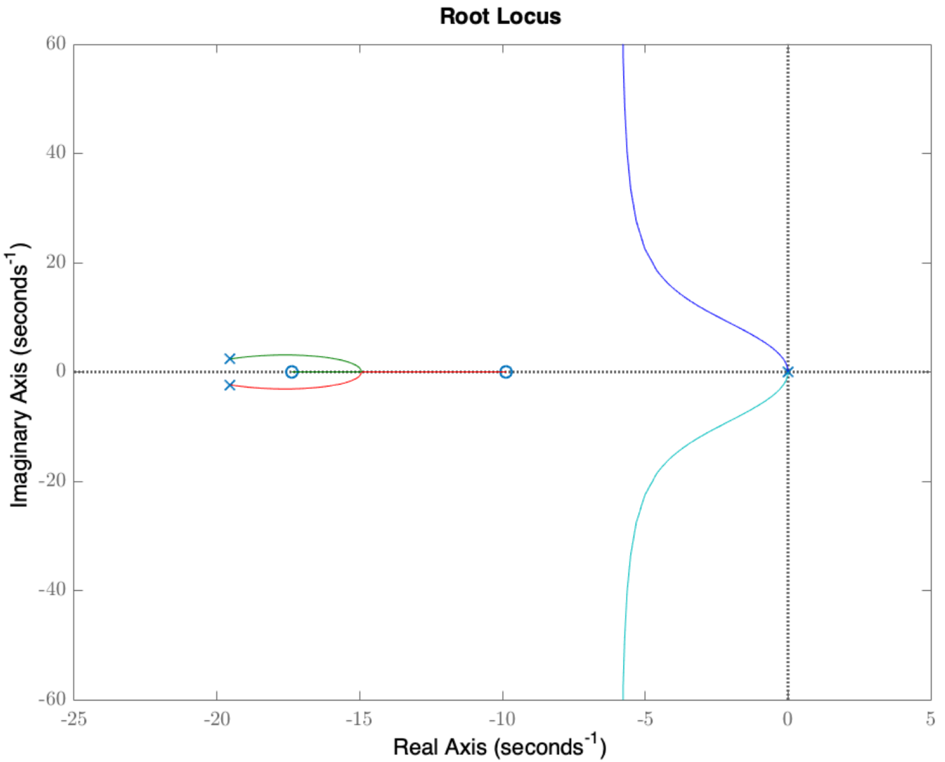
We approximated the system as time invariant. The only actually time varying aspect of our system in the velocity *Ux*

Looking at plots for velocity using our speed controller showed us a mean velocity of ***Ux,avg* = 8.6 m/s**

We use this to construct our A and B matrices such that: $\dot{x} = Ax + Bu$

LQR

We see the open loop performance of the system is stable for all positive gains K.

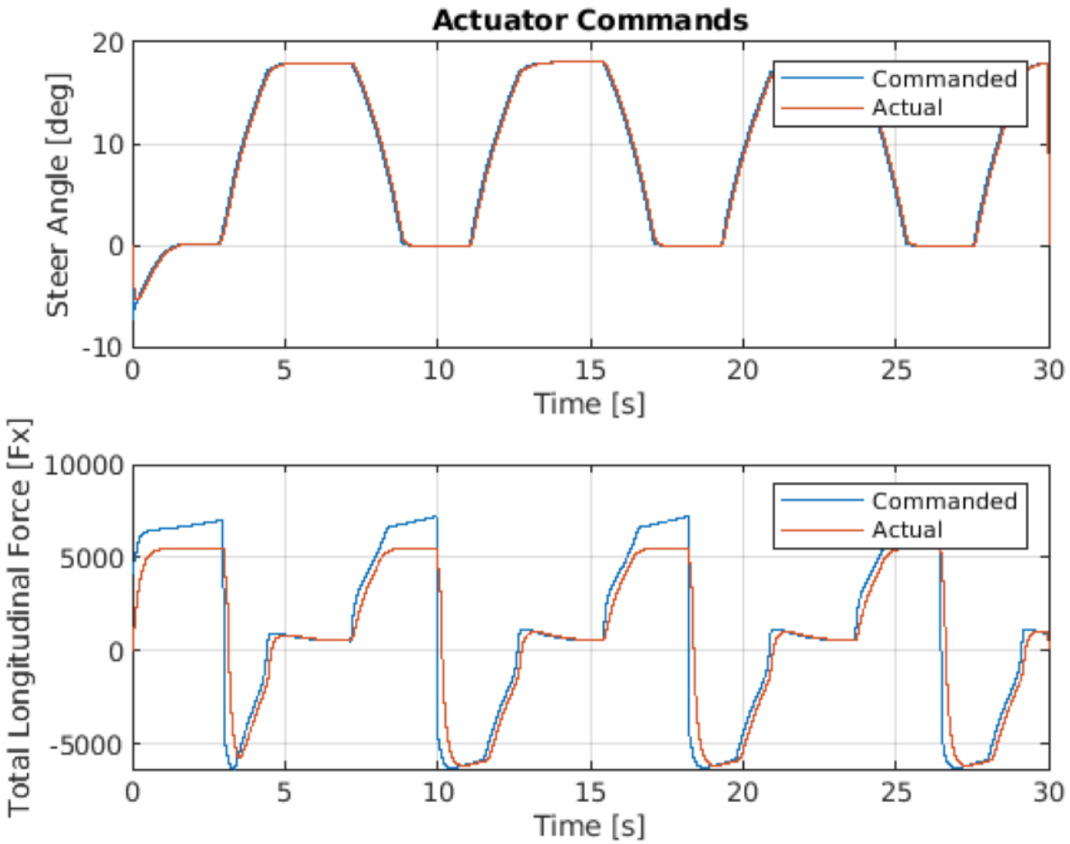


LQR Gains we are using: **K = [0.3266 0.2508 1.9693 0.2311]**

LQR

Simulation Results

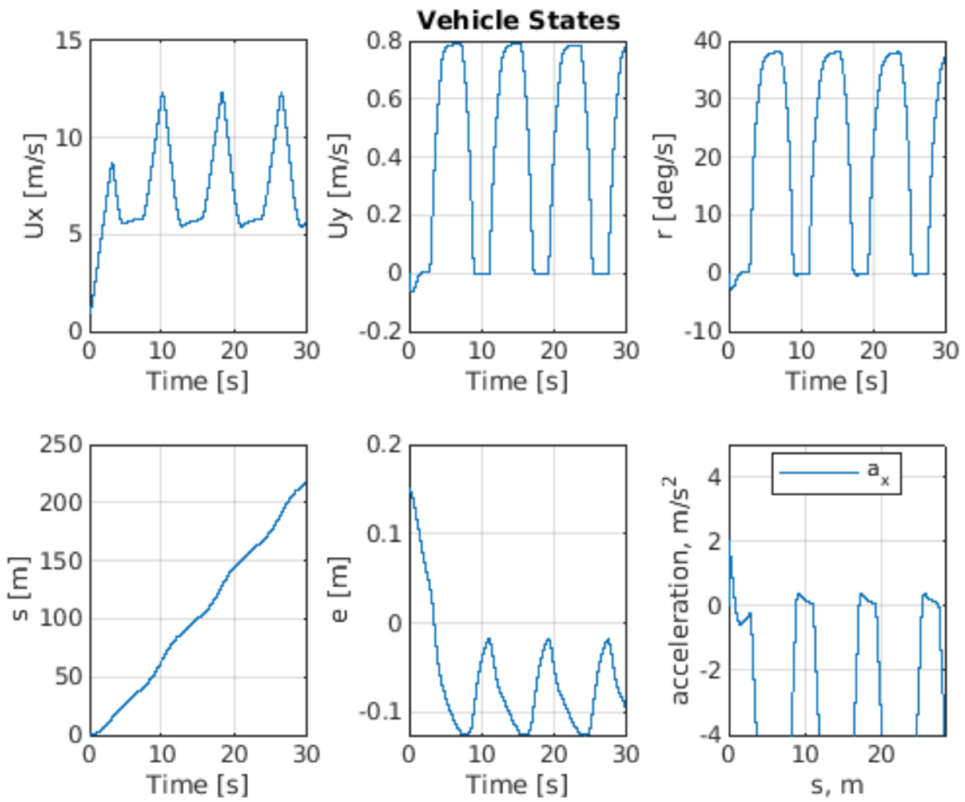
(sim-mode 1:
w/ actuator
dynamics
no noise)



LQR

Simulation Results

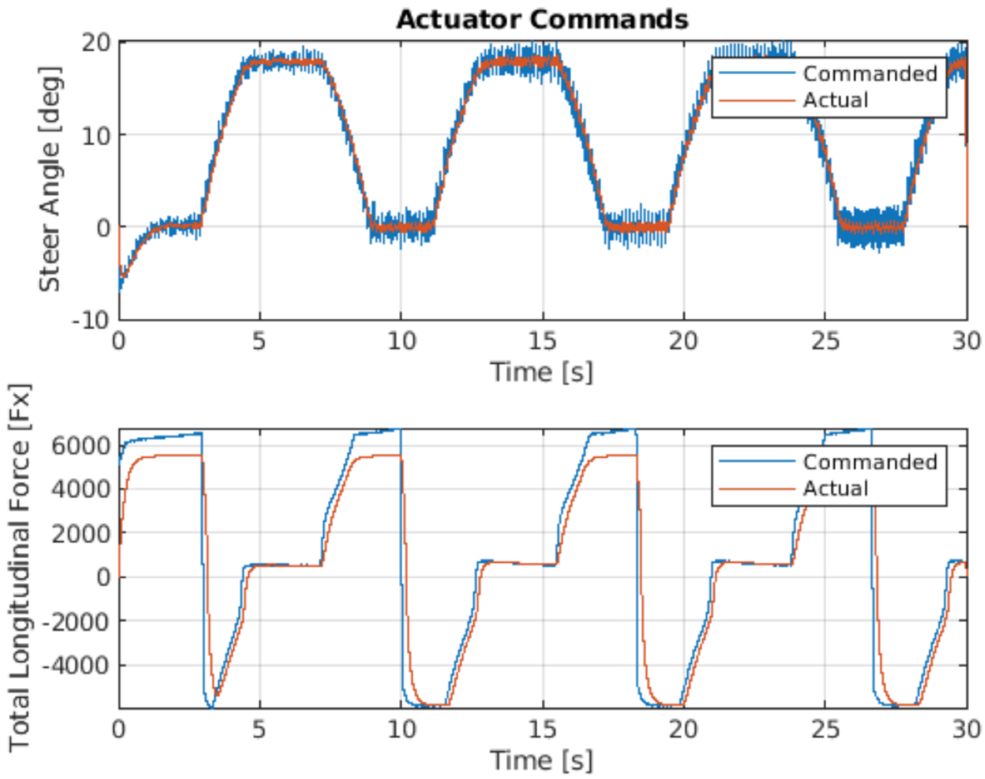
(sim-mode 1:
w/ actuator
dynamics
no noise)



LQR

Simulation Results

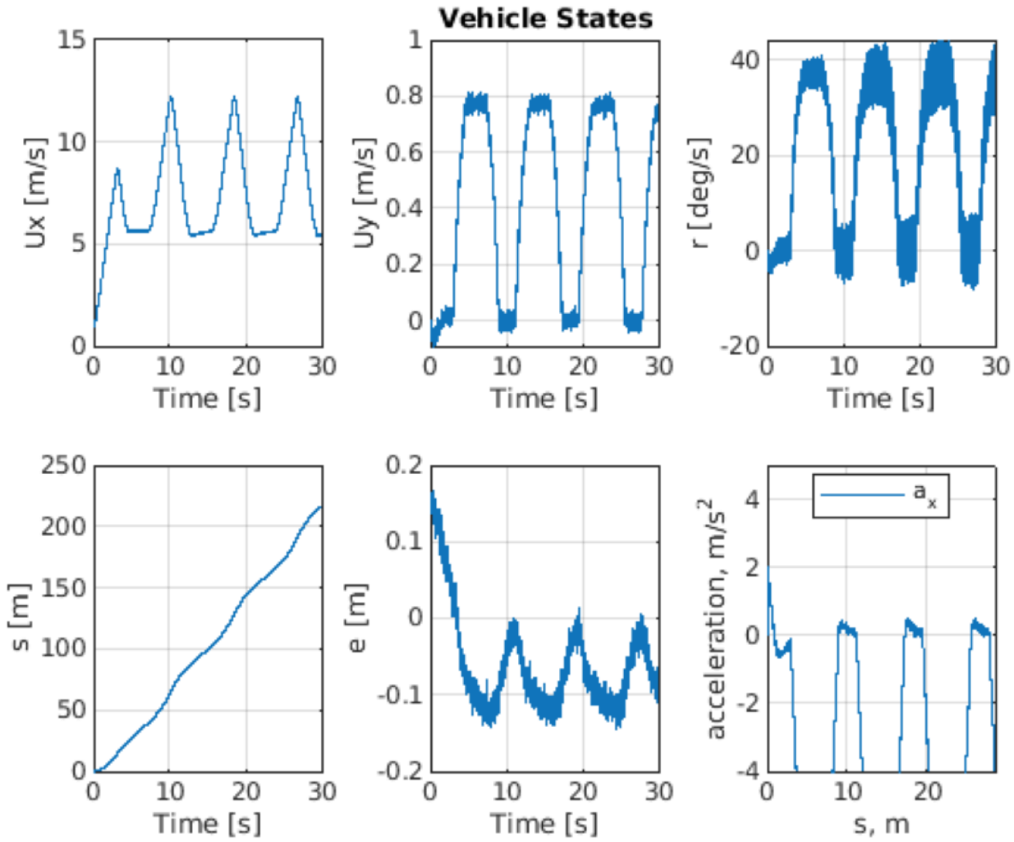
(sim-mode 2: w/ actuator dynamics AND noise)



LQR

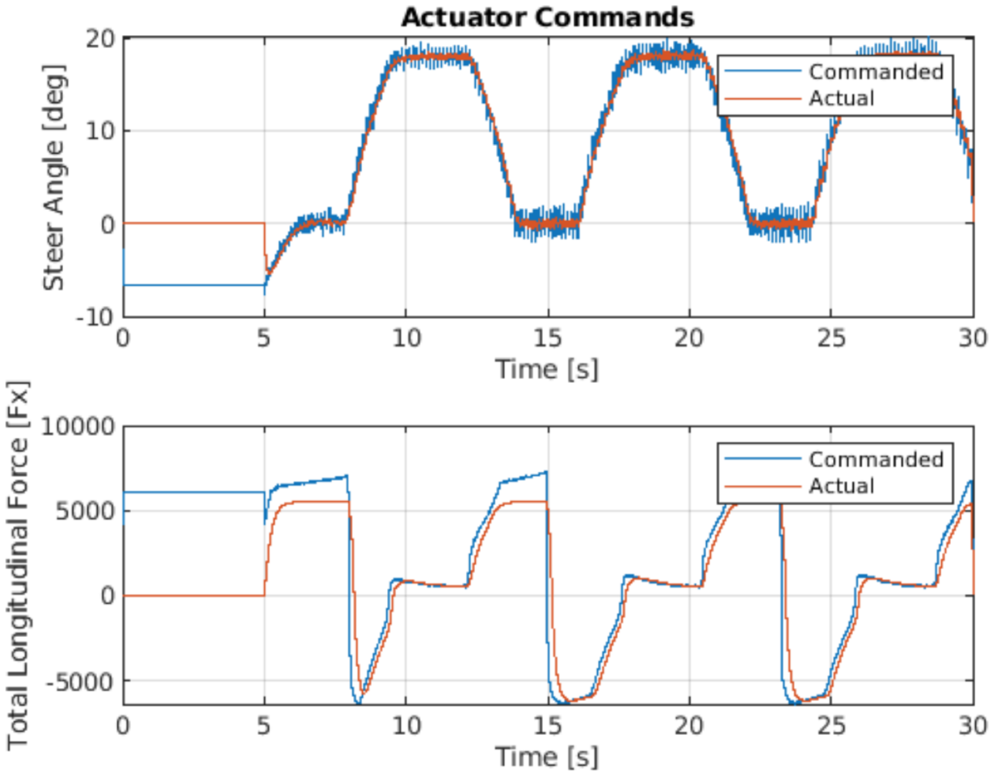
Simulation Results

(sim-mode 2: w/ actuator dynamics AND noise)



LQR

Simulation Results
(sim-mode 3)

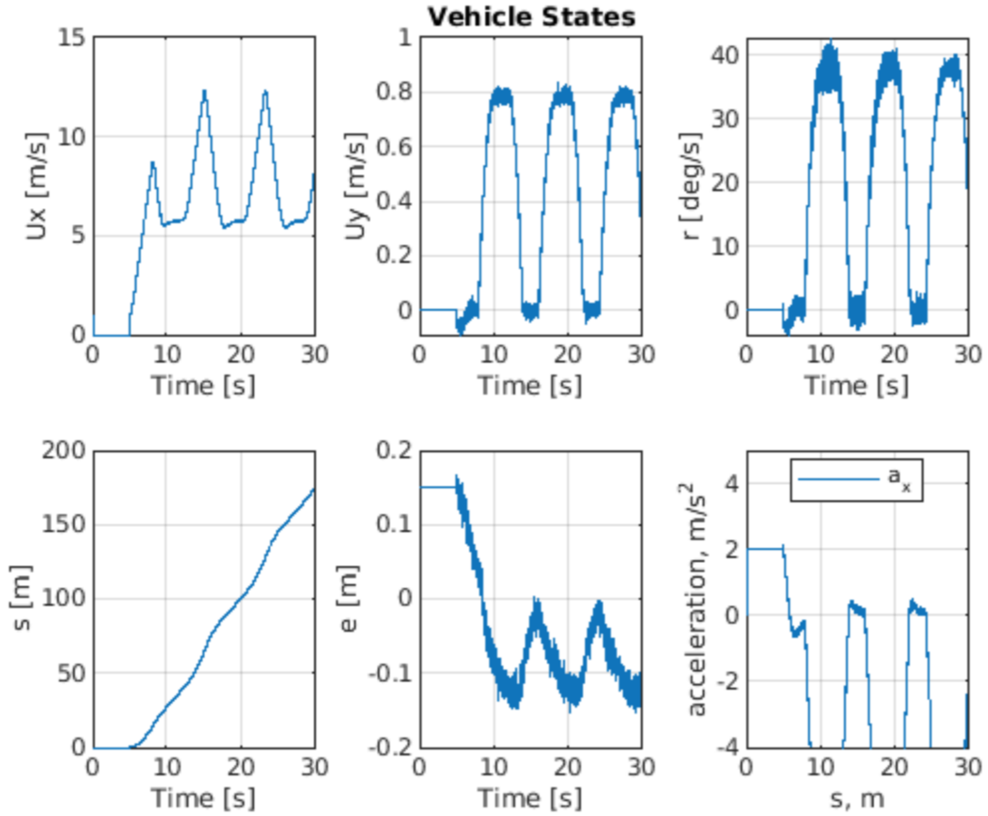


LQR

Simulation Results

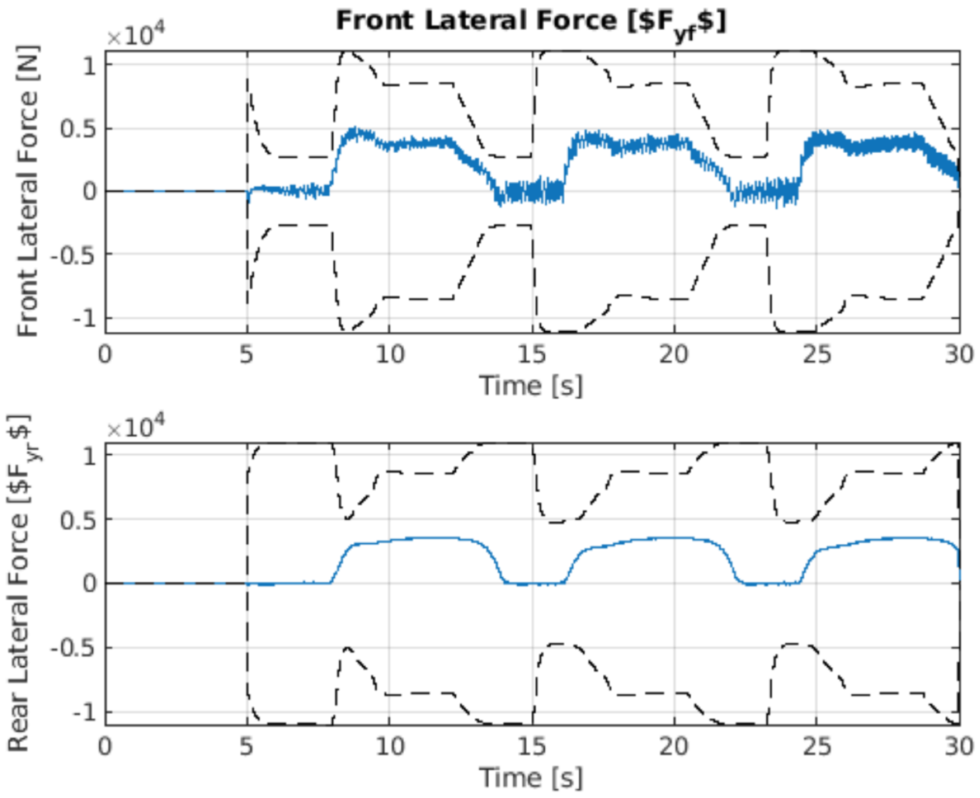
(sim-mode 3)

Error spec is met, our maximum error for this simulation is around 15 [cm].



LQR

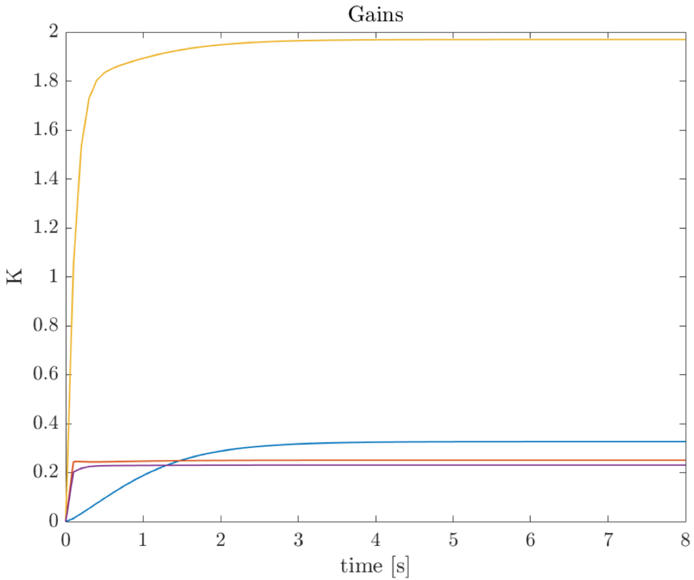
Simulation Results
(sim-mode 3)



LQR Solver Code

In order to further our understanding of LQR control, our team manually coded the a continuous time LQR solver instead of using the *lqr( )* function built into *Matlab*.

Below is a plot of the solver finding the regulator gains using an ODE solver.




```
function K = lqr_solver(A,B,Q,R)

% Continuous Time LQR Solver

n = size(Q,1);
P = zeros(n,n); % terminal cost matrix
tf = 8; % time horizon
dt = 0.1;

tspan = tf:-dt:0;
[t,x] = ode45(@(t,x) g1(x,A, B, Q, R), tspan, P);
reg_gains = zeros(length(tspan), n);

for j = 1:length(tspan)
    x_new1 = reshape (x(length(t)-(j-1),:), [n,n]);
    reg_gains(j,:)= inv(R)* B' * x_new1;
end

figure(11)
plot(t, reg_gains);
xlabel('time [s]'); ylabel('K')
title('Gains');

    % ODE45 Subfunction
    function dvdt = g1(x,A, B, Q, R)
        x = reshape(x, [4,4]);
        dvdt = -(Q - x*B*R.^(-1)*B.'*x + x*A + A.'*x);
        dvdt = reshape(dvdt, [n^2,1]);
    end

K = reg_gains(1, :); % returns gains

end
```

Simulation Mode Comparison

For our LQR controller, we similarly did not notice a significant difference between the different simulation modes, other than the obvious effect of noise in the graphing of all states as well as the delay in the lateral error graph.

The lateral error graph is displaying slightly larger error when noise is present, which, however, is a natural effect of the presence of noise, rather than a system change to our dynamics and control. Our actuator plots also track similarly in all three modes.

This is because our controller is robust enough to compensate for the noise and we used measurements to construct our derivatives for lateral error as well as heading error.

Readiness for Niki

Based on some stuff, we feel our **Lookahead** and **LQR** controllers are ready to go on Niki:

-Actuator plots show a smooth transitions to the required steering angle (no spikes or step changes)

-Accelerations are minimized, and the change in acceleration (the jerk) is minimized to a comfortable range.

-Lateral error is reduced to less than 0.25m for both controllers

Lessons Learned

Julianne: I've gained a better grasp on how adjusting system parameters affect a real-world system's performance. I appreciated starting from as abstract of a place as root locus to determine the range of values we could manipulate, and then seeing the effects of this on realistic data, especially in the lateral error and actuator commands. In class it feels like you can just choose any reasonably functional gain and run with it, but this project showed me further limitations of what we can actually implement.

Natalie: I learned a lot about how to design your own controller. I also learned that it is important to figure out what range of values for gains will make the system stable before starting to vary the gains to see the effects. And most importantly I figured out how to share screen on Zoom with sound, and that the Cars movie is actually accurate.

Nefeli: This was a super fun project! I am glad I was able to implement knowledge I gained from ENGR205 and AA203 into designing an LQR controller from scratch. Especially given that we could not use the lqr function, and we therefore had to use the continuous time ODE version of the Riccati recursion. We also got to see Bryson's rule in action when tuning our Q and R gains. Generally, it was fulfilling to bridge knowledge from different classes :)

Aubrey: I learned a lot doing this project. It was very fun to implement an LQR controller. I had taken various controls classes and this was the first time I really got my feet wet actually implementing the controller on a real world system. It was fun to derive the gains by manually coding a continuous time Riccati recursion. It was very effective and was fun to tune the Q and R matrices to get very good performance. I look forward to seeing our code run on Niki.

For your viewing pleasure: https://www.youtube.com/watch?v=iuJDhFRDx9M