EEL 6616 Adaptive Controls Final Project: Creating a One Size Fits All Steering Controller

Eduardo Rosado, Brandon Foggie

Mechanical and Aerospace Engineering – University of Central Florida
Orlando, FL
eduardorosado@knights.ucf.edu
brandonfoggie@knights.ucf.edu

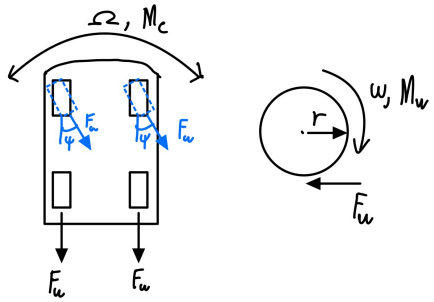
***Abstract*—This report contains information and data revolving the design of an adaptive control law used in multiple vehicles to attain similar steering responses. The study hopes to show light to the possibilities of adaptive controllers in the automotive industry.**

I. Introduction

When designing a car, one of the most important components is nailing down how good it drives. Handling qualities in cars can be a make-or-break factor for customers on the look for a new vehicle. This project involves designing an adaptive controller that yields desired handling qualities given any car model and some extra specifications. In order to accomplish this, a simple car model is used with second order system approximations for both throttle and steering engine and actuator models respectively. A robust controller is to be designed for sportier cars where the performance will be determined based on responsiveness throughout different input frequencies and amplitudes. We believe this is a very interesting project because it can serve well in the future where autonomous cars become more and more prevalent. Imagine if all any car manufacturer needed to do to fit their cars with autonomy is just call one company that has an all-in-one sensor suite, including control laws. This is the main motivation to endeavour in this idea and although we will only scratch the surface we hope to at least prove adaptive controllers for vehicle handling is worth looking into.

II. System Modeling

A car's steering characteristics is determined by wheelbase and speed, therefore a throttle and steering mechanism needs to be modeled in order to asses performance for the goal in mind (car handling qualities). We will assume a AWD car model where equal force is exerted by all four wheels. The front wheels rotate and induce a moment in the car that causes the car to turn due. The force of the wheels is calculated via the wheel speed and the moment it would induce per wheel. The diagram below shows the general diagrams from which the equations of motion are going to be derived.



The symbols are described below:

$$\omega = wheel\ angular\ velocity$$
$$M_w = wheel\ moment$$
$$F_w = wheel\ force$$
$$r = wheel\ radius$$
$$\psi = wheel\ rotation\ angle$$
$$\Omega = car\ angular\ velocity$$

The equations of motion are then organized by forces and moments in X and Y as shown:

$$\Sigma F_x = 2F_w + 2F_w cos\psi = m\ddot{x}$$
$$\Sigma F_y = 2F_w sin\psi$$
$$M_c = \Sigma F_y R = I_c\dot{\Omega}$$
$$M_w = F_w r = I_w\dot{\omega}$$

These can be organized into state space format as follows:

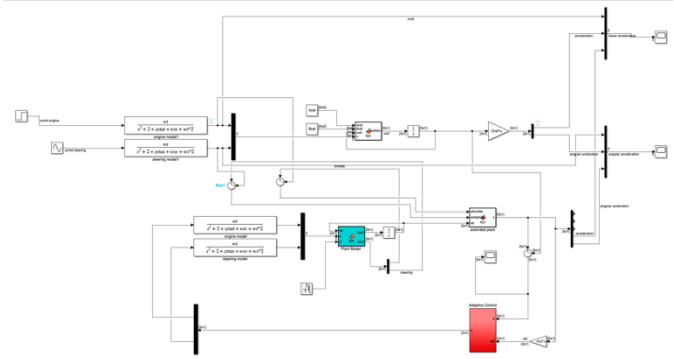
$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\Omega}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & -c_x & 0\\ 0 & 0 & -c_\Omega\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \Omega\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{4I_w}{mr} & 0\\ \frac{I_w R}{I_c}\psi & \frac{I_w R}{I_c}\end{bmatrix}\begin{bmatrix}\dot{\omega}\\ \psi\end{bmatrix}$$

Where road friction was modeled by the damping terms c_x and c_Ω for forward acceleration and angular acceleration respectively.

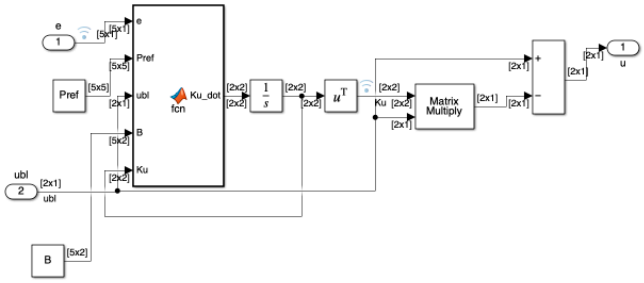
III. Control Law Design and Simulation

Given the dynamics that were explained above that contains multiple inputs and multiple outputs (MIMO) a choice had to be made in terms of what type of adaptive controller we wanted to use. So with this information at hand we chose to design a MIMO robust controller. With the knowledge that we also want to focus on the heading of the car and making sure that the heading converges to the input given as well as the acceleration we designed a baseline PI controller augmented with MRAC to be able to easily tune the system to our needs. The reason this method was chosen, is for its ease of use, due

to the extended states that are inherit within the controller design. Having the two added states being the error of the acceleration and heading these were obvious candidates to target for higher penalties within the Q matrix used in the LQR method to determine the optimal gains for our system. Utilizing this method also gives you a lot of control in terms of the response of the reference model. The same methodology was also used while tuning the gain matrix in the adaptive block of the model as shown below.



In terms of robustness, we wanted to utilize one of the methods we learned in class to account for things such as gravel or any other sudden losses of traction. To simulate this affect, disturbance was added to the state space model, and we used dead zone to account for
this extra noise within the system.



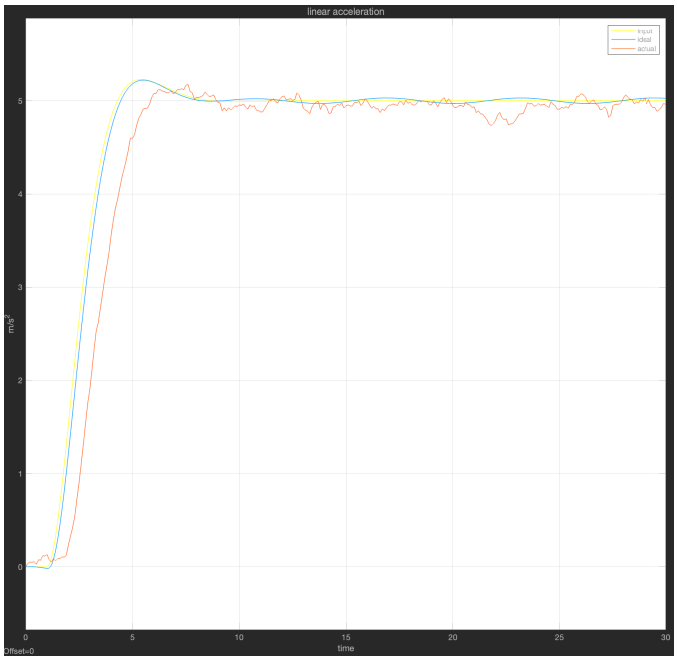

```
function Ku_dot    = fcn(e,Pref,ubl,B)
% Adaptive parameters

Gamau = [1000 0 ; 100 1000];

e0 =.1;

% Dead zone
if e >abs(e0)
Ku_dot   = Gamau*ubl*e'*Pref*B;

else
Ku_dot = zeros(2,2);
end
```

IV. RESULTS AND DISCUSSION

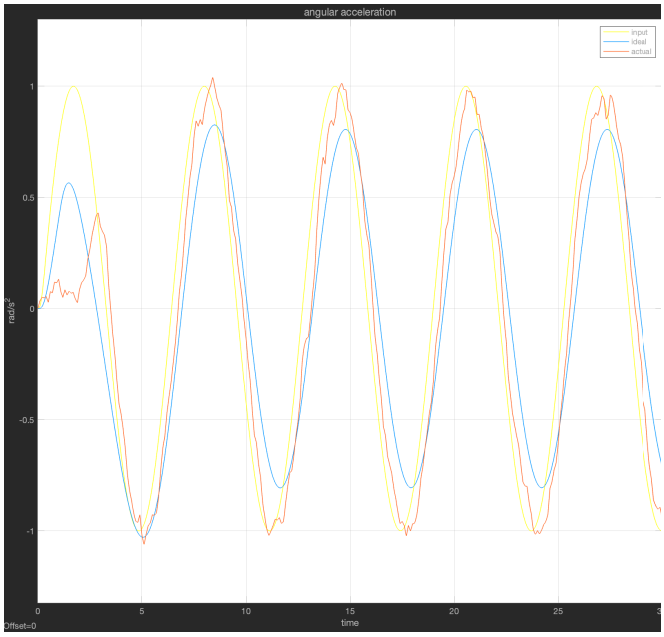
To gauge performance, the systems linear and angular accelerations will be shown for two separate car configurations. The main goal was to assess the turning performance but since the MIMO system already outputs forward acceleration then it doesn't hurt to show that performance. The first vehicle had the following physical properties:

Weight - 300
Car moment of inertia - 300
Wheel moment of inertia - 300
Wheelbase - 300
Wheel radius – 300

The gains were tuned for the first vehicle model to attain the following response in forward direction:



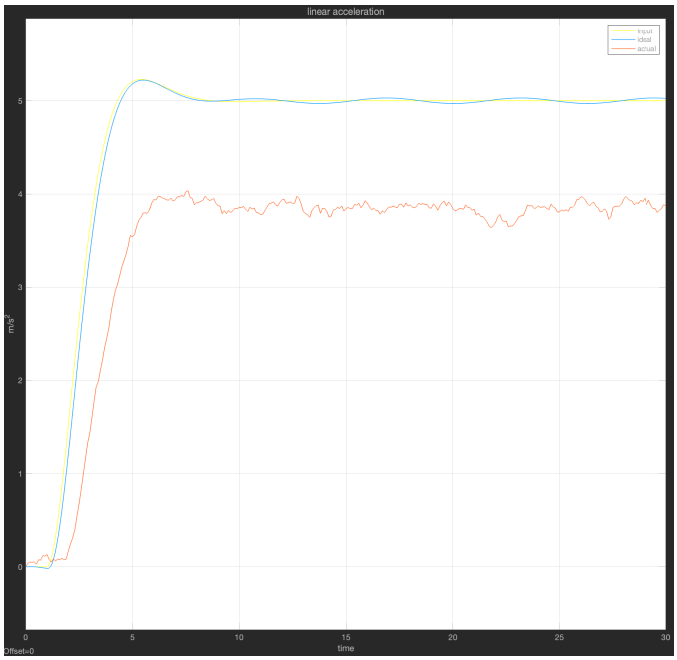
Although noisy, it follows the input well enough to start with. The angular acceleration response, the component we care most for, looks quite good with minor disturbances.

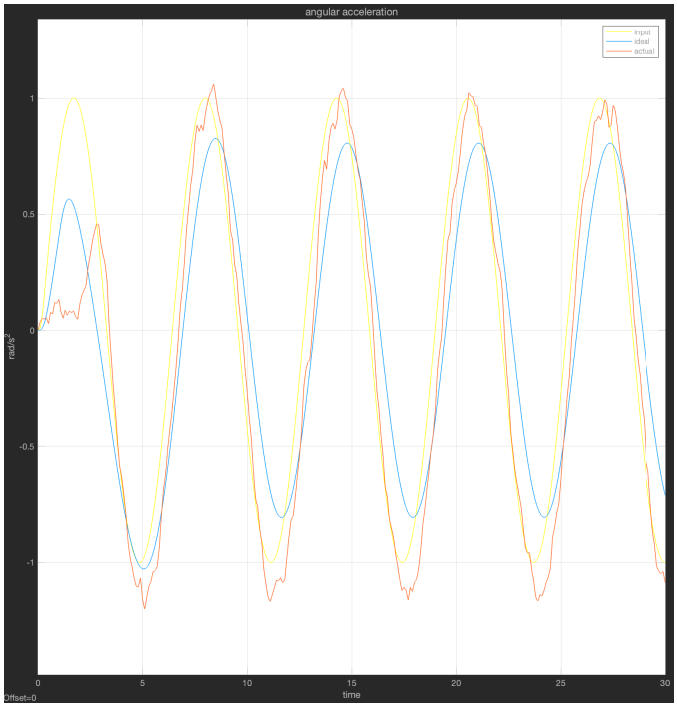


The same adaptive laws were applied without changing any tunable parameter within the controller. The reference model was changed due to changing vehicle physical properties, but any adaptive laws were remained untouched. The second vehicle had the following physical properties:

Weight - 300
Car moment of inertia - 300
Wheel moment of inertia - 300
Wheelbase - 300
Wheel radius – 300

Because the adaptive laws were left alone completely, there is a steady state error on the forward acceleration results. However, it seems like the angular acceleration tracked well without changing any tunable parameters on the second vehicle.





Adding the adaptive laws with deadzone modification allowed for more uncertainty tracking for the unknown parameters in the A matrix compared to the baseline controller which only had a viable solution for one vehicle but failed for any other vehicle configuration. Although still a work in progress, the results verify that with minimal changes you can implement the adaptive laws to different vehicles and attain good handling quality results.

V. Conclusion and next steps

The results show that a preliminary design of a one size fits all steering controller is effective in achieving desired handling qualities in a vehicle. For now, the implementations were done on a simple model which utilizes approximations for actuators and servos. Next steps include verifying with multiple automobile types and properly tuning for them to see how difficult it is to adapt to different physical properties in the system.

References

[1] E. Lavretsky and K. A. Wise, Robust and adaptive control: With aerospace applications. London: Springer, 2013.

Matlab code

```
% car sim setup

% steering model
ws = 10;
zetas = 0.7;

% throttle model

wt = 1;
zetat = 0.7;

%sim setup
Iw = 3;
Ic = 30;
m = 80;
r = 3;
R = 15;
wheel_damp = 1;
turn_damp = 1;

wdot0 = 2; %have to make a constant to
get reference model
psi0 = 2;  %have to make a constant to
get reference model

%base model

Ap = [0 1 0; 0 -wheel_damp 0; 0 0 -
turn_damp];
Bp = [0 0; r*Iw/(m*r) 0; Iw*R*psi0/Ic
Iw*R*wdot0/Ic];
Cp =  [0 1 0;0 0 1];
Dp = [1 0; 0 1];
lambda = [1 0; 0 1];

%extended model

A = [zeros(2,2) Cp;...
     zeros(3,2) Ap];
B = [Dp;...
     Bp];
C = [zeros(2,2) Cp];
D=Dp;
Q = diag([100 10 1 1 10]);
R = diag([1 1]);
Kxt = lqr(A,B,Q,R);
Aref = A-B*Kxt;
Cref = C-D*Kxt;
Bref = [-eye(2);zeros(3,2)];
Qref = diag([10000 1 0 1000 100]);
Pref = lyap(Aref',Qref);
run('carSim')
close all;
```