Alexander Chew
999766439
Andrew Choi
999796522
Navea Dasz
998884752

EME 171 Lab #5: Autonomous Electric Vehicle

Part 1: Modeling and Implementation

Aerodynamic drag and rolling resistance are the two nonlinear resistances that the vehicle experiences and the equations that define both resistances are:

$$F_{Drag} = \frac{1}{2}\rho A_f C_D v|v|, \qquad F_{Roll} = MgC_R\text{sgn}(v)$$

The state variables for this system are the motor flux linkage (P_L) and the vehicle momentum (P_M) and the system input is the battery pack which is treated as an ideal voltage source (u_{in}). The state equations of the system are listed below:

$$\bar{x} = [P_L \;\; P_M]; \qquad u = u_{in}$$

$$\dot{P}_L = u_{in} - \frac{R_w}{L_w}P_L - \frac{T_m G_R}{RM}P_M$$

$$\dot{P}_M = \frac{G_R}{R}\left(\frac{T_m}{L_w}P_L - \frac{b_t G_R}{RM}P_M\right) - \frac{1}{2M}\rho A_f C_D P_M\left|\frac{1}{M}P_M\right| - MgC_R sgn\left(\frac{1}{M}P_M\right)$$

The system is simulated in Matlab by applying a step input to u_{in} with a magnitude of 100 volts, as shown in Figure 1 below. The signum function (sgn()) is approximated by the equation, $sgn(v) \cong v / (|\, v \,| + n)$, where v is the flow of the resistive element and n is a small number.

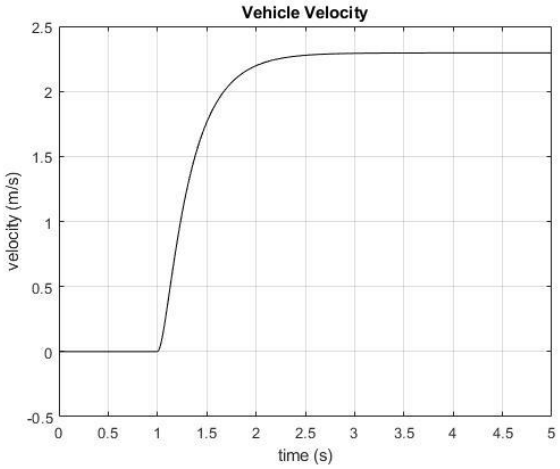

Figure 1: Vehicle velocity as a response to a 100 voltage input to u_{in}

Part 2: Controller Design

A velocity controller is implemented using proportional-integral (PI) control that generates an input voltage u_{in} based on the error between the vehicle's desired velocity (v_{ref}) and the actual vehicle velocity (P_M / M). The relationship between the system's input voltage, velocity difference, displacement difference, and the proportional and integral gains, K_p and K_i, respectively, is shown below:

$$u_{in} = K_p\left(v_{ref} - \frac{p_M}{M}\right) + K_i\left(d_{ref} - d\right)$$

While simulating a unit step response, the proper gains are determined by a trial-and-error process to meet the following specifications: a rise time less than 0.5 seconds, a settling time less than 2.0 seconds, and an overshoot of less than 10%. The controller gains K_p and K_i were determined to be 176 and 500 respectively, giving a rise time of approximately 0.1 seconds, a settling time of 0.56 seconds, and an overshoot percentage of 8.8%. The plot of the velocity response under a step input with the proposed PI controller is shown in Figure 2 below:

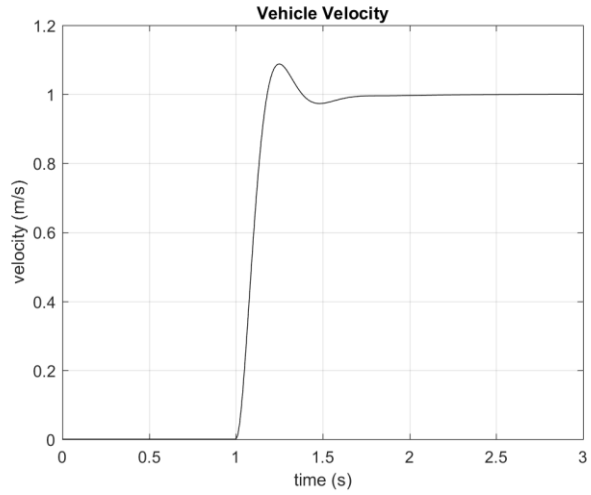

Figure 2: Velocity response to Step input with K_p = 176 and K_i = 500

K_p and K_i were found by first setting K_p and K_i to small values, then increasing K_p while keeping K_i at a small value until the overshoot percentage was just below 10%. Then K_i was increased until the steady state value was near one and the steady state error was very small. K_p primarily affects the speed of the response while K_i primarily affects the accuracy of the response. However, both K_p and K_i did have some impact upon the primary effect of the other; for instance, if K_i was too large, overshoot percentage could not get below a certain level, so although they primarily affected one characteristic, they did have some affect on the others.

When K_p is set to zero, the vehicle velocity oscillates around equilibrium until settling at equilibrium after 10 seconds, so it is as if the system is underdamped. When K_i is set to zero, the system reaches steady-state fast, however the system has a large steady-state error of around 20%. By setting each term individually to zero, the effect each constant has on the controller is clearly evident.

Part 3: Testing

Using the controller designed in the previous part, the vehicle is simulated over a realistic driving scenario and its performance is assessed with respect to vehicle emissions and energy consumption. The actual vehicle velocity under the LA92 velocity profile is shown in Figure 3, and both the actual velocity and the LA92 reference velocity are shown in Figure 4.

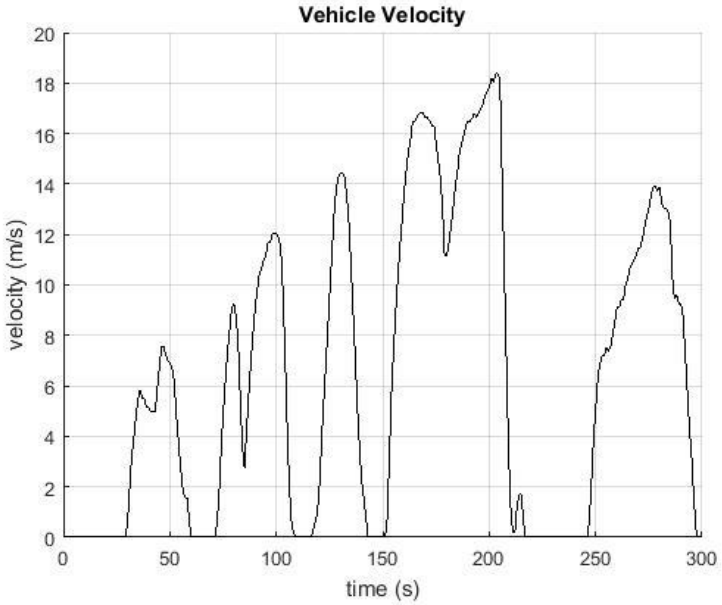

Figure 3: Vehicle Velocity Response to LA92Oracle.m velocity input

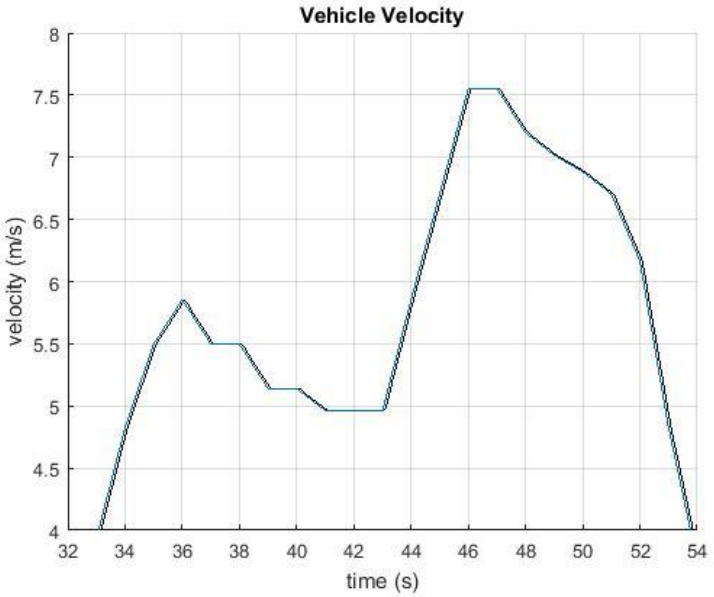

Figure 4: Vehicle Velocity and Reference Velocity showing only 32s to 54s and 4m/s to 8m/s

The actual velocity lags about 0.085 seconds behind the command velocity. This was found by zooming into the plot and using points from each line at the same velocity and finding the

difference in time. There was not significant overshoot, undershoot, or oscillations in the response, and the performance of the controller did not change with the vehicle speed.

The average accelerating efficiency in terms of power in vs power out was determined to be 62.7%. This result seems relatively reasonable because of the inefficiencies in the drivetrain and the high coefficient of drag compared to more aerodynamic vehicles.

The ratio of meters traveled per joule of energy was found to be 0.0015 (m/J). Because a Joule is such a small unit of energy this answer is reasonable.

Code Printout

lab5_master.m

```
global Rw Lw Tm M btal R Gr Cr g Cd rho Af n kp ki
Rw = 0.3; %[ohms], armature winding resistance
Lw = 0.015; %[H], armature winding inductance
Tm = 1.718; %[Wb], transduction coefficient
M = 2200; %[kg], vehicle mass
btal = 0.05; %[Nms/rad], drive shaft friction
R = 0.2; %[m], wheel radius
Gr = 5; %gear ratio
Cr = 0.006; %rolling resistance coefficient
g = 9.81; %[m*s^-2]
Cd = 0.32; %drag coefficient
rho = 1.21; %[kg*m^-3]
Af = 2.05; %[m^2]
n = 0.01;
kp = 176;        %proportional gain
ki = 500;  %integral gain


p3IN = 0;
p9IN = 0;
dref_IN = 0;
dact_IN = 0;

%initial = [p3IN p9IN];
initial = [p3IN p9IN dref_IN dact_IN 0];

%tspan = 0:0.01:5;
```

```
tspan = 0:0.01:300;
tspan2= 0:1:300;

[t, s] = ode45(@lab5_eqns,tspan,initial);
ext = zeros(length(t),2);
%ds = zeros(length(t),2);
ds = zeros(length(t),5);

for i = 1:length(t)
   [ds(i,:), ext(i,:)] = lab5_eqns(t(i), s(i,:));
end

vref=zeros(length(t),1);
for i=1:length(t)
 vref(i) = LA92Oracle(t(i));
end

Pin=(s(:,1)/Lw).*ext(:,1);
Pout = (s(:,2)/M).*ds(:,2);
Pin_acc = Pin(Pout>0);
Pout_acc = Pout(Pout>0);
Pin_accavg = mean(Pin_acc);
Pout_accavg = mean(Pout_acc);
Peff = Pout_accavg/Pin_accavg;

mpJ = s(end,4)/s(end,5);

disp(mpJ)

disp(Peff)

%plot for vehicle velocity
figure('Name','Vehicle Velocity','NumberTitle','off','Color','white')
hold on
axis([32,54,4,8])
plot(t,s(:,2)/M,'k'), grid on
plot(tspan2,LA92Oracle(tspan2))
hold off
title('Vehicle Velocity')
ylabel('velocity (m/s)')
```

```
xlabel('time (s)')
```

Lab5_eqns.m

```
function [ds, ext] = lab5_eqns(t,s)
global Rw Lw Tm M btal R Gr Cr g Cd rho Af n kp ki

p3 = s(1); % motor flux linkage
p9 = s(2); % vehicle momentum
dref = s(3);
dact = s(4);

vref = LA92Oracle(t);

uin = kp*(vref - (p9/M))+ ki*(dref - dact);

p3_dot = uin - (Rw/Lw)*p3 -(Tm*Gr/(R*M))*p9;
p9_dot = Gr/R*((Tm*p3/Lw)-(btal*Gr*p9/(R*M)))-(rho*Af*Cd*p9*(abs(p9/M))/(2*M))-
(M*g*Cr*(p9/M)/(abs(p9/M)+n));
dref_dot = vref;
dact_dot = p9/M;
Pin_d = (p3/Lw)*(uin);


ext(1) = uin;
ext(2) = 0;

%ds = [p3_dot; p9_dot];
ds = [p3_dot; p9_dot; dref_dot; dact_dot; Pin_d];
```