Lab 4

Rayne Milner
EME 171 - Mechatronics Systems

December 7, 2018

1 System

Following is the model and bond graph of the system.

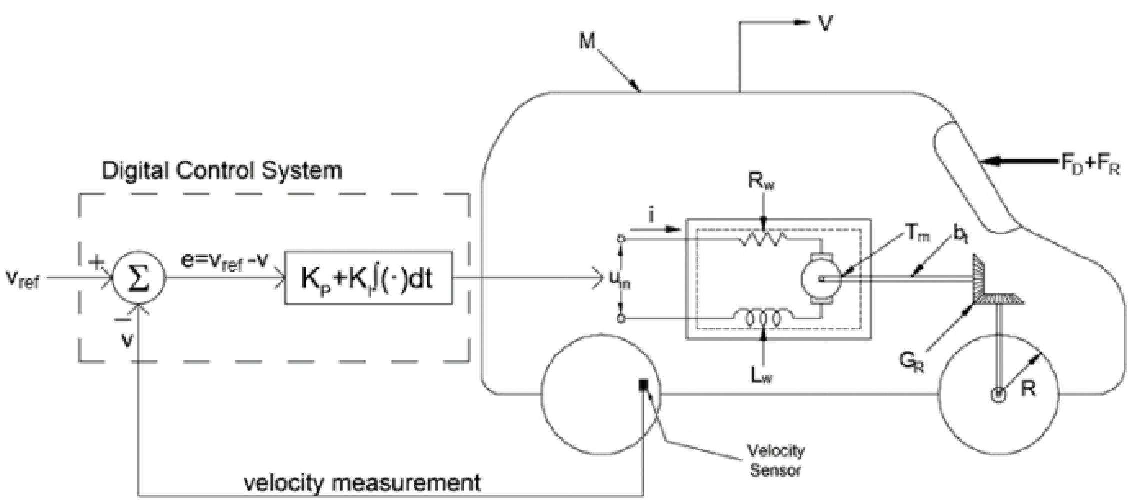

Figure 1: Full Model of System and Controller Block Diagram

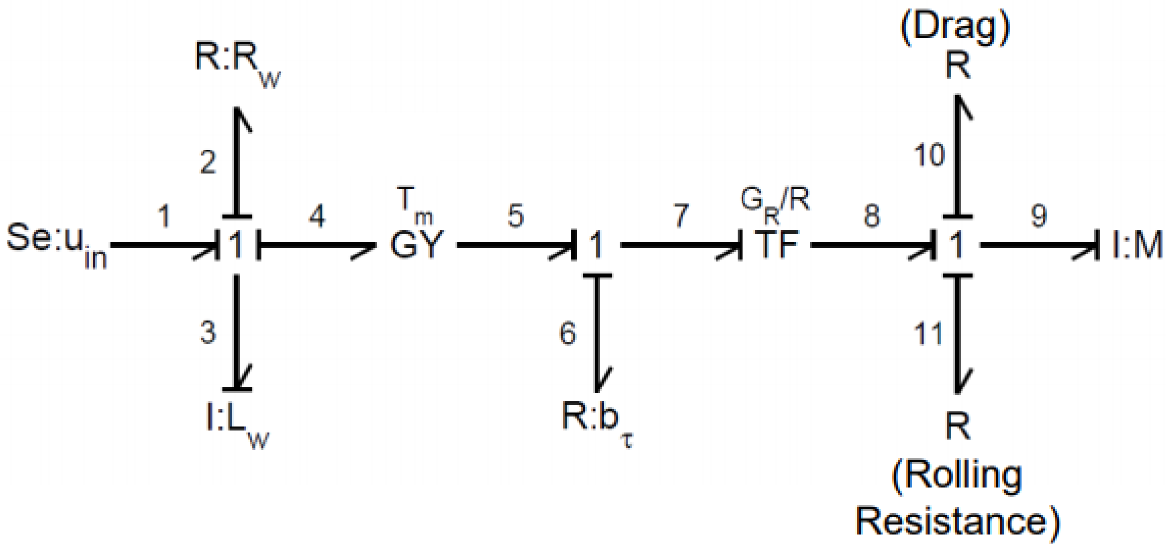

Figure 2: Bond Graph of System with Causality

1.1 State Equations

Following are the dynamic equations that describe the state variables for the system. Variables names are assigned from the bond graph. For example, P_3 refers to the flux linkage associated with bond three. Additional state variable have been added, they are the reference displacement and vehicle displacement, these quantities will be used in our automatic controller.

$$\dot{p_3} = u_{in} - \frac{R_w}{L_w}\,p_3 - \frac{T_m\,G_r}{R\,m}\,p_9$$

$$\dot{p_9} = \frac{G_r\,T_m}{R_w\,L_w}\,p_3 - \frac{(G_r)^2\,b_\tau}{(R_w)^2\,m}p_9 - m\,g\,C_r\,sign(p_9) - \frac{1}{2}\rho\,A_f\,C_d\,\frac{1}{m}\,p_9 * |\frac{p_9}{m}|$$

$$\dot{d_{ref}} = v_{ref}$$

$$\dot{d} = \frac{p_9}{M}$$

1.2 Initial Conditions

Initial conditions were determined by using the state equations. The initial state is defined by equilibrium, meaning the derivative of momentum values are zero. Other conditions that are determined: the initial displacement and reference displacement are equal to zero. The car is initially moving with some given speed that is set by the user (p$_9 = vi * M$). One non-trivial equation results, $p_3(0)$ can be found from this equation.

$$0 = \tfrac{G_r\,T_m}{R_w\,L_w}\,p_3 - \tfrac{(G_r)^2\,b_\tau}{(R_w)^2}20 - m\,g\,C_r\,sign(20) - \tfrac{1}{2}\rho\,A_f\,C_d\,\tfrac{1}{m}\,(20)^2$$

$$\longrightarrow p_3(0) = \tfrac{R_w\,L_w}{G_r\,T_m}[\tfrac{(G_r)^2\,b_\tau}{(R_w)^2}\,20 + m\,g\,C_r + \tfrac{1}{2}\,\rho\,A_f\,C_d\,(20)^2]$$

1.3 Inputs

The inputs to our system are one effort source U_{in} (the voltage that controls the electric motor) and one flow source V_{ref} which is the desired velocity of the car. The effort source is defined by our controller where K_p and K_i are controller constants that must be determined through trial and error. The flow source is determined from a velocity profile, the velocity is first set to be equal to one in order to tune the controller. Then the velocity is given by data collected by the California Air Resource Board which is graphed in the analysis section.

$$u_{in}(t) = K_p(V_{ref} - \frac{p_9}{M}) + K_i * (d_{ref} - d)$$

1.4 Time control Parameters

The end time and step size were given in problem statement. They are 2.5s and 0.01s respectively.

2 Analysis: Part 1

The implemented controller was designed to bring the car from a velocity of zero to a reference velocity of $1m/s$ with performance requirements as follows: rise time less than 0.5s, settling time less than 2.0s and overshoot less than 10%.

This was done by trial and error, tuning K_i and K_p until the conditions were met. K_i is a constant that modulates the control response to displacement error. This constant largely determines steady state error and when this is set to zero, the system struggles to reach the desired final velocity. K_p is a constant that modulates the control response to velocity error. K_p determines the speed of the response, if this constant is set to zero, the system is too under-damped and takes on the order of $10s$ to reach the desired value.

The final values of K_i and K_p were $K_i = 175$ and $K_p = 64$. This gave performance requirements: $t_r = 0.46s$, $t_w = 1.93s$ and $\%O.S. = 3.1\%$ The following chart shows the velocity of the car with these control parameters.

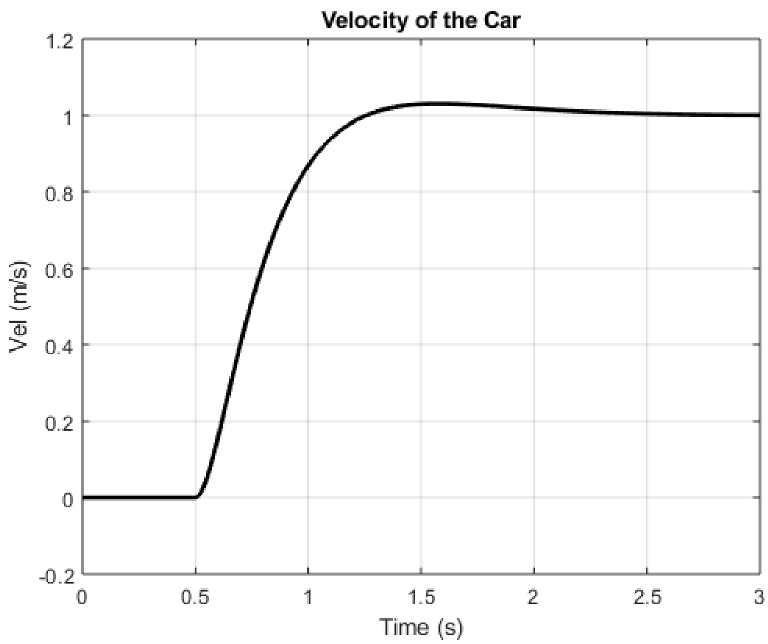

Figure 3: Plot of velocity with automatic control, $v_{ref} = 1$.

3 Analysis: Part 2

To simulate a realistic driving scenario, the V_{ref} is assigned by experimental data collected by the California Air Resource Board, known as "LA92 Dynamometer Driving Schedule". We see in figure 4 how well the vehicle follows the command velocity, the vehicle lags about 0.25s behind the command velocity and the vehicle overshoots velocity by less than a half of a percent. Additionally, The controller works well at both high and low speeds, with the lag and overshoot remaining the same under both conditions.

Average energy efficiency for this cycle is 43 watt hours per meter which was calculated by integrating the input energy over time and dividing that by total distance traveled. Assuming that one gallon of gasoline has about $1.2 \text{x} 10^8$ joules of energy, then this vehicle would get around 0.5 miles per gallon. This figure seems inaccurate because it is much lower than conventional vehicles. However, when the controls are tuned to be more strict or more lenient, it does not significantly affect the energy efficiency.

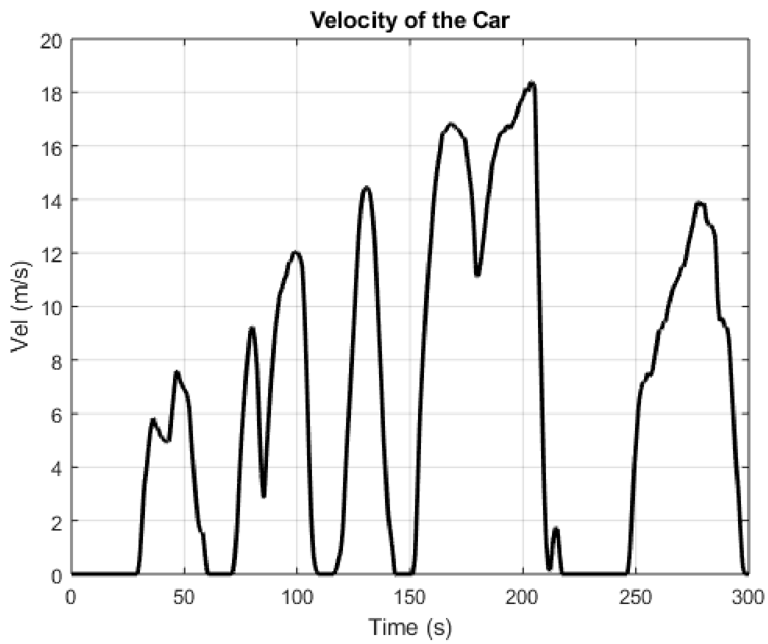

Figure 4: Plot of velocity with automatic control, v_{ref} given by LA92.

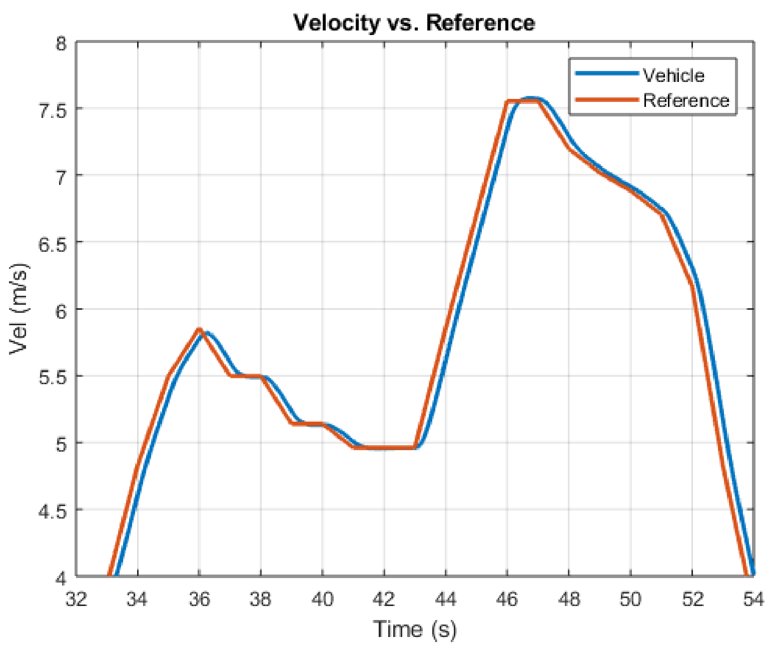

Figure 5: Plot of vehicle velocity versus command velocity, enlarged to show detail.

4 Analysis: Part 3

This system may be linearized and then analyzed through bode plot. Linearization gives the state space representation, which then can be used to compute the transfer function of this system $\frac{v}{u_{in}}(s)$. Matlab was used to find the transfer function from the state space matrices A, B, C and D.

$$\frac{v}{u_{in}}(s) = C\,(s\,I - A)^{-1}\,B + D$$
$$\frac{v}{u_{in}}(s) = \frac{1.302}{s^2 + 0.6809s + 55.91}$$

The Bode plot for this system is depicted in fig. 6. This system does not approach the 0dB threshold, and -180deg phase occurs when frequency goes to infinity. This system has infinite gain and phase margin and no changes to the system will significantly affect the stability.

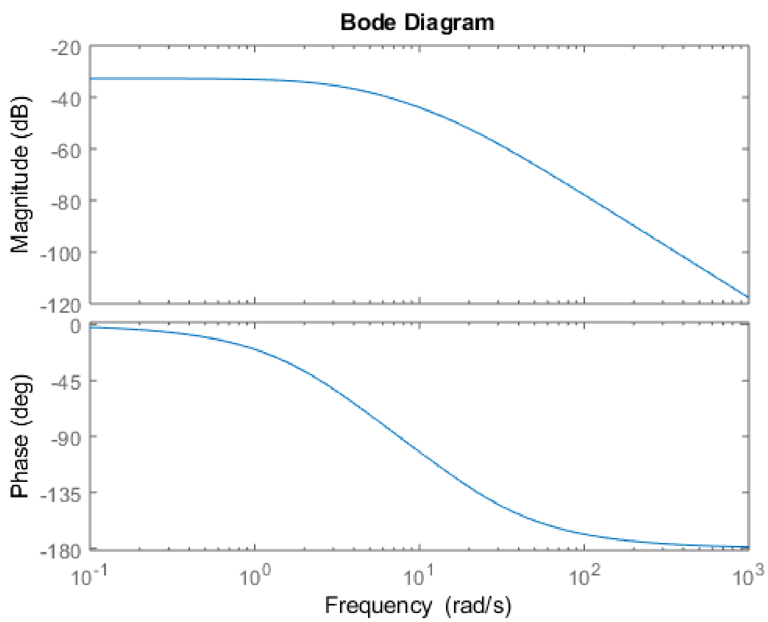

Figure 6: Bode plot of the system with nominal parameters.

Depicted in figure 7 is the Bode plot where winding resistance in the electric motor is increased by a factor of 5. This plot still has a phase margin of infinity, however, the gain plot now can cross the zero 0dB line. Additionally, we see the phase and gain curves both have a dip at the 10^0 frequency mark. This system has equivalent stability to the nominal system.

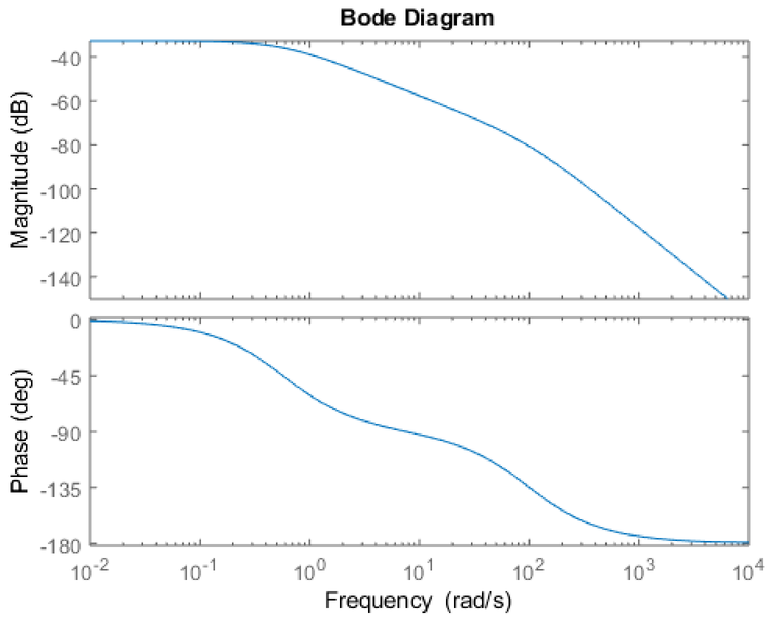

Figure 7: Plot of vehicle velocity versus command velocity, enlarged to show detail.

Finally, in figure 8, we have the case where winding resistance is decreased by a factor of 10. These plots have a significantly different shape from the previous figures. We see a peak in the gain plot and we see that the phase plot has a significantly steeper slope at the 10^1 frequency mark. This means that a phase lag or an increase in gain could bring the 0dB gain crossover close to the -180 deg phase, causing the system to become unstable.

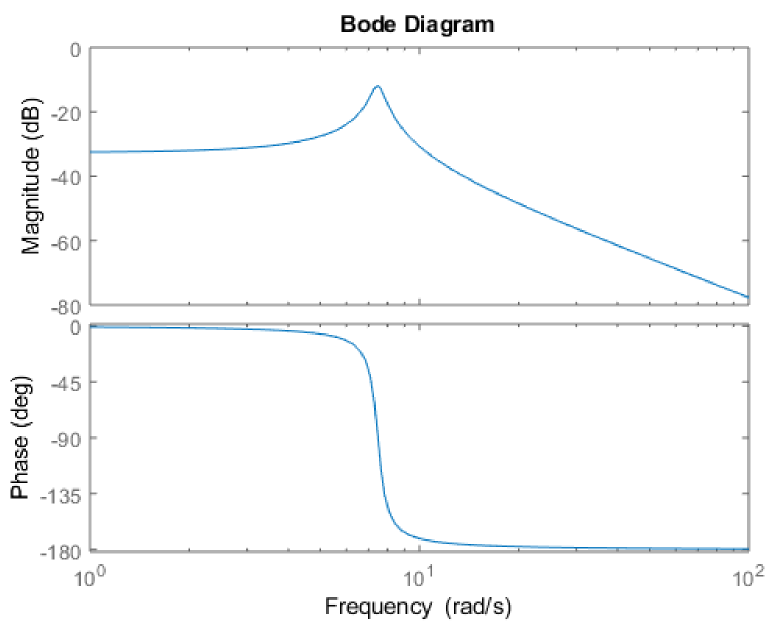

Figure 8: Plot of vehicle velocity versus command velocity, enlarged to show detail.

Lab 2 Master File

The purpose of the master file is to enter system parameters, call the differential equation solver, and finally post-process and plot results.

Contents

Clean Up Workspace

```
clear all
close all
clc
```

Input System Parameters

```
global Rw Lw Tm M bt R Gr Cr g Cd ro Af Uino Kp Ki vi

Rw = 0.01; %ohms, Armature winding resistance
Lw = 0.015; %H, Armature winding inductance
Tm = 1.718; % weber ,Transduction coefficient
M = 2200; %kg, Vehical mass
bt = 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,acceleration due to gravity
Cd = 0.32; %Drag Coefficient
ro = 1.21; %kg/m^3, Air density
Af = 2.05; %m^2, Vehical Frontal Area
Kp =200;
Ki = 250;
vi = 0;
```

Define Initial Conditions

The initial condition values for each system state variable are defined. Expressions for displacements are non-zero and determined by solvnig state equations when dP values are set to zero

```
 P3o = (Rw*Lw)/(Gr*Tm)*((Gr^2*bt)/Rw^2*vi+M*g*Cr+0.5*ro*Af*Cd*vi^2);
 P9o = vi*M;
 Uino = (Rw/Lw)*P3o+20*(Tm*Gr)/(R);

 initial = [P3o,P9o,0,0,0];
% [Flux linkage (P3), Momentum of car (P9), Reference displacment, vehical
% displacement, power]
```

Setup Time Array

The timestep was given in problem specification

```
tspan = 0:0.05:300;
```

Call ode45() Ordinary Differential Equation Solver

```
[t,s]=ode45(@lab5_sol_e,tspan,initial);

P3=s(:,1);  % Flux linkage
P9=s(:,2);  % Momentum of car
dref = s(:,3); %reference displacement
d = s(:,4); %displacement of the car
power = s(:,5); %power input
```

Obtain derivatives

```
for i=1:length(t)
    [ds] = lab5_sol_e(t(i),s(i,:));
end
```

Outputs

```
%This makes an array with the command velocity values
vref=zeros(length(t),1);
for i=1:length(t)
 vref(i) = LA92Oracle(t(i));
end

%These lines compute rise time, settling time and %O.S. for our
%unit step input
tr = t(max(find(P9 < M*0.9))) - t(max(find(P9 < M*0.1)));
ts = t(max(find(P9 > M*1.02 | P9 < M*.98)));
OS = max((P9/M)/vref);

%Calculates engery effiency for the LA92 cycle.
EE = max(power)/max(d);

%outputs plot of velocity
figure('Name','displacement','NumberTitle','off','Color','white')
plot(t,(P9/M),'k','LineWidth',2);grid on
title('Velocity of the Car')
ylabel('Vel (m/s)')
xlabel('Time (s)')

%outputs vehicle velocity against reference
figure('Name','displacement1','NumberTitle','off','Color','white')
plot(t,[(P9/M),vref],'LineWidth',2);grid on
title('Velocity vs. Reference')
legend('Vehicle','Reference')
ylabel('Vel (m/s)')
xlabel('Time (s)')
axis([32,54,4,8])
```

Bode Plots

```
%statespace matricies
A = [[-Rw/Lw, -(Tm*Gr)/(R*M)];[Tm*Gr/(R*Lw),-Gr^2*bt/(R^2*M)]];
B = [1;0];
C = [0,1/M];
D = [0];

%creating TF variables
s  = tf('s');
I = eye(2);

%Computing the trasnfer function
tfvu = C*(s*I - A)^-1*B+D;

%Plotting the Bode Diagram
bode(tfvu)
```

Contents


```
function [ds,ext] = lab5_sol_e(t,s)
```

input system parameters

```
global Rw Lw Tm M bt R Gr Cr g Cd ro Af Ki Kp
```

rename state variables

```
P3=s(1);  % Flux linkage
P9=s(2);  % Momentum of car
dref = s(3); %reference displacment
d = s(4); %vehical displacement
power = s(5);
```

```
Not enough input arguments.

Error in lab5_sol_e (line 8)
P3=s(1);  % Flux linkage
```

input specifications

Here we define the inputs into our system. These input sources are defined graphically in the problem statement. The graphical information has been translated into equations.

```
T1=0.5;           % s, controller start

if t<T1; vref=0;
else vref=LA92Oracle(t);
end

Uin = Kp*(vref-P9/M)+Ki*(dref-d);
```

equations of motion

These are the dynamic equations describing our state variables given in the form s'=f(s,inputs).

```
dP3 = Uin - (Rw/Lw)*P3-(Tm*Gr)/(R*M)*P9;
dP9 = (Gr*Tm)/(Rw*Lw)*P3 - (Gr^2*bt)/(Rw^2*M)*P9 - M*g*Cr*((P9/M)/(abs((P9/M))+0.000001))-0.5*ro*Af*Cd*(1/M)*P9*abs(P9/M);
ddref = vref;
dd =  P9/M;
dpower = Uin;
```

stacking up the derivatives for output as a vector

```
ds=[dP3;dP9;ddref;dd;dpower];
```

```
end
```

Published with MATLAB® R2018b