Linear Control Systems

(EE-379)

DE-43 Mechatronics

Syndicate – A

Assignment 1

Name and CMS ID:

1. NC Syed Muhammad Daniyal Gillani Reg# 375785

Submitted to: Dr. Anjum Naeem Malik

Analysis of 2nd Order System

Proposed system:

Vehicle Suspension System

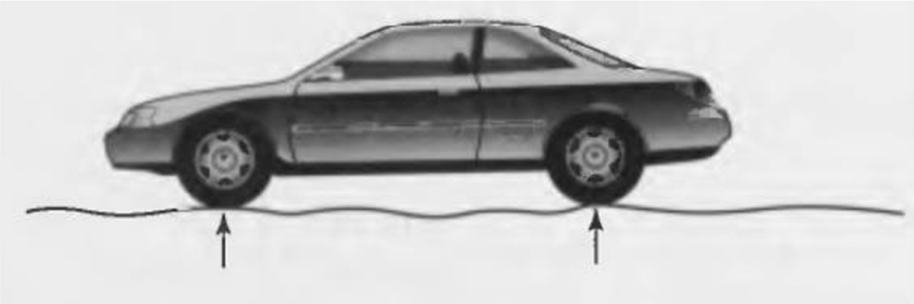

Figure 1: Vehicle (copied from book)

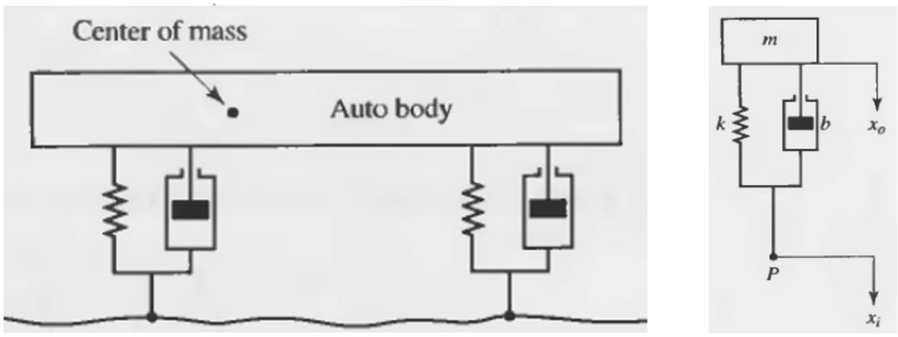

Figure 2: Graphical model of overall system (left) Graphical model of 1/4th side (right)

For the analysis only one quarter of vehicle suspension will be taken under consideration

Mathematical modelling:

All values have been taken from a research paper [1].

1. ***Governing Differential Equation***:

$$m * \ddot{x_o} + b * \dot{x_o} + k * x_o = b * \dot{x_i} + k * x_i$$

2. ***LaplaceTransform***:

$$m * s^2 * X_o(s) + b * s * X_o(s) + k * X_o(s) = b * s * X_i(s) + k * X_i(s)$$

3. ***TransferFunction***:

$$G(s) = X_o(s)/X_i(s) = (b * s + k)/(m * s^2 + b * s + k)$$

4. ***Proposed Parameters***:

$m = 500kg\ (quarter\ car\ body\ mass)$

$k = 30000\ N/m(spring\ constant)$

$b = 3000\ N \cdot s/m(damping\ coefficient)$

5. $\boldsymbol{Specific\ Transfer\ Function}$:

$$G(s) = (3000 * s + 30000)/(500 * s^2 + 3000 * s + 30000)$$

6. $\boldsymbol{NaturalFrequency(\omega_n)}$:

$$\omega_n = \sqrt{(k/m)} = \sqrt{(30000/500)} = 7.75 rad/s$$

7. $\boldsymbol{DampingRatio(\zeta)}$:

$$\zeta = b/\left(2 * \sqrt{(k * m)}\right) = 3000/\left(2 * \sqrt{(30000 * 500)}\right) = 0.387$$

8. $\boldsymbol{Damped\ Natural\ Frequency(\omega_d)}$:

$$\omega_d = \omega_n * \sqrt{(1 - \zeta^2)} = 7.455 rad/s$$

9. $\boldsymbol{Time - DomainSpecifications}$:

$\boldsymbol{RiseTime(t_r)}$: $t_r \approx tr = \tan^{-1}(-\frac{\sqrt{1-\zeta^2}}{\zeta})/(\sqrt{1-\zeta^2} \cdot \omega_n) = 1.64s$

$\boldsymbol{PeakTime(t_p)}$: $t_p = \pi/(\omega_n * \sqrt{1-\zeta^2}) = 0.476s$

$\boldsymbol{Overshoot(M_p)}$: $M_p = e^{\left(-\zeta * \pi/\sqrt{(1-\zeta^2)}\right)} * 100 \approx 23.5\%$

$\boldsymbol{SettlingTime(t_s)}$: $t_s = 4/(\zeta * \omega_n) = 4/(0.387 * 7.75) = 1.37s$

10. $\boldsymbol{Characteristic\ Equation}$:

$$500 * s^2 + 3000 * s + 30000 = 0$$

MATLAB analysis:

Code given in appendix A

Time domain characteristics:

Rise Time: 0.1282

Transient Time: 1.3114

Settling Time: 1.3114

Overshoot: 37.2784

Peak Time: 0.3224

```
Transfer Function:

suspension_sys =

       3000 s + 30000
  -----------------------
  500 s^2 + 3000 s + 30000

Continuous-time transfer function.
Model Properties
Time-Domain Response Characteristics:
         RiseTime: 0.1282
    TransientTime: 1.3114
     SettlingTime: 1.3114
      SettlingMin: 0.9003
      SettlingMax: 1.3728
        Overshoot: 37.2784
       Undershoot: 0
             Peak: 1.3728
         PeakTime: 0.3224
```

Stability analysis:

The roots of the system after calculating give us:

-3.0000 + 7.1414i

-3.0000 - 7.1414i

Both the roots lie in the left half plane thus the system is ***stable system and the condition for BIBO stability is also satisfied.*** having an ***underdamped response*** due to roots being ***complex conjugate pairs*** having real and imaginary parts.

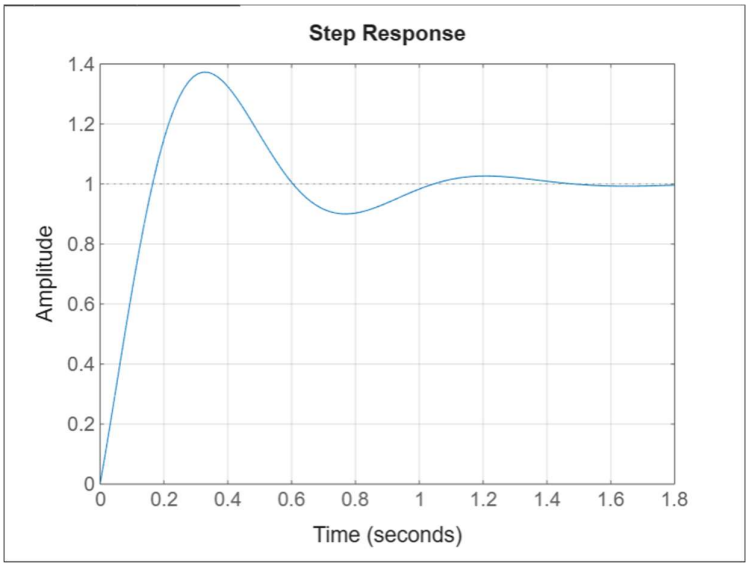

Figure 2-a: Time response of system

Routh Hurwitz Table (Used an online generator [2])**:**

Routh-Hurwitz Table

s^2	500	30000
s	3000	0
1	30000	0

Disturbance analysis:

Three types of disturbance functions were input to the system. The functions and their code were generated via ChatGPT whereas the values and necessary information was taken from research paper [3]. The response of the system to these disturbances is shown as under:

1. Response due to bumpy roads/road vibrations:

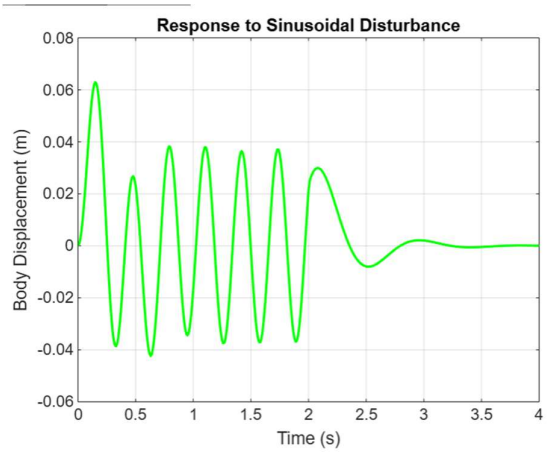

Figure 2: Response due to sinusoidal disturbance input given to the system for 2 seconds.

2. Response due to speed breakers:

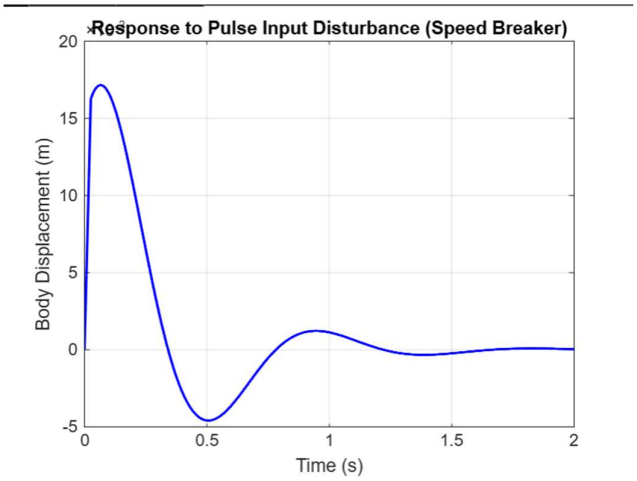

Figure 3: Response due to speed breakers

Response due to potholes

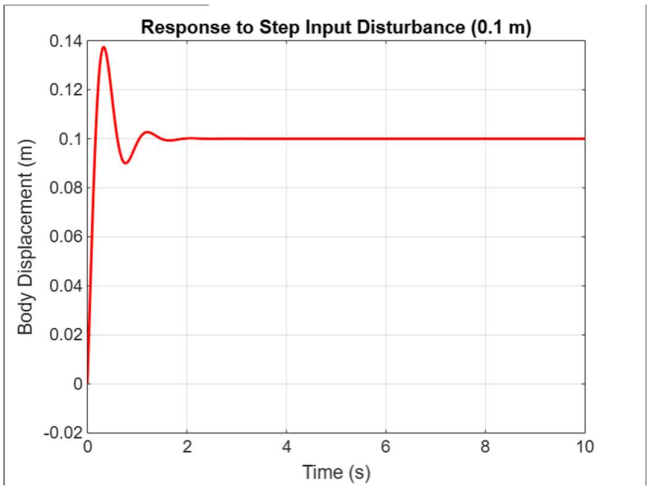

Figure 4: Response due to potholes.

The plots show the output response of the system due to disturbances and as can be seen, due to the system being stable, the disturbances die out after some time depicting the damping capabilities of the suspension system.

Root Locus Analysis:

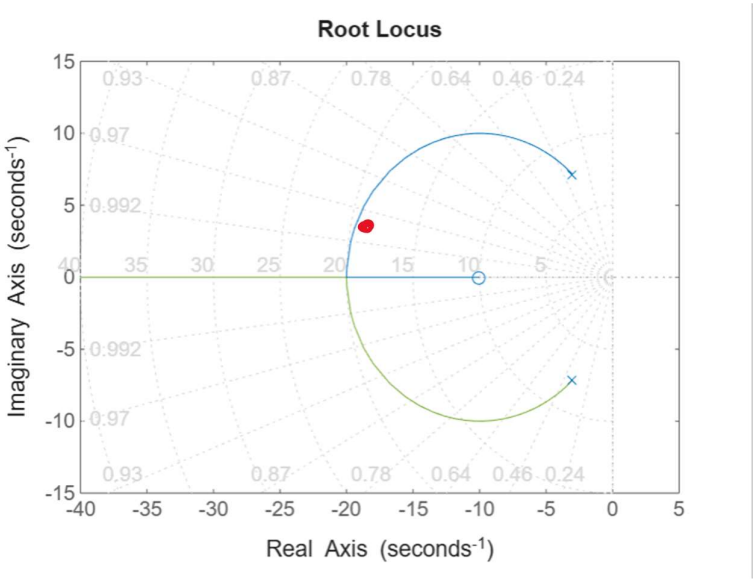

Figure 5: Root locus plot

The curve depicts possible root locations for the system based on different values of gain K or variations in system parameters. The set of straight lines moving outward from the origin depict different values of damping ratios. With the help of root locus plot, we can change system parameters to meet necessary overshoot and damping requirements while ensuring system stability. This can be done by keeping the roots at the intersecting point of the curve

and the required damping ratio's line for example, to have a damping ratio of 0.97 our roots must lie at roughly -18+6i and -18-6i as shown by the red dot in above figure.

References:

[1] A. Mohammadzadeh, "2006-940: ANALYSIS AND DESIGN OF VEHICLE SUSPENSION SYSTEM USING MATLAB AND SIMULINK." Accessed: Dec. 01, 2024. [Online]. Available: https://peer.asee.org/analysis-and-design-of-vehicle-suspension-system-using-matlab-and-simulink.pdf

https://www.mathworks.com/help/ident/ref/dynamicsystem.lsim.html

[2] *Streamlit.app*, 2024. https://routhhurwitz.streamlit.app/#routh-hurwitz-table (accessed Dec. 01, 2024).

[3] S. Vaishnav, J. Paul, and R. Deivanathan, "Model development and simulation of vehicle suspension system with magneto-rheological damper," *IOP Conference Series: Earth and Environmental Science*, vol. 850, no. 1, p. 012035, Nov. 2021, doi: https://doi.org/10.1088/1755-1315/850/1/012035.

Appendix:

Appendix A:

```
close all
% MATLAB Code for Stability and Time-Domain Analysis of a Second-Order System
% System parameters
m = 500;       % Mass (kg)
b = 3000;      % Damping coefficient (N·s/m)
k = 30000;     % Spring stiffness (N/m)
% Transfer function: G(s) = Xo(s) / Xi(s)
num = [b k];                % Numerator
den = [m, b, k];            % Denominator
suspension_sys = tf(num, den);          % Transfer function representation
% Display the transfer function
disp('Transfer Function:');
suspension_sys
```

```
% Time-Domain Analysis
info = stepinfo(suspension_sys);            % Step response characteristics
disp('Time-Domain Response Characteristics:');
disp(info);
% Plot step response
figure;
step(suspension_sys);
title('Step Response');
grid on;
% Stability Analysis using Poles
poles = pole(suspension_sys);               % Calculate poles of the system
disp('Poles of the system:');
disp(poles);

% Routh-Hurwitz Criterion
disp('Routh-Hurwitz Stability Check:');
disp(['Poles are all in the left-half plane, so the system is stable and underdamped.']);

% Root Locus Analysis
figure;
rlocus(suspension_sys);
title('Root Locus');
grid on;

% Disturbance Analysis
% 1. Step Input Disturbance (e.g., a road bump)
t_step = 0:0.01:10;             % Time vector for step input
step_magnitude = 0.1;           % Step input magnitude (m)
step_input = step_magnitude * ones(size(t_step)); % Step input
```

```
[y_step, t_out_step] = lsim(suspension_sys, step_input, t_step); % Response


figure;
plot(t_out_step, y_step, 'r', 'LineWidth', 1.5);
title('Response to Step Input Disturbance (0.1 m)');
xlabel('Time (s)');
ylabel('Body Displacement (m)');
grid on;

% 2. Pulse Input Disturbance (e.g., speed breaker)
t_pulse = 0:0.001:2;            % Time vector for pulse input
pulse_magnitude = 0.1;          % Pulse input magnitude (m)
pulse_duration = 0.025;         % Pulse duration (25 ms)
pulse_input = (t_pulse <= pulse_duration) * pulse_magnitude; % Define pulse input
[y_pulse, t_out_pulse] = lsim(suspension_sys, pulse_input, t_pulse); % Response
figure;
plot(t_out_pulse, y_pulse, 'b', 'LineWidth', 1.5);
title('Response to Pulse Input Disturbance (Speed Breaker)');
xlabel('Time (s)');
ylabel('Body Displacement (m)');
grid on;

% 3. Sinusoidal Disturbance (e.g., road vibration)
t_sine = 0:0.01:4;                  % Time vector for sinusoidal input
sin_magnitude = 0.1;                % Magnitude of sinusoidal input
frequency = 20;                     % Frequency of sinusoidal input (rad/s)
sin_disturbance = sin_magnitude * sin(frequency * t_sine) .* (t_sine <= 2); % Sinusoidal input
active for t <= 2s
[y_sine, t_out_sine] = lsim(suspension_sys, sin_disturbance, t_sine); % Response
figure;
```

```
plot(t_out_sine, y_sine, 'g', 'LineWidth', 1.5);
title('Response to Sinusoidal Disturbance');
xlabel('Time (s)');
ylabel('Body Displacement (m)');
grid on;

% BIBO Stability (Impulse Response)
figure;
impulse(suspension_sys);
title('Impulse Response (BIBO Stability Check)');
grid on;
disp('Analysis complete.');
```