1/ Introduction

In this lab, I will perform engineering computation on MATLAB to numerically integrate the differential equations generated from a bond graph model of a system. This process of integrating the time dependent equations is called "simulation" and it is a powerful method to analyze linear and non-linear systems. The system that is simulated in this lab is a simple model of a four-wheeled vehicle suspension which consists of 3 components: mass, spring, and damper. The assumptions that are made in this analysis are: the spring and damper elements behave linearly, the wheel mass and the stiffness are not represented, the effect of force due to gravity is eliminated.

The goal of this lab is to convert the given differential equations into a computer function that evaluates the equations at any given point in time and numerically integrate differential equations with MATLAB ode 45.

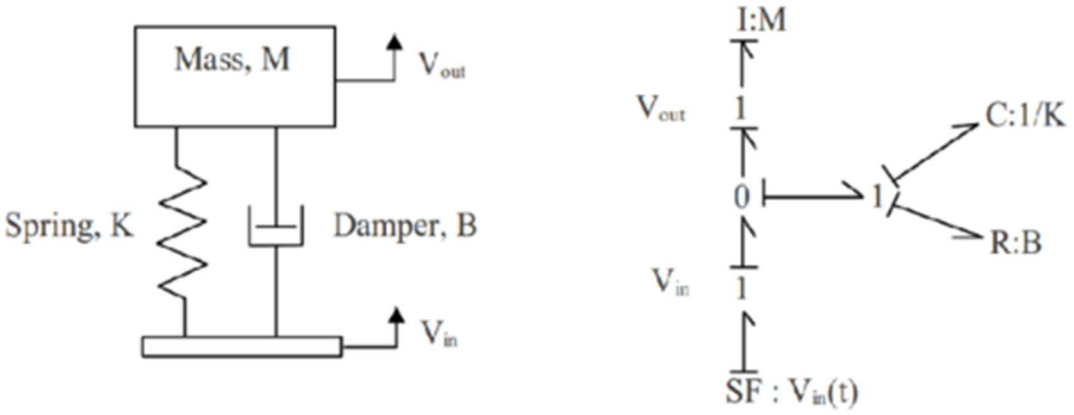


Fig 1: Mechanical Schematic of Quarter Car Model (left) and equivalent bond graph (right)

The state equations for the linearized system are:

$$\dot{p}(t) = Kq(t) + B\left(v_{in}(t) - \frac{p(t)}{M}\right)$$

$$\dot{q}(t) = v_{in}(t) - \frac{p(t)}{M}$$

$$\dot{y}(t) = v_{in}(t)$$

There are 3 first order ODEs to describe the dynamics of the system change with respect to time. These equations are given in explicit form.

Momentum of the sprung mass: p(t)

Change in displacement between the sprung mass and the ground: q(t)

Vertical displacement of the road: $y_{in}(t)$

2/ Calculations

This system has three state variables, so there are three initial conditions. For this lab, all the initial conditions are zero.

3/ Systems Parameter

This system has 3 constant parameters:

Quarter car mass: M = 267 kg

Linear shock absorber damping coefficient: B = 1398 Nsm^{-1}

Linear spring stiffness: K = 1.87 × 104 Nm^{-1}

4/ Time Parameter

Natural Frequency $\omega_n = \sqrt{\frac{K}{M}}$

Natural Frequency $f_n = \frac{\omega_n}{2\pi}$

Damping ratio $\xi = \frac{B}{2\sqrt{MK}}$

To show 5 oscillations on the graph after the bump finishes

$$t_{final} = \frac{5}{f_n} + 2$$

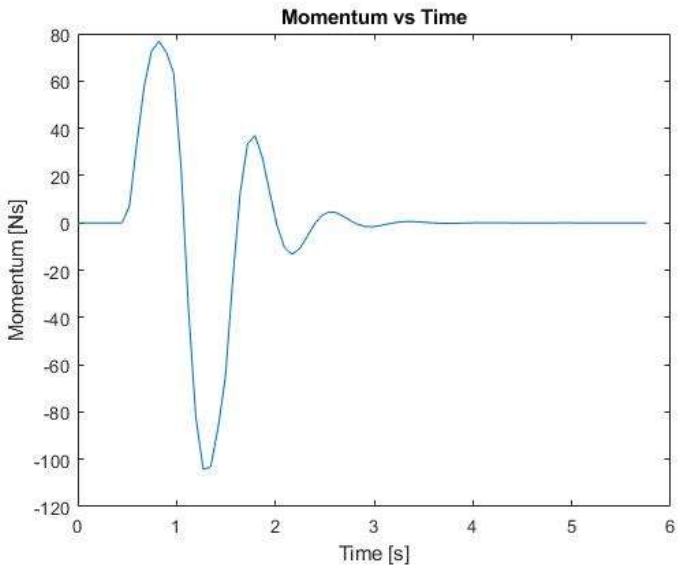
For 10 steps period

$$\mathrm{dt} = \frac{1}{10 * f_n}$$

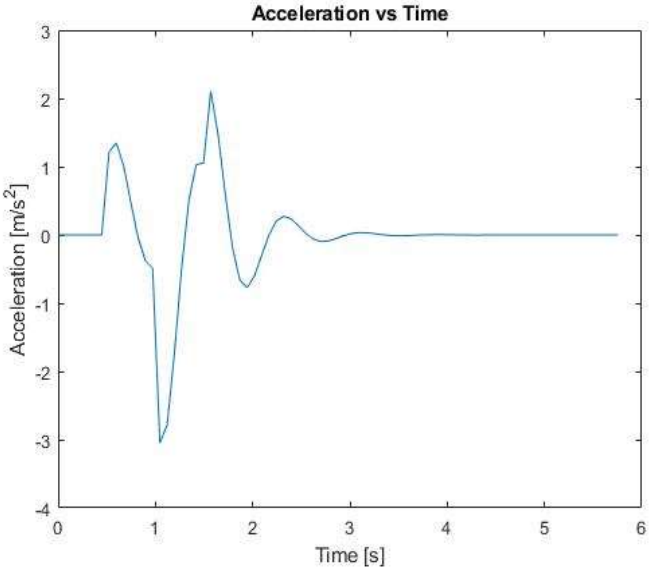
Number of steps

$$N_{step} = round\left(\frac{t_{final}}{dt}\right) + 1$$

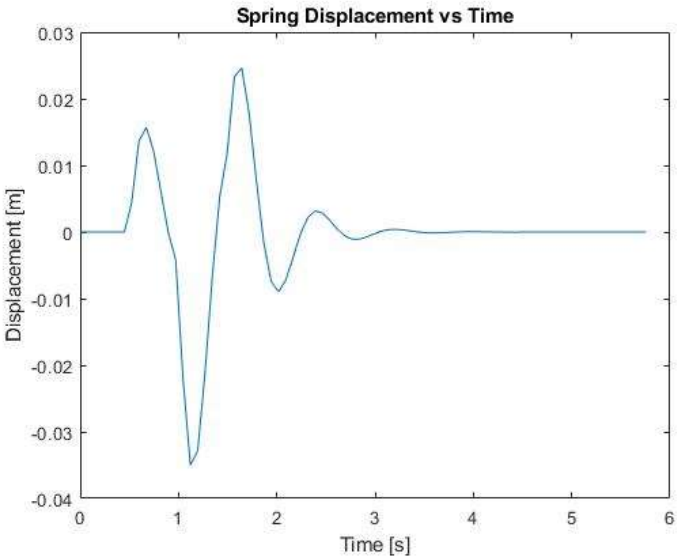
4/ Simulations



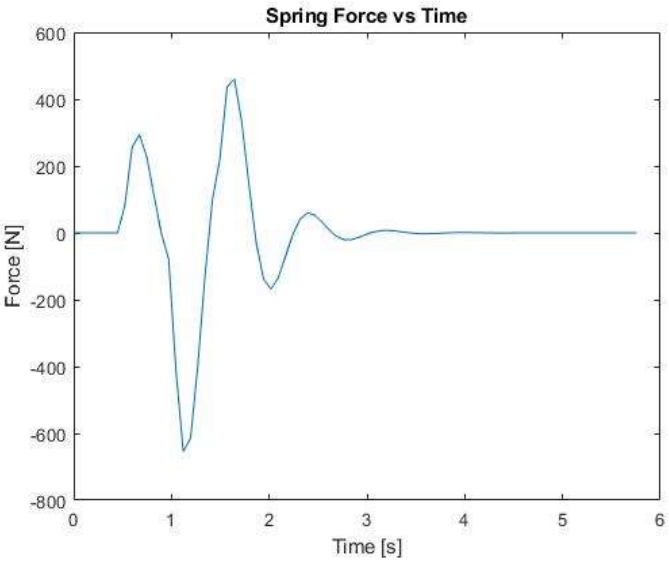
2a/ Momentum vs time graph



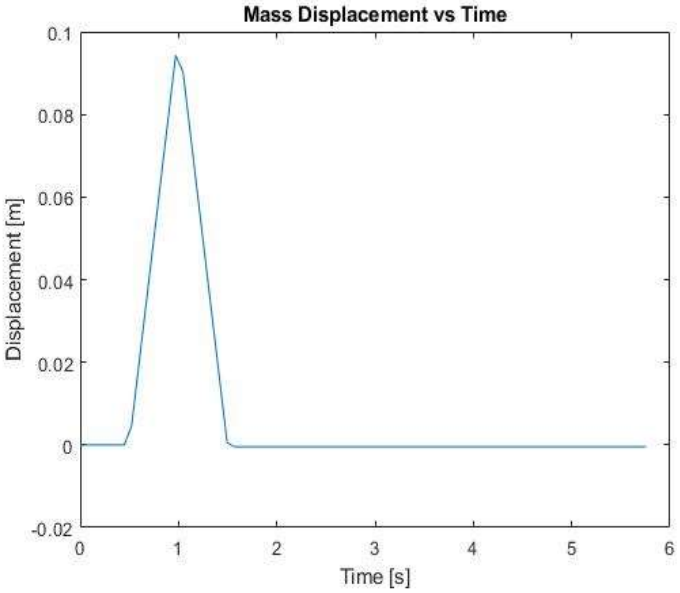
2b/ Acceleration vs time graph



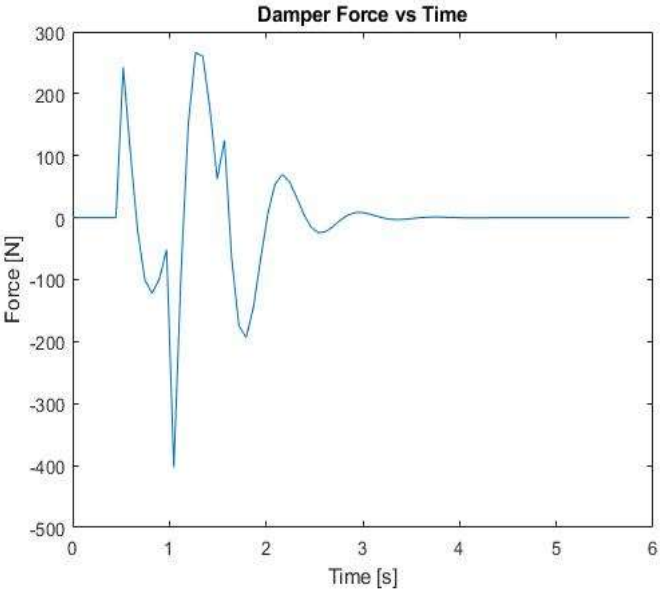
2c/ Spring displacement vs time graph



2d/ Spring force vs time graph

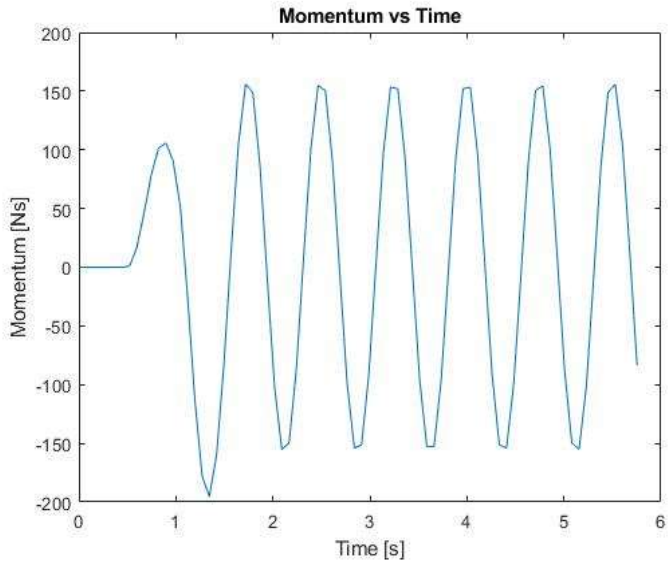


2e/ Mass displacement vs time graph

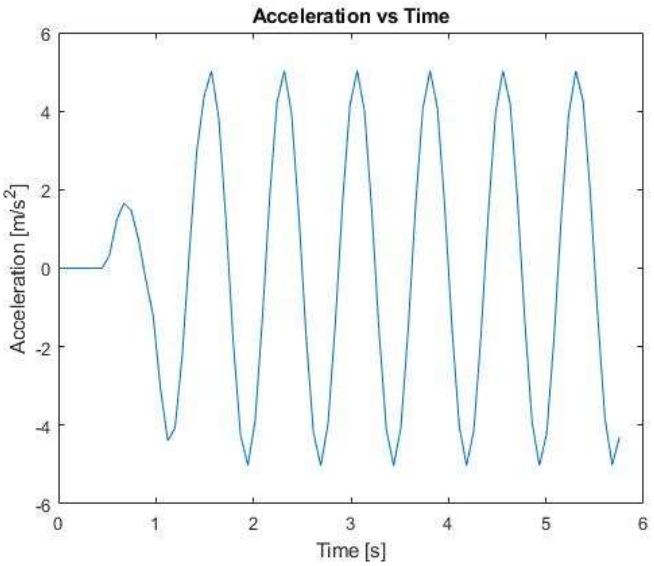


2f/ Damper force vs time graph

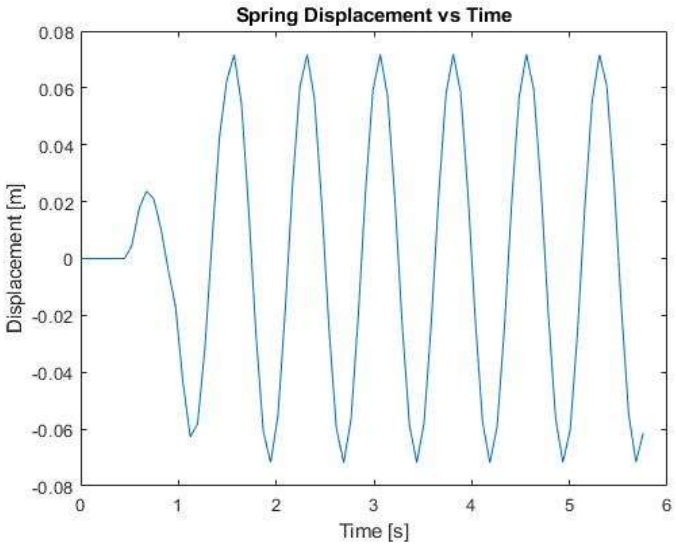
Fig 2 a-f: Suspension system traversing the triangle bump in the road



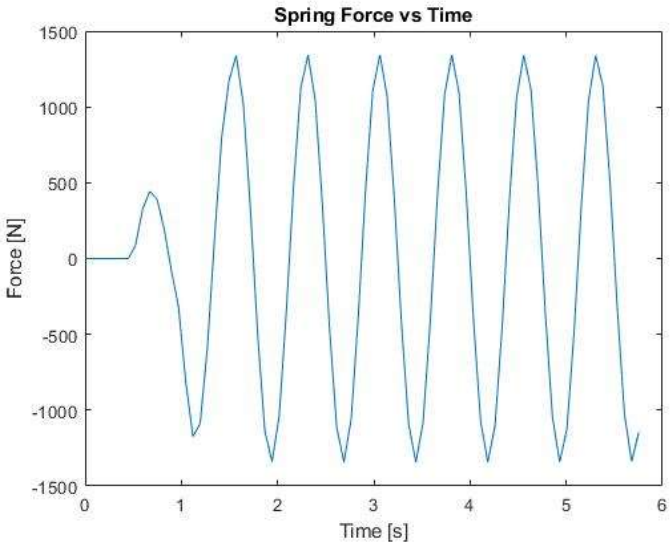
3a/ Momentum vs time graph



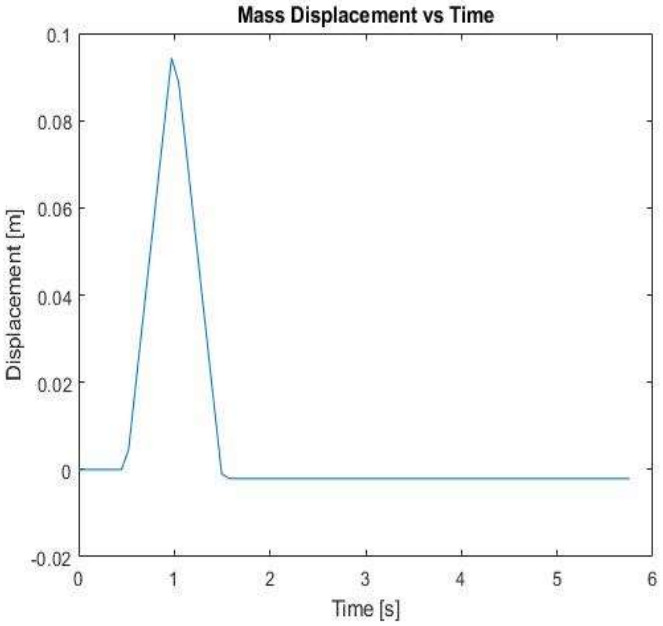
3b/ Acceleration vs time graph



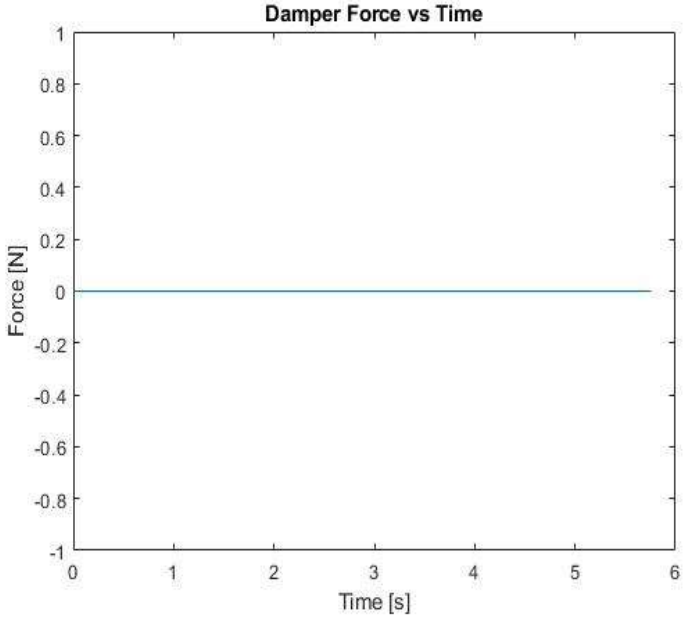
3c/ Spring displacement vs time graph



3d/ Spring force vs time graph



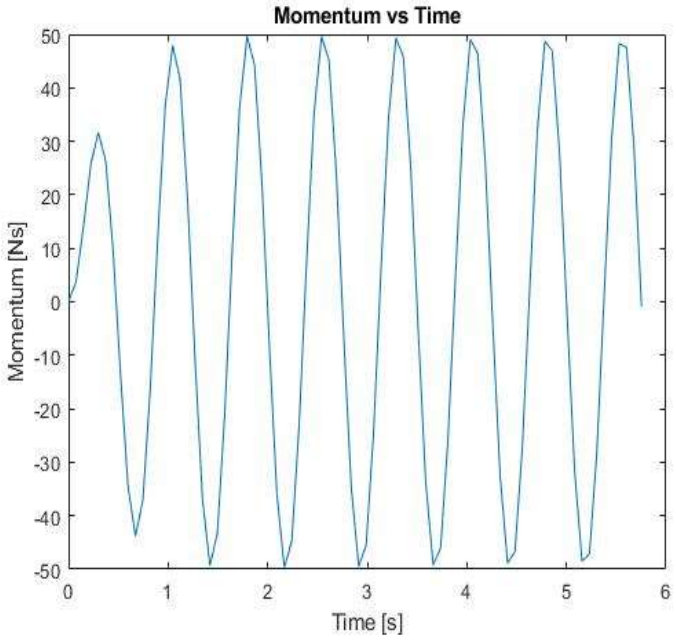
3e/ Mass displacement vs time graph



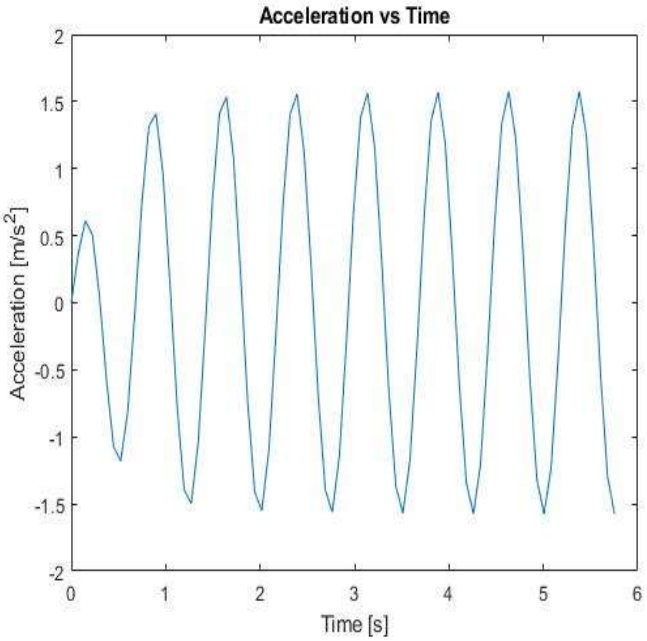
3f/ Damper force vs time graph

Fig 3 a-f: Effect of removing the damper from the suspension system moving on triangle bump in the road

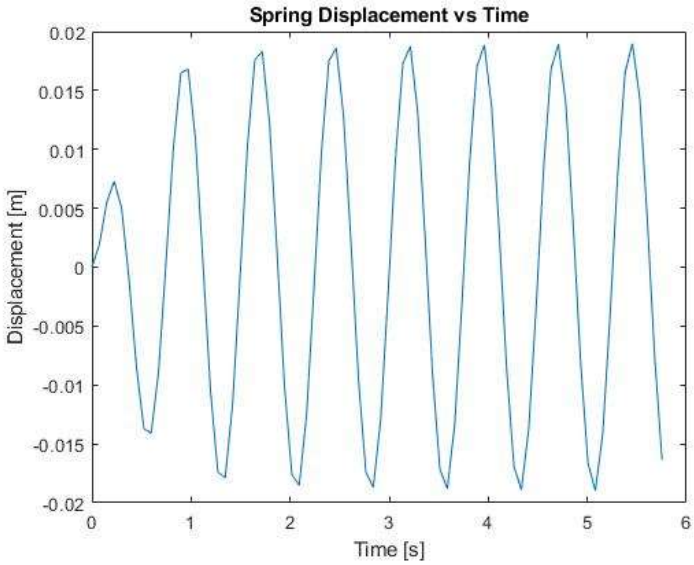
Explanation of no damping: In comparison between the system with damper in figure 2 a-f and the system without damper in figure 3 a-f, there are differences in the momentum, spring displacement, spring force, damping force, and acceleration. In the system with damper, the momentum, spring displacement, acceleration, and spring force oscillates intensively to their peak values when the car hits the road bump, but the oscillations quickly die out and approach zero steady-state value. This means the vibrations after hitting the road bump are eliminated by the damper. While in the system with no damper, the momentum and spring displacement oscillate in a sinusoidal shape after the car passes the bump on the road, which means the car still vibrates consistently after passing the road bump. In the system with damper, the damper absorbs the vibration from the car when it passes the bump, this action can be seen in the oscillation in figure 2f. In the opposite, in the system with no damper, the damping force stays constant at zero at all time in figure 3f. From these differences, it can be showed that the removal of the damper has negative effect on the motion of the car.



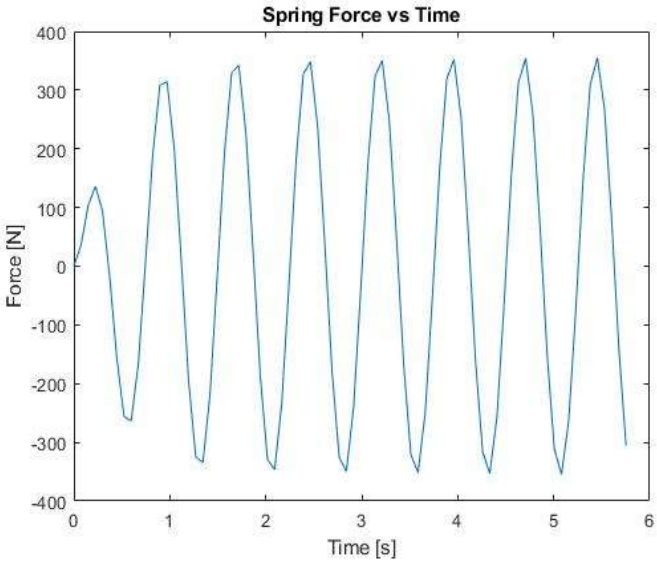
4a/ Momentum vs time graph



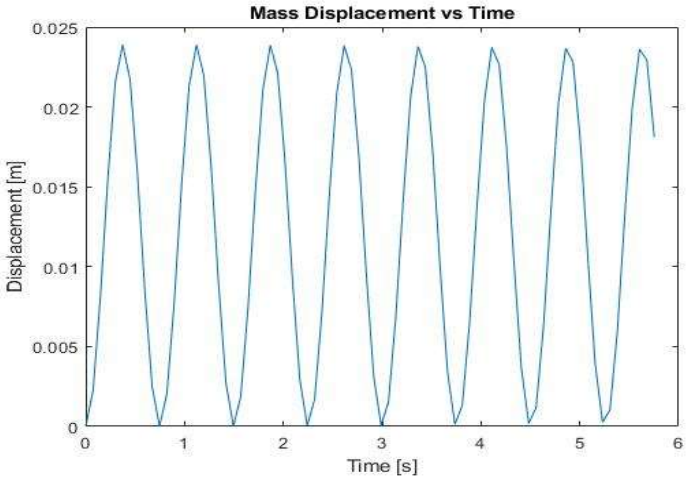
4b/ Acceleration vs time graph



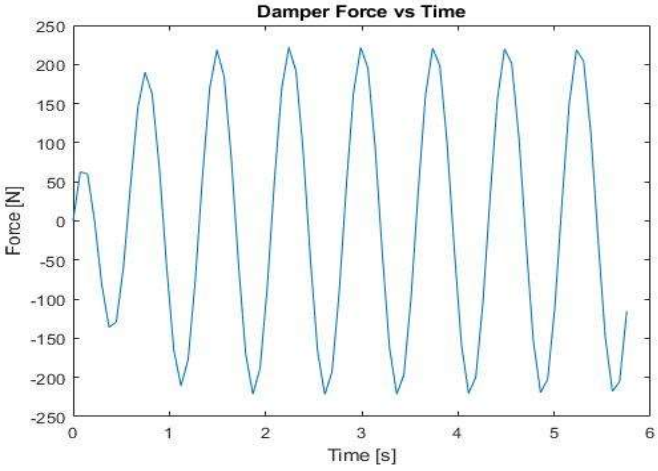
4c/ Spring displacement vs time graph



4d/ Spring force vs time graph



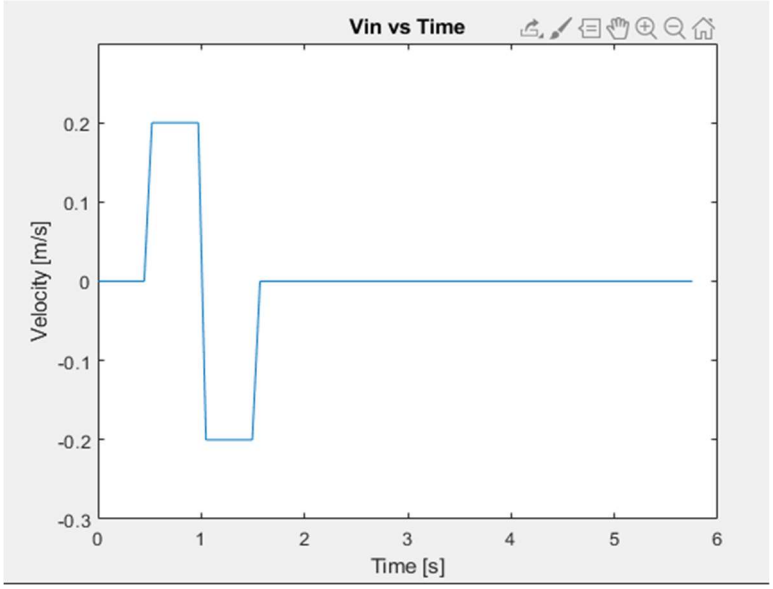
4e/ Mass displacement vs time graph



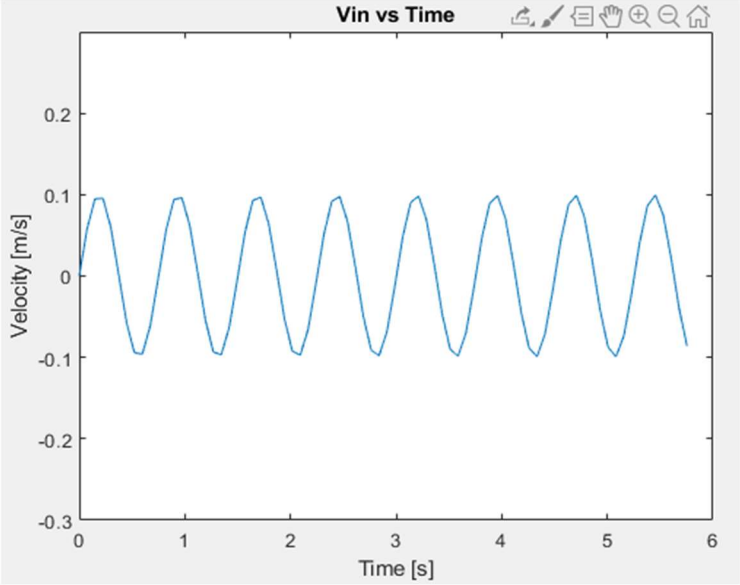
4f/ Damper force vs time graph

Fig 4 a-f: Suspension system going through a sinusoidal undulating road input with velocity amplitude of 0.1 m/s.

Explanation of second road input behavior: in figure 4 a-f, the road input was changed from a triangle bump into a sinusoidal undulating input with velocity amplitude of 0.1 m/s. With this change, the momentum, spring displacement, acceleration, spring force, and damper force oscillate sinusoidally with respect to time. This is also what I expect in real life when the car goes through a sinusoidal shape track, the center of mass of the car and suspension system will vibrate sinusoidally. The damper force will also oscillate in a sinusoidal shape to counter the sinusoidal vibration of the car.



5a/ Velocity Input for triangular road bump terrain



5b/ Velocity input for sinusoidal undulating road input

Explanation of velocity input trend: in figure 5 a-b, the road input was changed from a triangle bump into a sinusoidal undulating input with velocity amplitude of 0.1 m/s. With this change, the input velocity curve changed from rectangle step input shape to sinusoidal shape. This is also what I expect in reality. When the car moves through different terrains, its input velocity would also be different.

5/ Code

eval_quarter_car_rhs.m – Evaluate the right hand side

```
function xdot = eval_quarter_car_rhs(t, x, w, c)
% EVAL_QUARTER_CAR_RHS - Returns the time derivative of the states, i.e.
% evaluates the right hand side of the explicit ordinary differential
% equations for the 1 DoF quarter car model.
%
% Syntax: xdot = eval_quarter_car_rhs(t, x, w, c)
%
% Inputs:
%   t - Scalar value of time, size 1x1.
%   x - State vector at time t, size 3x1, [p, q, yin]
%   w - Anonymous function, w(t, x, c), that returns the input vector
%       at time t, size 1x1.
%   c - Constant parameter structure with 4 items: K, B, M, A.
% Outputs:
%   xdot - Time derivative of the states at time t, size 3x1.

% unpack the states into useful variable names
% replace the question marks with useful variable names
p = x(1);   % momentum of spring mass
q = x(2);   % change in displacement betweeen the spring mass
y = x(3);   % vertical displacement of the road
```

```
% evaluate the input function
r = w(t, x, c);

% unpack the inputs into useful variable names
vin = r(1);
% unpack the parameters into useful variable names
m = c.m;    % mass
k = c.k;    % spring const
b = c.b;    % damping coeff

% calculate the derivatives of the state variables
pdot = k*q + b*(vin - (p/m));   % eqn 1
qdot = vin - (p/m);             % eqn 2
ydot = vin;                     % eqn 3

% pack the state derivatives into an mx1 vector
xdot = [pdot; qdot; ydot];
end
```

eval_triangle_bump.m – Generate the bump in the road

```
function r = eval_triangular_bump(t, x, c)
% EVAL_TRIANGULAR_BUMP - Returns the road bump velocity at any given time.
%
% Syntax: r = eval_triangular_bump(t, x, c)
%
% Inputs:
%   t - A scalar value of time, size 1x1.
%   x - State vector at time t, size 3x1 where the states are [mass momentum,
%       spring deflection, and road height].
%   c - Constant parameter structure with 4 items: k, b, m, A
% Outputs:
%   r - Input vector at time t, size 1x1, [vin].

% unpack parameter structure p
A = c.A;    % amplitude of step

% define a triangular road bump input by transcribing the input mathematical
% equation into equivalent code to calculate vin at any given time
if 0.5 <= t && t <1
    vin = A;
elseif 1 <= t && t < 1.5
    vin = -A;
else
    vin = 0;
end

% pack the input into an 1x1 vector
r = [vin];
end
```