

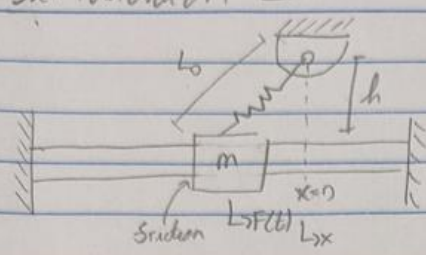
# Dynamic Systems and Controls

Kieran Cosgrove

A)

Kieran Cosgrove

## Simulation 1



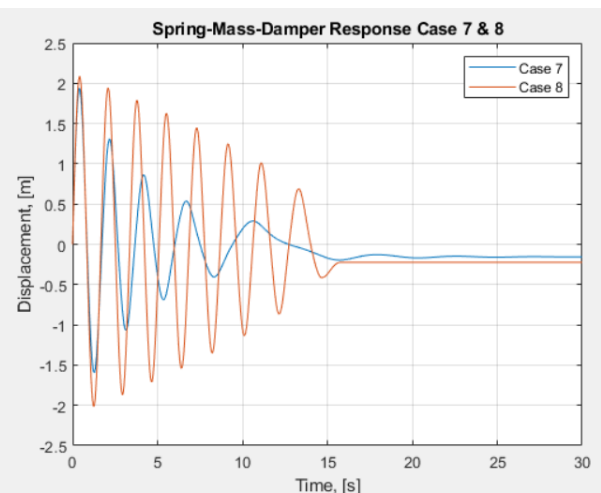
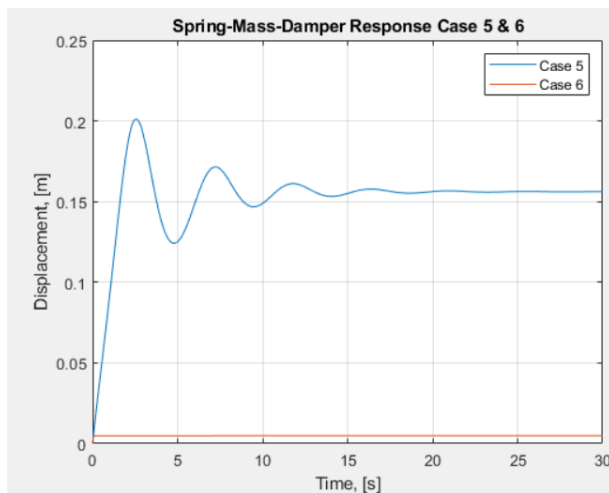
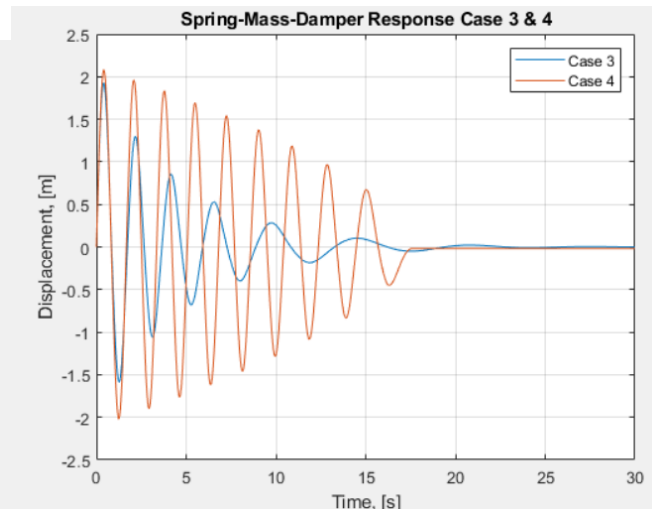
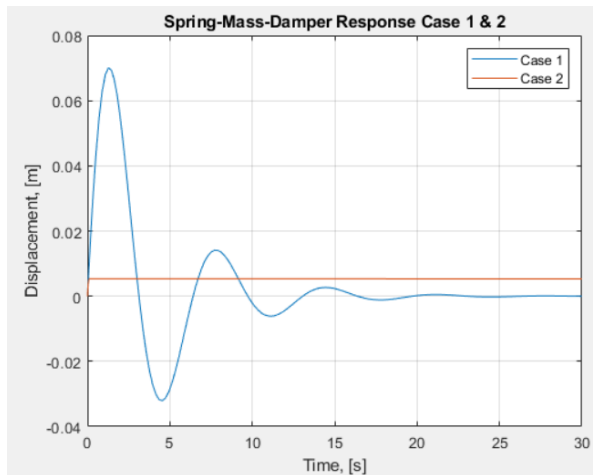
$f_1 = \frac{p_1}{I_1}$   
 $e_3 = R_3 f_1$   
 $e_0 = S'_E$   
 $* e_2 = \tilde{m}(\theta) e_4$   
 $e_4 = q_4 / q_4$   
 $* f_1 \tilde{m}(\theta) = f_1$

$\dot{q}_4 = f_4 = f_1 \tilde{m}(\theta) = \frac{p_1}{I_1} \tilde{m}(\theta)$   
 $p_1 = e_1 = e_0 - e_3 - e_2 = S'_E - R_3 f_1 - \tilde{m}(\theta) e_4 = S'_E - R_3 \frac{p_1}{I_1} - \tilde{m}(\theta) \frac{q_4}{q_4}$   
 $x^2 + h^2 = (L_0 - q_4)^2 \quad \frac{d}{dt}(x^2 + h^2) = \frac{d}{dt}(L_0 - q_4)^2$   
 $2\dot{x}x = -2q_4(L_0 - q_4)$   
 $\dot{x} = -\frac{q_4}{x}(L_0 - q_4) \quad * \text{substitute } q_4 \text{ and } \dot{q}_4$   
 $\dot{x} = -\frac{p_1 \tilde{m}(\theta)}{I_1 x} (L_0 - q_4)$   
 $\dot{x} = \frac{p_1}{I_1 x}$   
 $\tilde{m}(\theta) = -\sin(\theta) = \frac{x}{\sqrt{x^2 + h^2}}$

$$\dot{p}_1 = S'_E - R_3 \frac{p_1}{I_1} + \frac{x(L_0 - \sqrt{x^2 + h^2})}{(\sqrt{x^2 + h^2})^2 (c)}$$

$F_N = I \cdot g + \cos(\theta) F_{spring} = I \cdot g - \frac{1}{(\sqrt{x^2 + h^2})^2} \cdot \frac{(x^2 + h^2)^{1/2} - L_0}{c}$   
 $F_r = \text{abs}(F_N) \text{sign}(p_1)$

B.1



## B.2

1) Maximum excursion for case 1 is 0.7m, and for case 2 was 0.005m. For case 2, the friction immediately slows down the spring and overcomes its force, meaning that it pretty much stops immediately after starting. For the other case, the viscous friction was underdamped and allows it to oscillate a few times before dissipating its energy.

2) The amplitude decreases due to damping, which dissipates energy stored in the system. Final positions are given below:

Case 1	-1.5173e-05m
Case 2	0.0053m
Case 3	-0.0013m
Case 4	-0.0182m

3) Case 4 oscillates at the highest frequency between all the cases because it takes the longest for it to dissipate its energy. When the system has more energy, it will oscillate at a higher frequency because the damping portion of the force is much smaller than the spring force portion, which allows it to move

faster. Case 4 lasts the longest because it is a constant force that is smaller than the force created from the linear damping.

### B.3

1) The final positions are given below:

Case 5	0.1562m
Case 6	0.0049m
Case 7	-0.1572m
Case 8	-0.2247m

2) Case 7 uses linear damping, while case 8 uses coulomb friction to damper the energy. The linear damping force decreases with velocity, allowing the damping force to be 0 when the object comes to rest. This means it will come to rest at the natural state of leaving no energy stored in the system. In case 8 however, there is coulomb friction. This means that after some threshold force is reached, the system will come to a complete stop. The system will have energy stored in the spring that is inadequate to overcome the coulomb friction, which means it will come to rest at a point where there is energy, different from case 7. This is why the two settle on different final positions.

Matlab scripts:

```
% Example ODE45 evaluation of a mass-spring-damper system provided in the
% notes in class.
%
% m-files needed: MSD.m
%
% Last modified: 05 October 2021

close all;

%% Specify time range of interest and initial conditions
tspan=[0 30]; % Time range, [s]
x0=[7.5;0]; % Initial momentum and displacement

%% Integrate state-space equations
[t,x] = ode45(@MSD,tspan,x0);
[t2,x2] = ode45(@MSD2,tspan,x0);
%% Convert momentum state variable to the velocity of the mass
I=1; % Mass, [kg]
qdot=x(:,1)/I;

%% Plot results
figure
plot(t,x(:,2),t2,x2(:,2))
title('Spring-Mass-Damper Response')
xlabel('Time, [s]')
ylabel('Displacement, [m]')
legend('x')
grid on

% figure
```

```

% plot(t,qdot)
% title('Spring-Mass-Damper Response')
% xlabel('Time, [s]')
% ylabel('State Variable')
% legend('x','v')

% This function evaluates the state-space equations at each time called by
% the ode45 function (or other integrator)

function xdot=MSD2(t,x)

% Specify system parameters
R=0.5;           % Damping coefficient, [N.s/m] % 5, 15
I=1;            % Mass, [kg] % 1
C=1/20;         % Compliance, [m/N]
L0 = 0.5; % m
h = 1.05*L0;

%% Specify different types of forcing functions
% Step force on at t = 0
E1=0;

%% Force on, then off at t = 3
% if t<3
%     E1=2;           % Step force, [N]
% else
%     E1=0;
% end
%
% %% Multiple force steps
% if t<2
%     E1=2;
% elseif t>6
%     E1=-4;
% else
%     E1=0;
% end

%% Look at linear and nonlinear examples
% Linear case
% A=[-R/I -1/C;
%     1/I 0];
% b=[1;0];
% xdot=A*x+b*E1;

% Nonlinear case
% damping = (R/I)*x(1);
normal_force = I*9.81 + h/(x(2)^2+h^2)^(1/2)*(L0-(x(2)^2+h^2)^(1/2))/C;
damping = 0.1*abs(normal_force)*sign(x(1));
xdot1 = E1-damping+x(2)*(L0-(x(2)^2+h^2)^(1/2))/((x(2)^2+h^2)^(1/2)*C);
xdot2 = (1/I)*x(1);

xdot=[xdot1;xdot2];

```

```
% This function evaluates the state-space equations at each time called by  
% the ode45 function (or other integrator)
```

```
function xdot=MSD(t,x)
```

```
% Specify system parameters
```

```
R=0.5;           % Damping coefficient, [N.s/m] % 5, 15
```

```
I=1;            % Mass, [kg] % 1
```

```
C=1/20;         % Compliance, [m/N]
```

```
L0 = 0.5; % m
```

```
h = 1.05*L0;
```

```
%% Specify different types of forcing functions
```

```
% Step force on at t = 0
```

```
E1=0;
```

```
%% Force on, then off at t = 3
```

```
% if t<3
```

```
%     E1=2;           % Step force, [N]
```

```
% else
```

```
%     E1=0;
```

```
% end
```

```
%
```

```
% %% Multiple force steps
```

```
% if t<2
```

```
%     E1=2;
```

```
% elseif t>6
```

```
%     E1=-4;
```

```
% else
```

```
%     E1=0;
```

```
% end
```

```
%% Look at linear and nonlinear examples
```

```
% Linear case
```

```
% A=[-R/I -1/C;
```

```
%     1/I 0];
```

```
% b=[1;0];
```

```
% xdot=A*x+b*E1;
```

```
% Nonlinear case
```

```
damping = (R/I)*x(1);
```

```
%normal_force = I*9.81 + (x(2)^2+h^2)^(1/2)*(L0-(x(2)^2+h^2)^(1/2))/C;
```

```
%damping = 0.1*abs(normal_force)*sign(x(1));
```

```
xdot1 = E1-damping+x(2)*(L0-(x(2)^2+h^2)^(1/2))/((x(2)^2+h^2)^(1/2)*C);
```

```
xdot2 = (1/I)*x(1);
```

```
xdot=[xdot1;xdot2];
```

