Simulating vibration using MATLAB (Draft):

This file contains MATLAB codes to simulate vibration for different conditions. We will not discuss relevant theory for each case. For theory, readers might refer to following standard textbooks.

1. Engineering Vibration by Daniel J. Inman (Fourth edition, 2014)
2. Theory of Vibrations with Applications by William T. Thomson and Marie Dillon Dahleh (Fifth edition, 2013)

Note: In the beginning, we have given codes for simplest cases. These codes are trivial and not at all new. These are given for the sake of completeness only. Relatively interesting material such as **shock response spectrum (SRS)** appear towards the end. So readers are advised to pick and choose code segments as per their need. Also note that this is a rough draft. Many more modifications will be done to this in future.

Notations:

SDOF : Single degree of freedom

m: Mass

k : Stiffness

ω_n: Natural frequency

ζ: Damping factor $(\frac{c}{c_c})$

c : Damping constant

c_c : Critical damping constant

r : Frequency ratio $(\frac{\omega}{\omega_n})$

Our governing equation of motion (for free vibration) is

$$m\ddot{x} + c\dot{x} + kx = 0$$

Or equivalently,

$$\ddot{x} + 2\zeta\omega_n x + \omega_n^2 x = 0$$

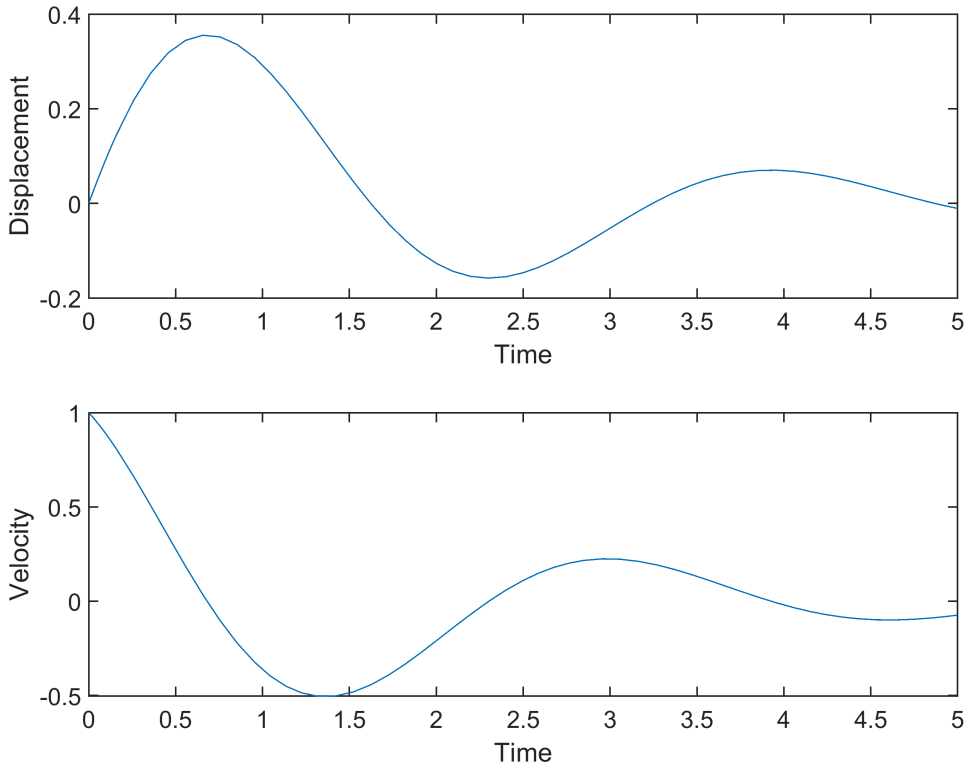
For undamped case, $\zeta = 0$.

Free SDOF System (Undamped and Damped):

To simulate undamped case, change the value of `zeta` to 0 in the following code. The other values for mass and stiffness are chosen arbitrarily. We have used MATLAB's `ode45` differential equation solver. For a quick introduction to solving differential equations using MATLAB, readers may refer to this link.

```
% Change the value of zeta to impart required amount of damping
% Change the initial condition as required
```

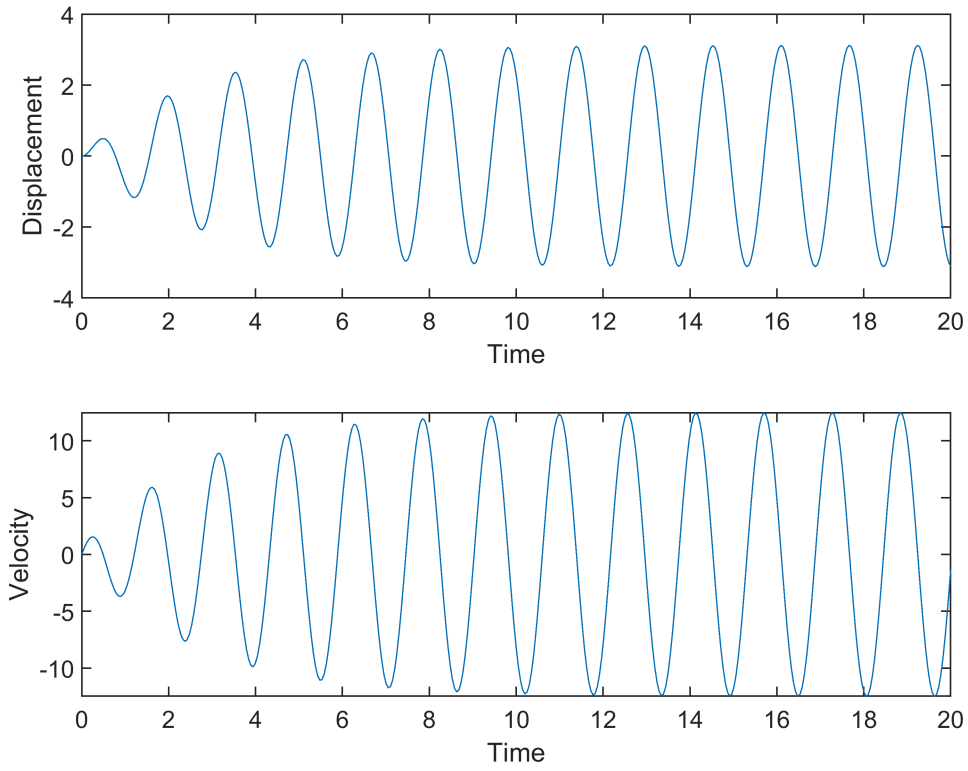
```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.25;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)];
[t,y] =  ode45(fun,[0,5],[0;1]);
figure
subplot(2,1,1)
plot(t,y(:,1))
ylabel('Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



Forced SDOF System (Undamped and Damped):

```
% Sinusoidal input force with input frequency different from natural frequency.
% Vary damping and initial conditions to see changes.
m = 1; k = 16; wn = sqrt(k/m); zeta = 0.1;f0 = 10;w = wn;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ (f0/m)*cos(w*t)];
[t,y] =  ode45(fun,[0,20],[0.0;0.0]);
figure
subplot(2,1,1)
plot(t,y(:,1))
ylabel('Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
```

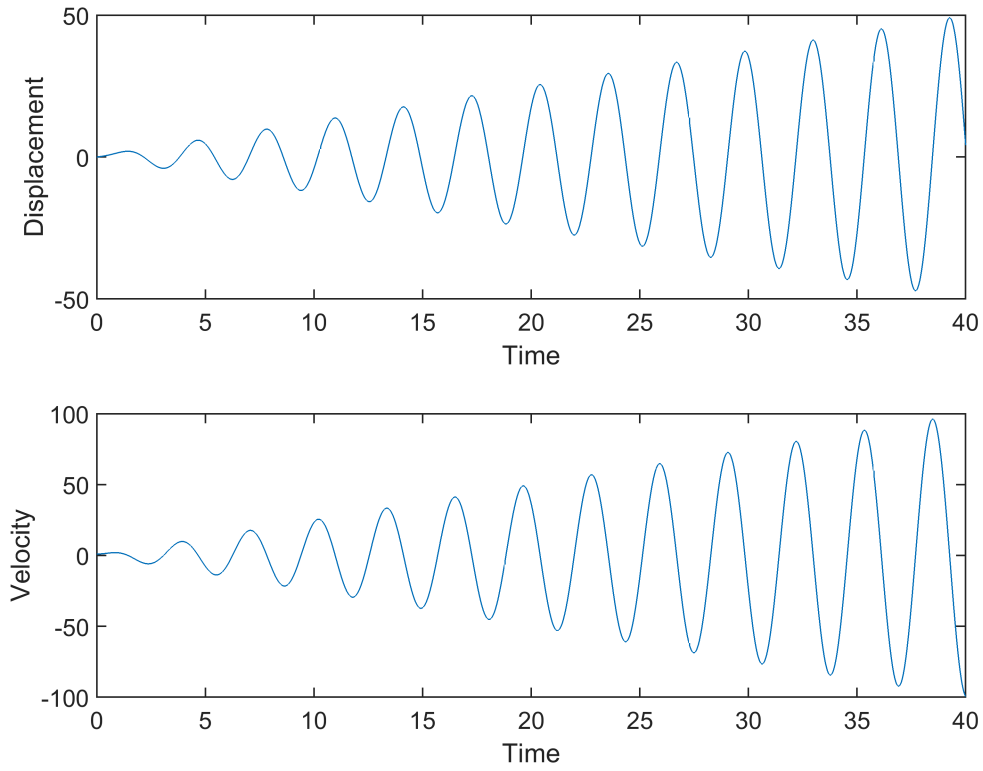
```
xlabel('Time')
ylabel('Velocity')
```



Forced SDOF system (resonance case):

As it is a special case, it is treated separately.

```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.0;f0 = 5;w = wn;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*sin(w*t)];
[t,y] =  ode45(fun,0:0.01:40,[0;1]);
figure
subplot(2,1,1)
plot(t,y(:,1))
ylabel('Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```

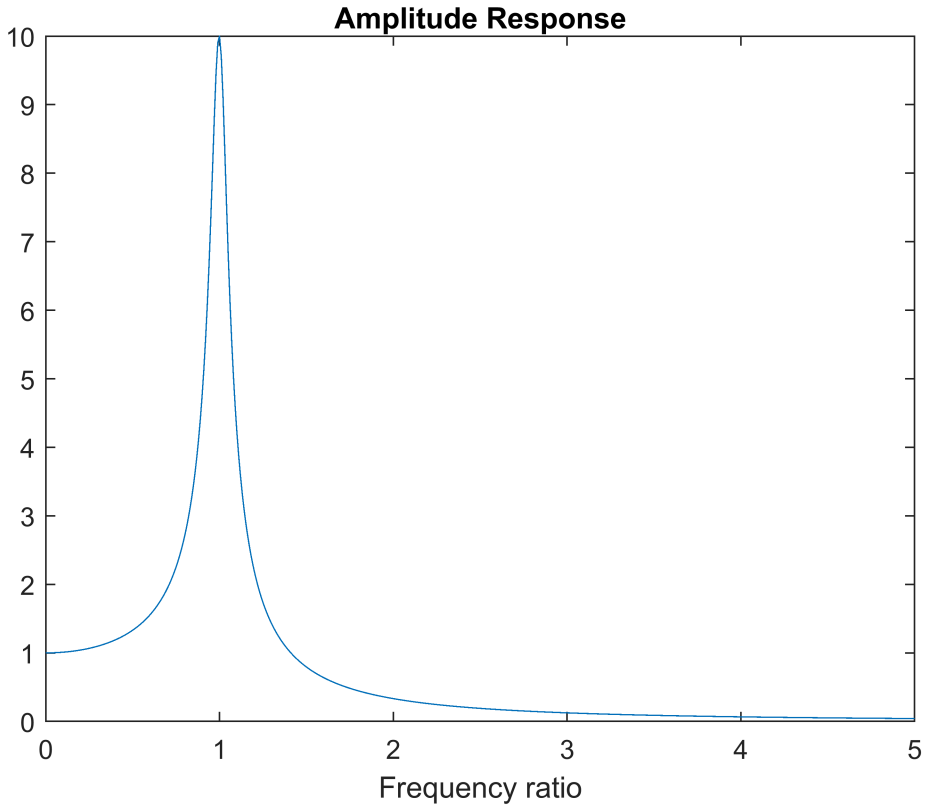


Observe that both velocity and acceleration are increasing gradually (see y-axis scale in each plot). Their envelope is linear. Curious readers who want to plot envelope of this signal can refer to this link.

Frequency response of the system:

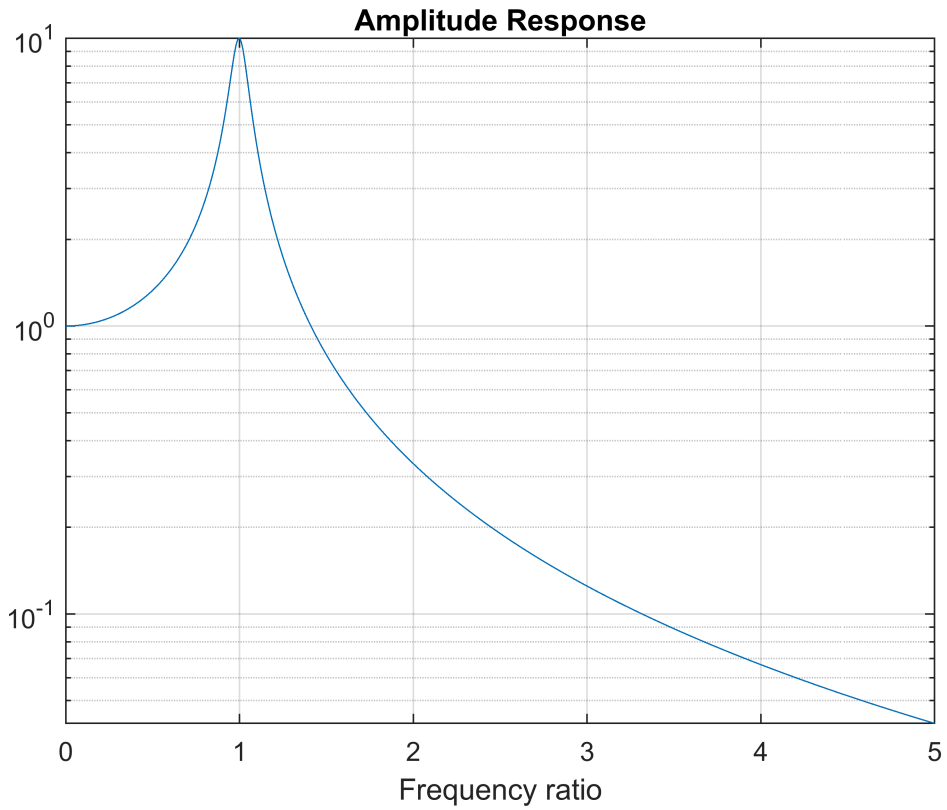
Plotted on a linear scale.

```
% Both Amplitude response and Phase response
k = 1;zeta = 0.05;f0 = 1;
r = 0:0.01:5;
y = (f0/k)./((1-r.^2)+1j*(2*zeta*r));
figure
plot(r,abs(y))
title('Amplitude Response')
xlabel('Frequency ratio')
```



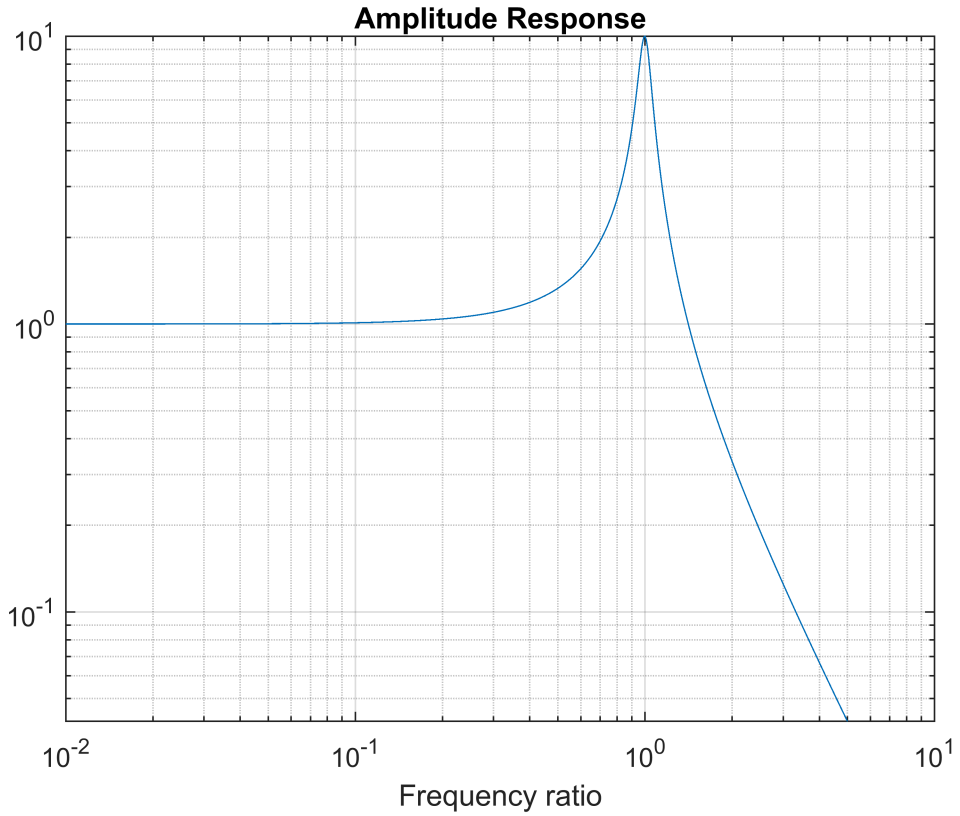
Semilog plot of gain:

```
figure
semilogy(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```



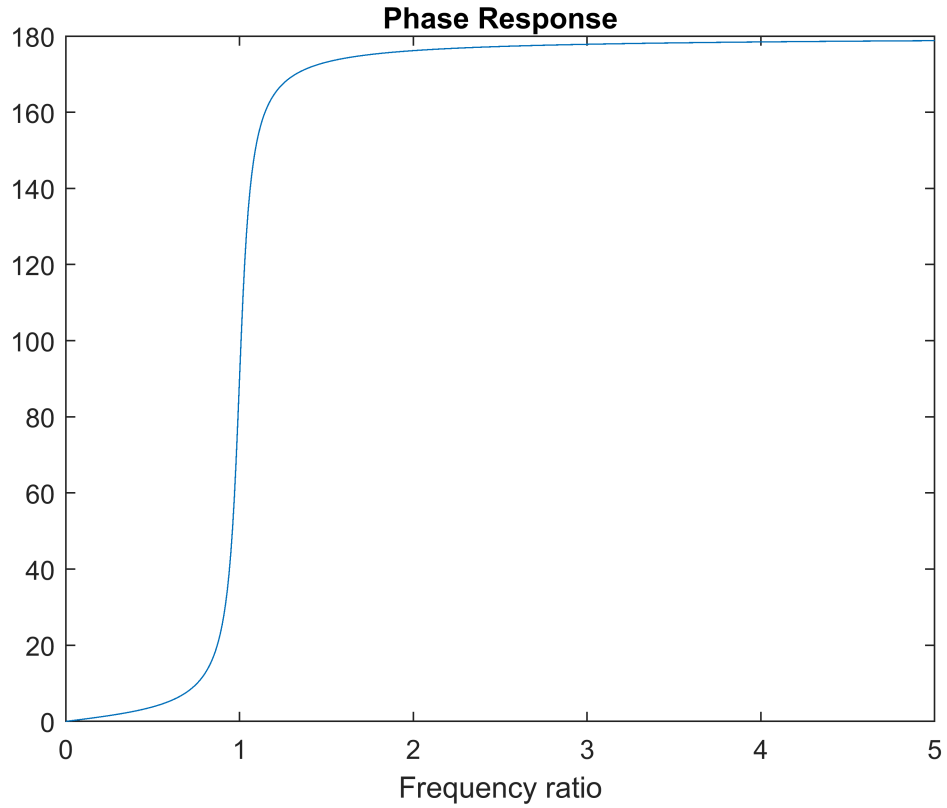
Log-log plot of gain:

```
figure
loglog(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```



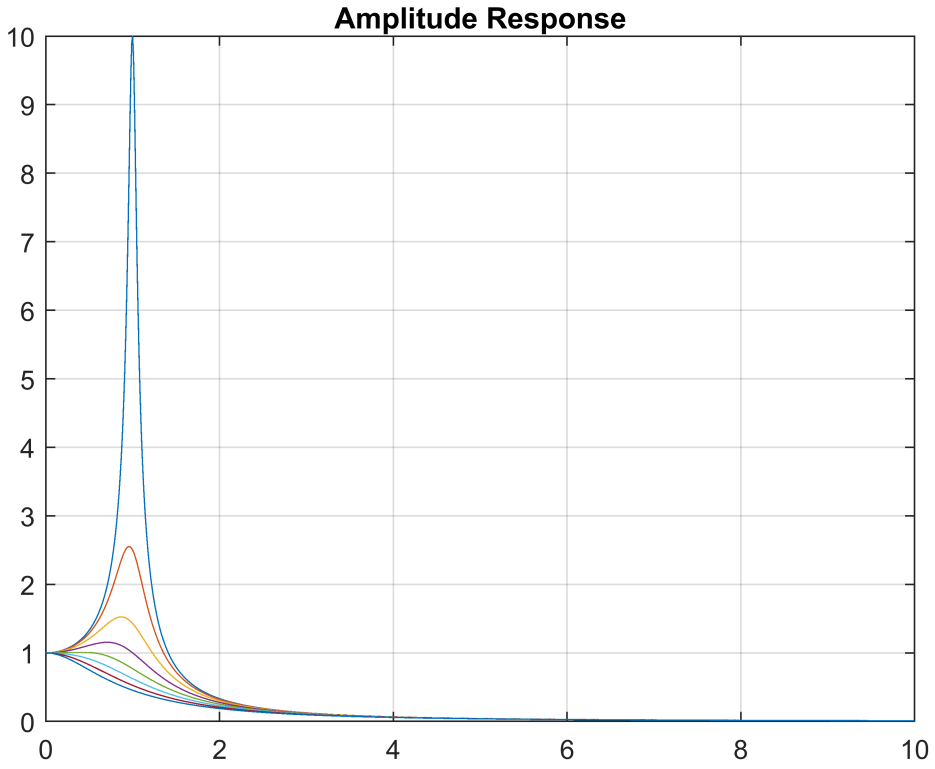
Phase response:

```
figure
plot(r,-(180/pi)*angle(y))
title('Phase Response')
xlabel('Frequency ratio')
```



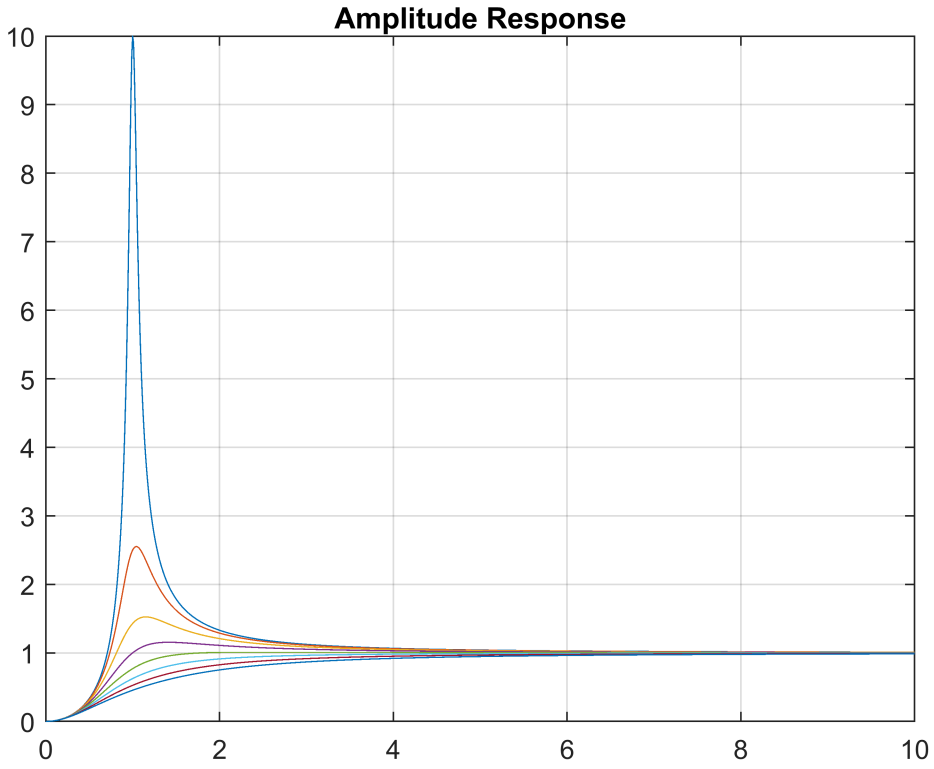
Frequency response plot for varying values of ζ:

```
% Plots the Frequency Response of SDOF System with...
% different values of damping factor(zeta)
% First figure is the Gain plot
clear;
y= zeros(1001,1);
figure
for zeta= 0.05:0.15:1.2
    i=0;
  for r=0:0.01:10
      y(i+1)= 1/((1-r^2)+j*(2*zeta*r));
      i=i+1;
  end

  plot(0:0.01:10,abs(y))
  hold on

end
hold off
grid on
title('Amplitude Response')
```



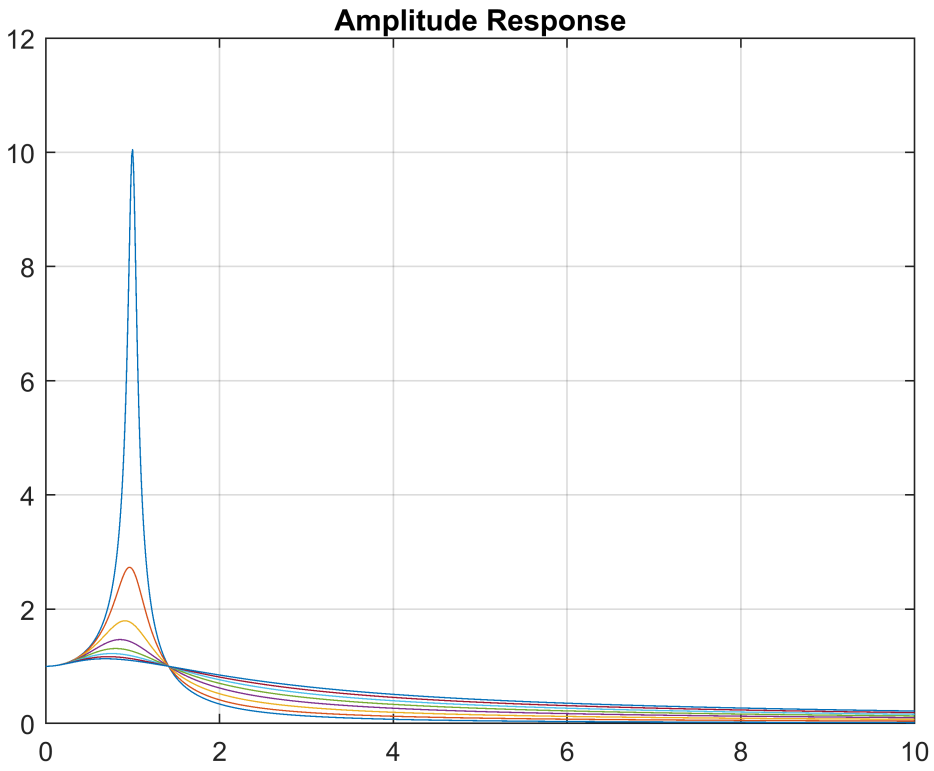
Rotating Unbalance:

```
% Amplitude response for varying values of zeta.
y= zeros(1001,1);
figure
for zeta= 0.05:0.15:1.2
    i=0;
  for r=0:0.01:10
      y(i+1)= r^2/((1-r^2)+j*(2*zeta*r));
      i=i+1;
  end

  plot(0:0.01:10,abs(y))
 hold on

end
hold off
grid on
title('Amplitude Response')
```



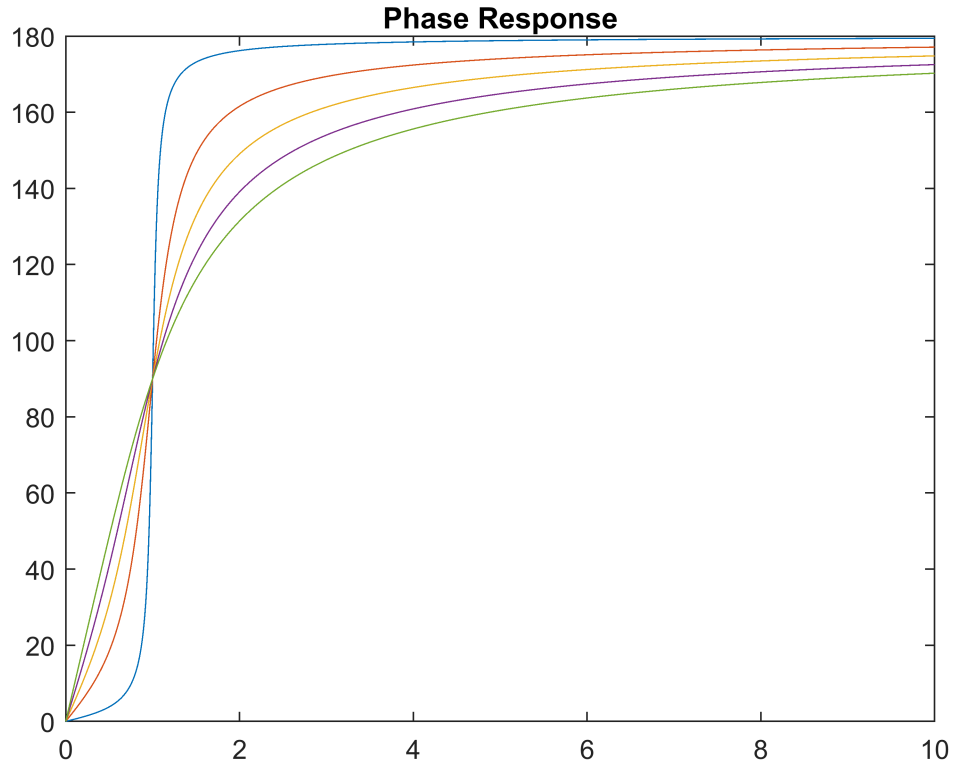
Support Motion Amplitude response:

```
% Amplitude response for varying values of zeta.
y= zeros(1001,1);
figure
for zeta= 0.05:0.15:1.2
    i=0;
  for r=0:0.01:10
      y(i+1)= (1+(1j*2*zeta*r))/((1-r^2)+j*(2*zeta*r));
      i=i+1;
  end

  plot(0:0.01:10,abs(y))
 hold on

end
hold off
grid on
title('Amplitude Response')
```



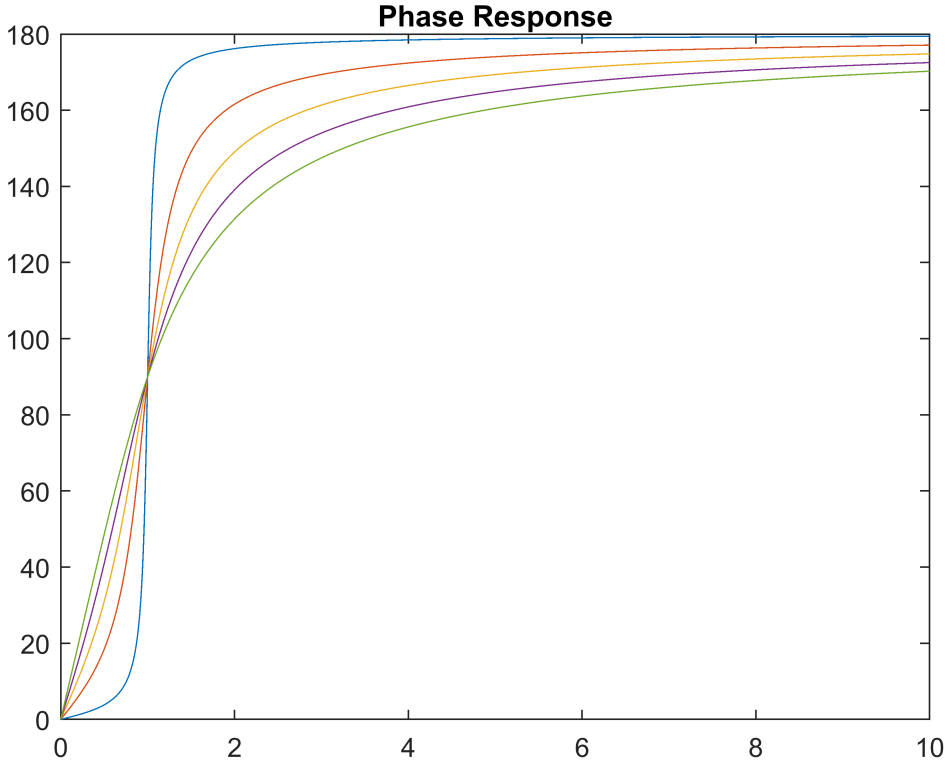
Phase plot:

```
% This figure plots phage angle
y= zeros(1001,1);
figure
for zeta= 0.05:0.2:1
    i=0;
  for r=0:0.01:10
    y(i+1)= 1/((1-r^2)+j*(2*zeta*r));
    i=i+1;
  end

 plot(0:0.01:10,-(180/pi)*angle(y))
 hold on

end
hold off
title('Phase Response')
```



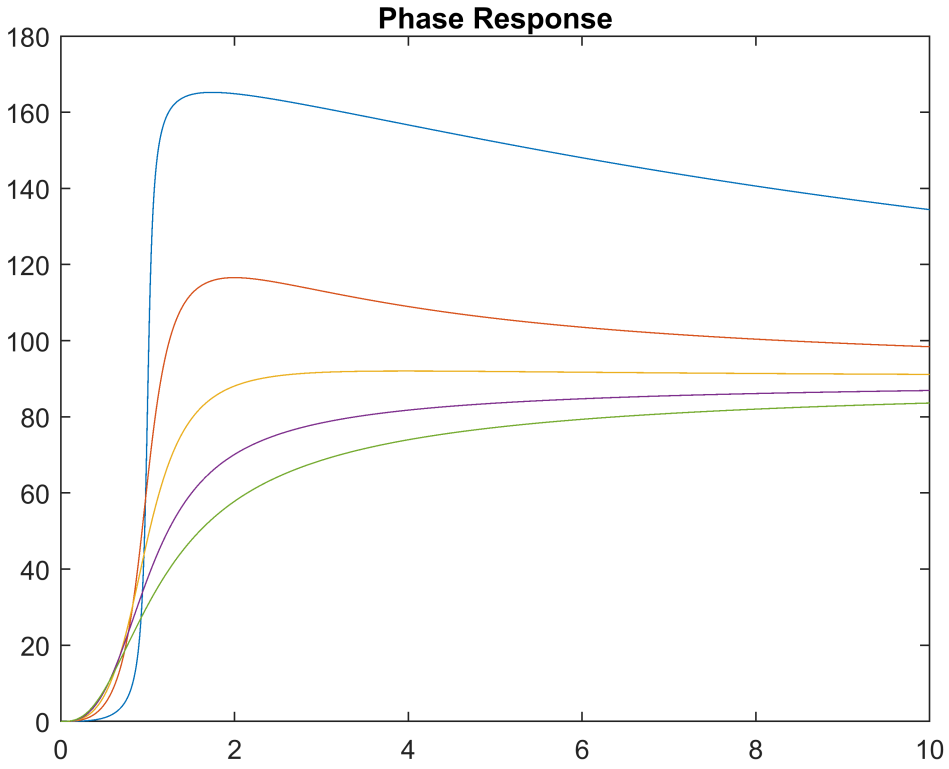
Rotating Unbalance (phase plot):

```
% Phase response for varying values of zeta.
y= zeros(1001,1);

figure
for zeta= 0.05:0.2:1
    i=0;
  for r=0:0.01:10
    y(i+1)= r^2/((1-r^2)+j*(2*zeta*r));
    i=i+1;
  end

 plot(0:0.01:10,-(180/pi)*angle(y))
 hold on

end
hold off
title('Phase Response')
```

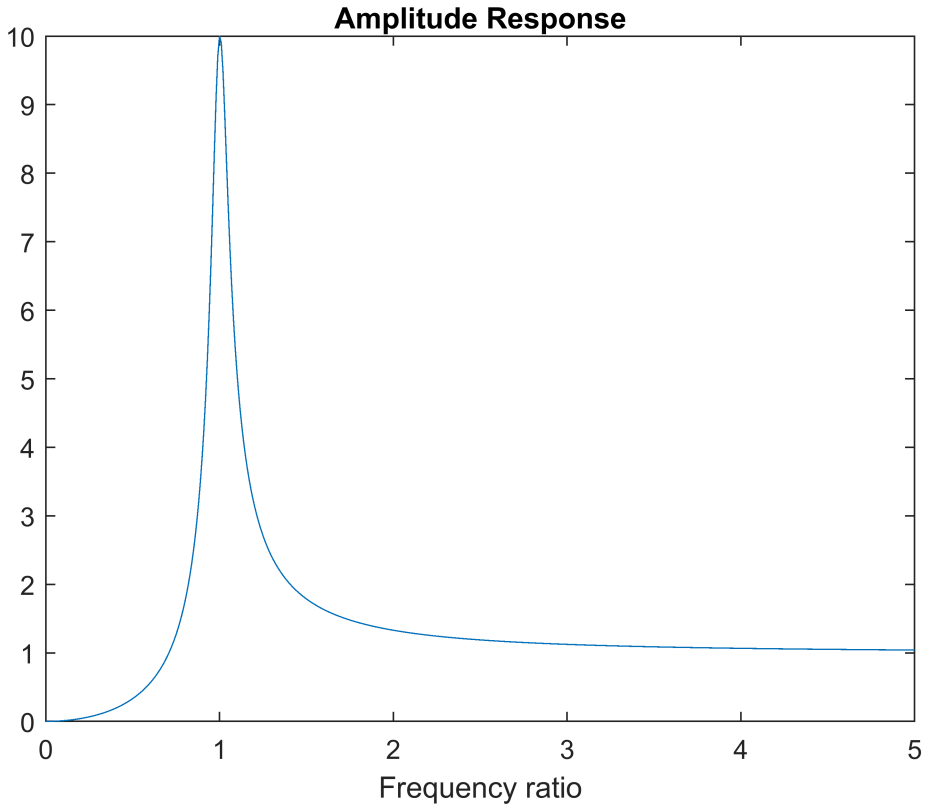


Support Motion (phase plot):

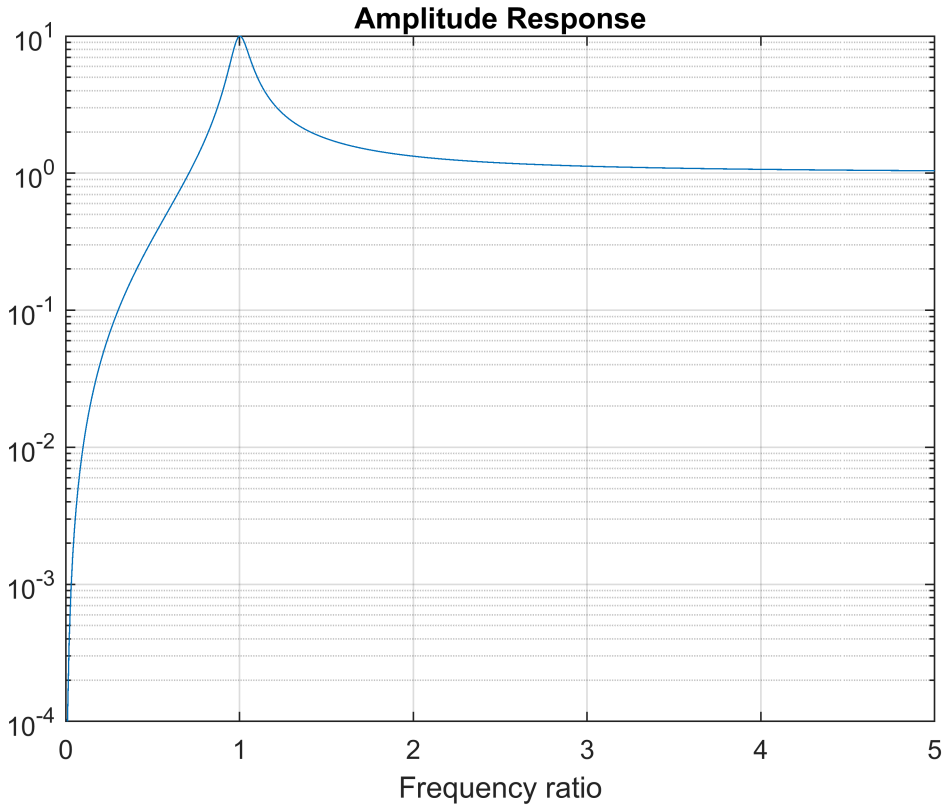
```
% Phase response for varying values of zeta.
y= zeros(1001,1);

figure
for zeta= 0.05:0.2:1
    i=0;
  for r=0:0.01:10
    y(i+1)= (1+(1j*2*zeta*r))/((1-r^2)+j*(2*zeta*r));
    i=i+1;
  end

 plot(0:0.01:10,-(180/pi)*angle(y))
 hold on

end
hold off
title('Phase Response')
```



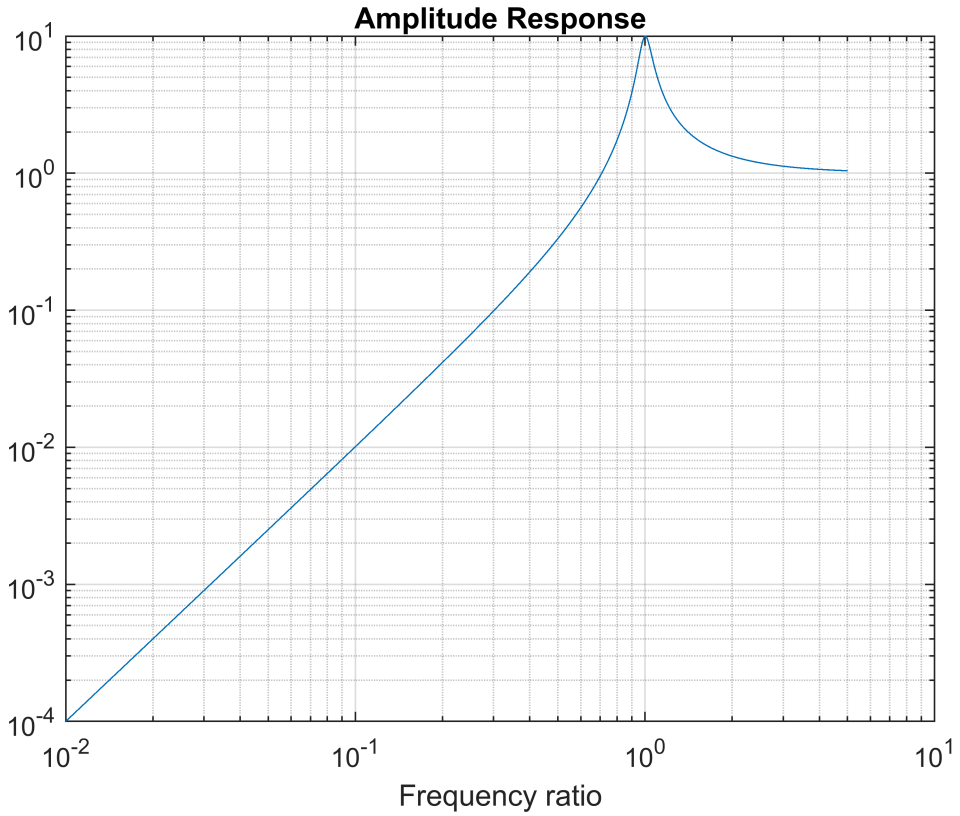

```
%%
% For rotating unbalance
% The amplitude response is calculated as a function of frequency ration.
%m = 1; wn = sqrt(k/m);
k = 1;zeta = 0.05;f0 = 1;
r = 0:0.01:5;
y = (f0/k)*r.^2./((1-r.^2)+1j*(2*zeta*r));
figure
plot(r,abs(y))
title('Amplitude Response')
xlabel('Frequency ratio')
```



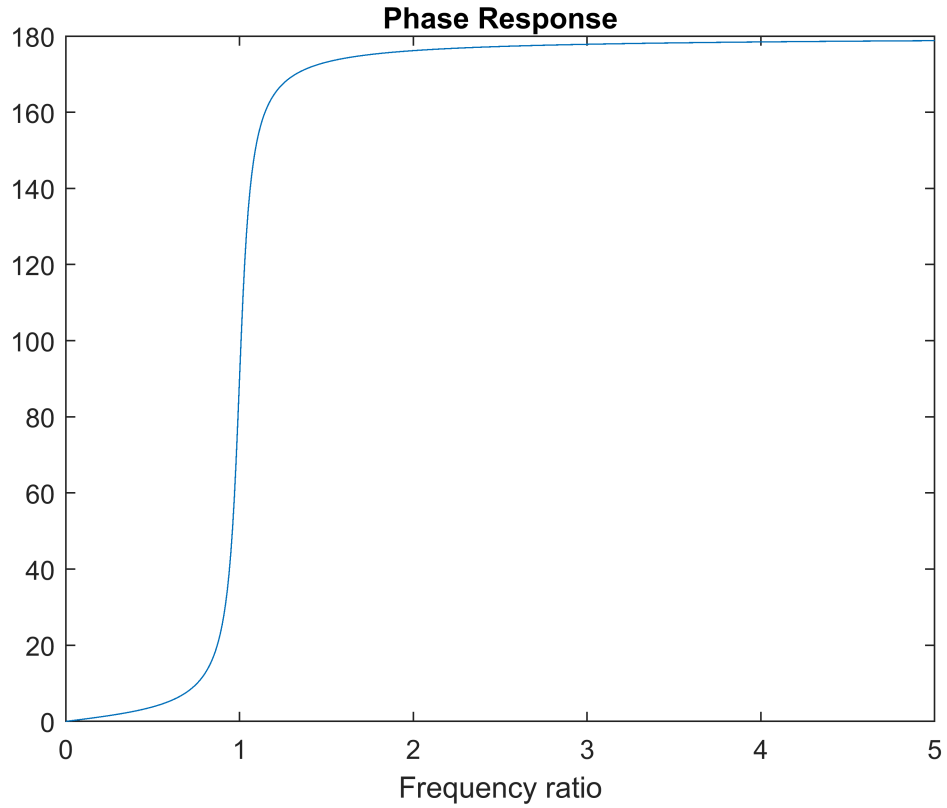

```
figure
% Semilog plot of gain
semilogy(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```



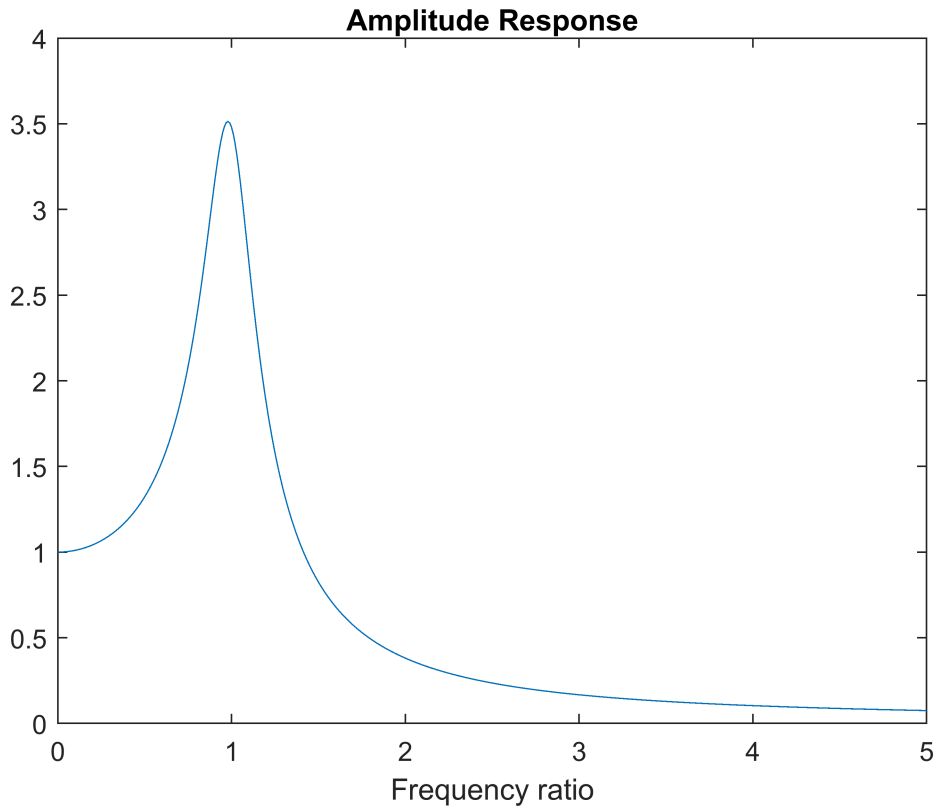

```
figure
% Loglog plot of gain
loglog(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```



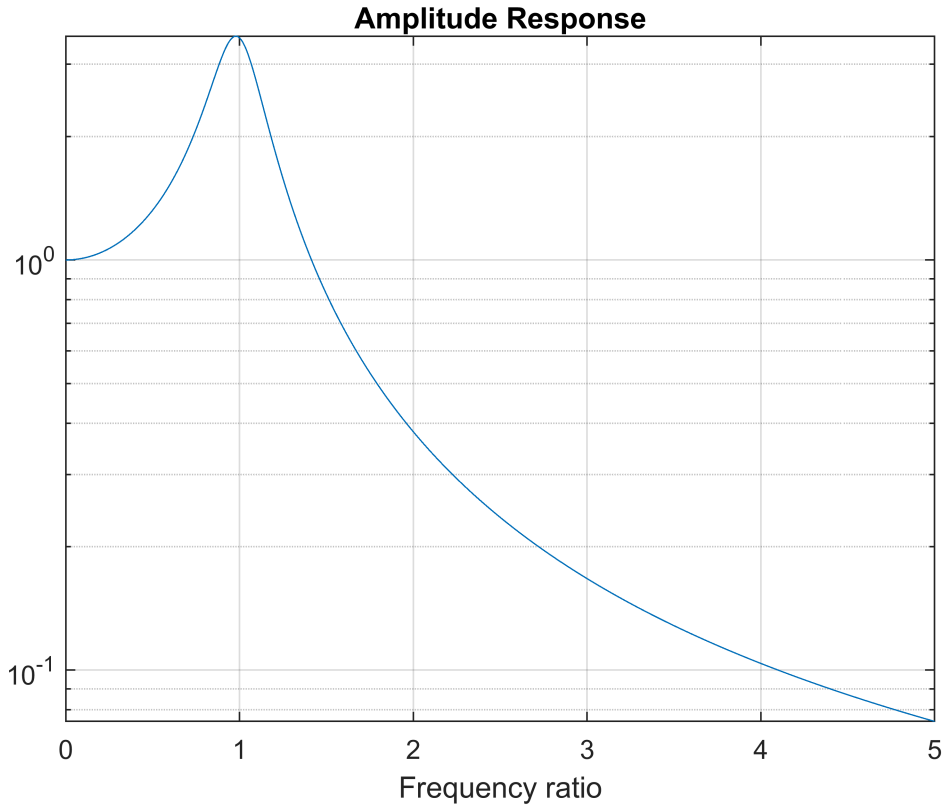

```
% Phase response
figure
plot(r,-(180/pi)*angle(y))
title('Phase Response')
xlabel('Frequency ratio')
```



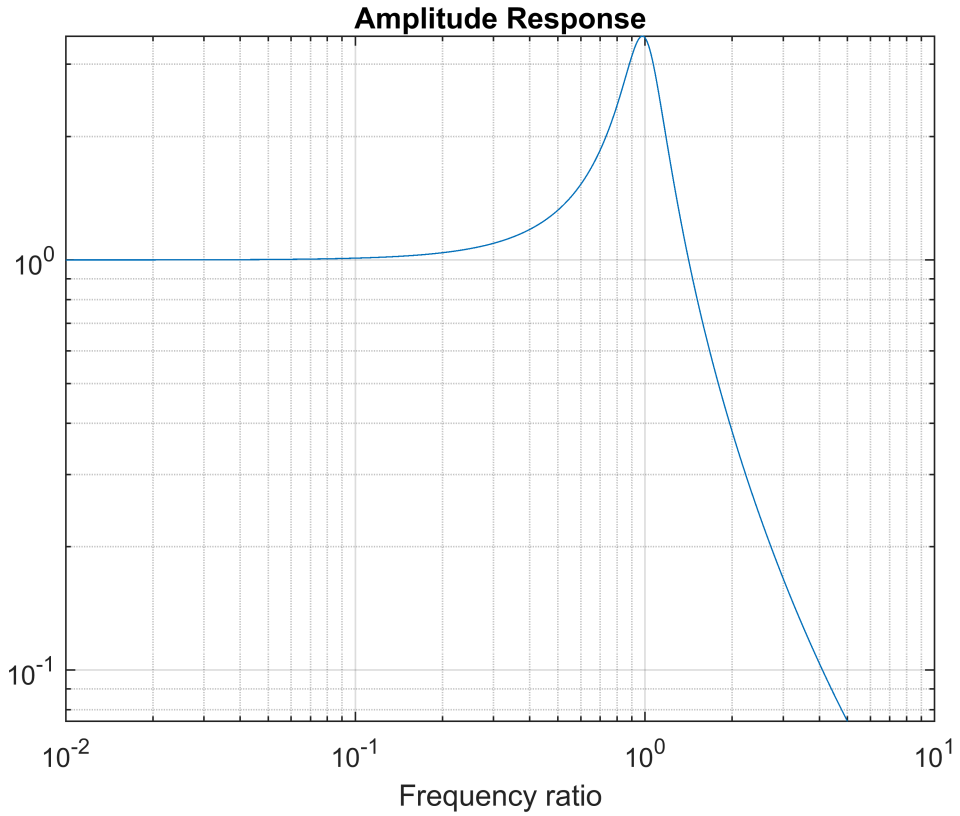

```
%%
% Amplitude and Phase response for Support motion.
%m = 1; wn = sqrt(k/m);
k = 1;zeta = 0.15;f0 = 1;
r = 0:0.01:5;
y = (f0/k)*(1+1j*2*zeta*r)./((1-r.^2)+1j*(2*zeta*r));
plot(r,abs(y))
title('Amplitude Response')
xlabel('Frequency ratio')
```



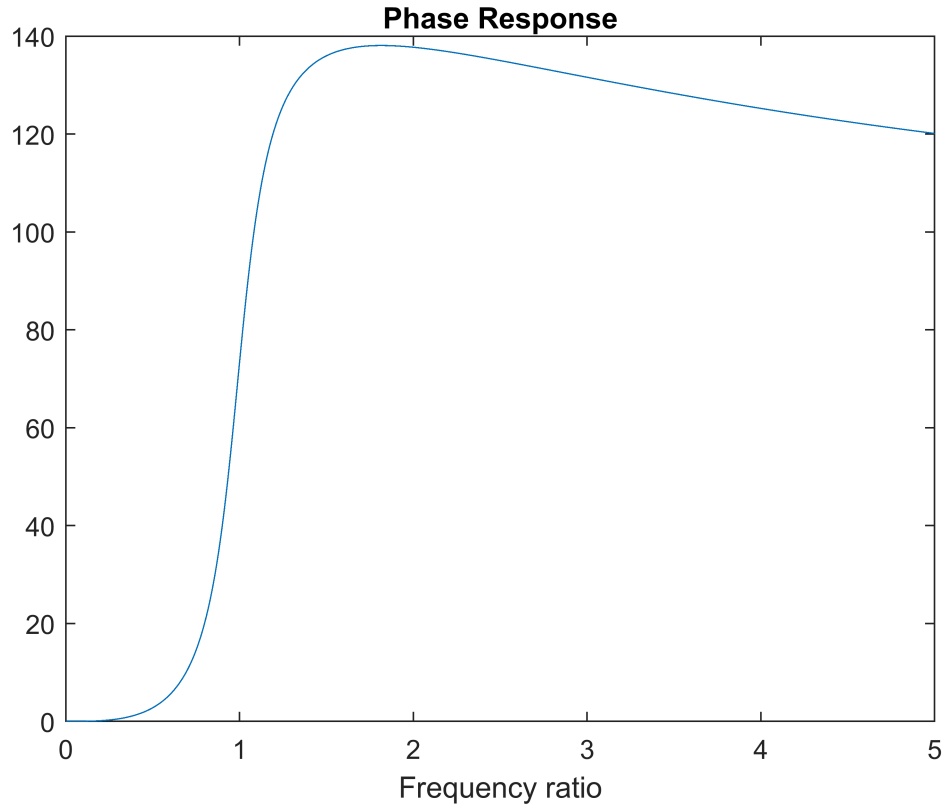

```
figure
% Semilog plot of gain
semilogy(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```




```
figure
% Loglog plot of gain
loglog(r,abs(y))
grid on
title('Amplitude Response')
xlabel('Frequency ratio')
```

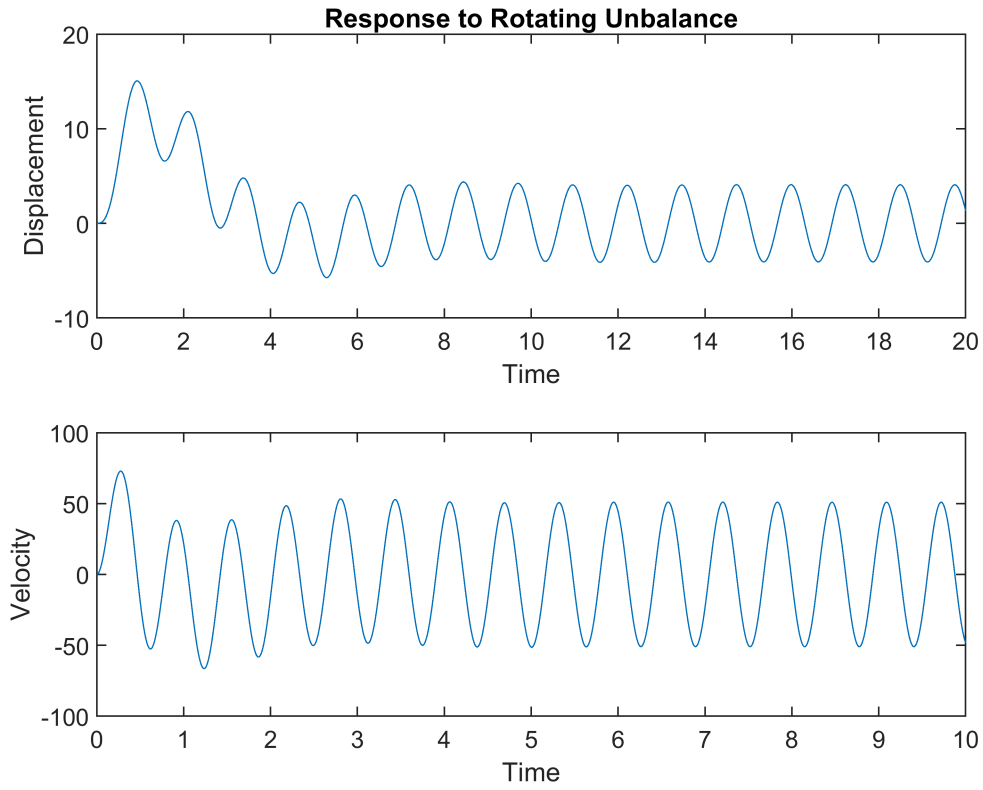



```
% Phase response
figure
plot(r,-(180/pi)*angle(y))
title('Phase Response')
xlabel('Frequency ratio')
```



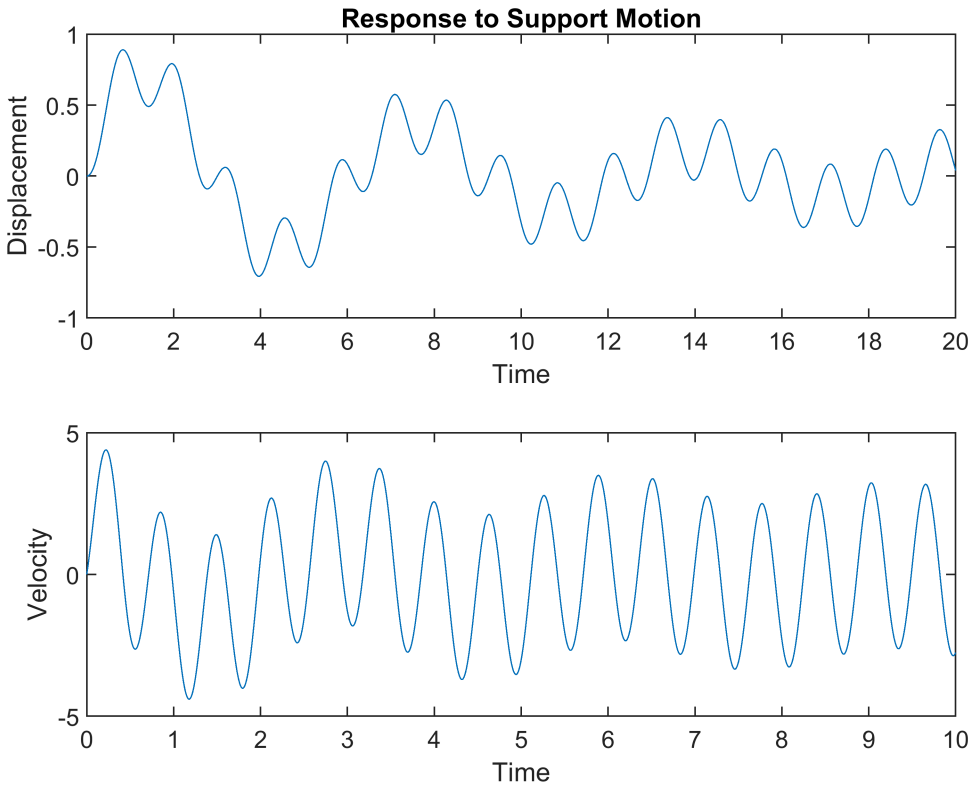
Actural displacement in case of Rotating Unbalanace (Thomson):

```
% w is the external frequency.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.5;f0 = 5;w = 10;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*w^2*sin(w*t)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Rotating Unbalance')
ylabel('Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



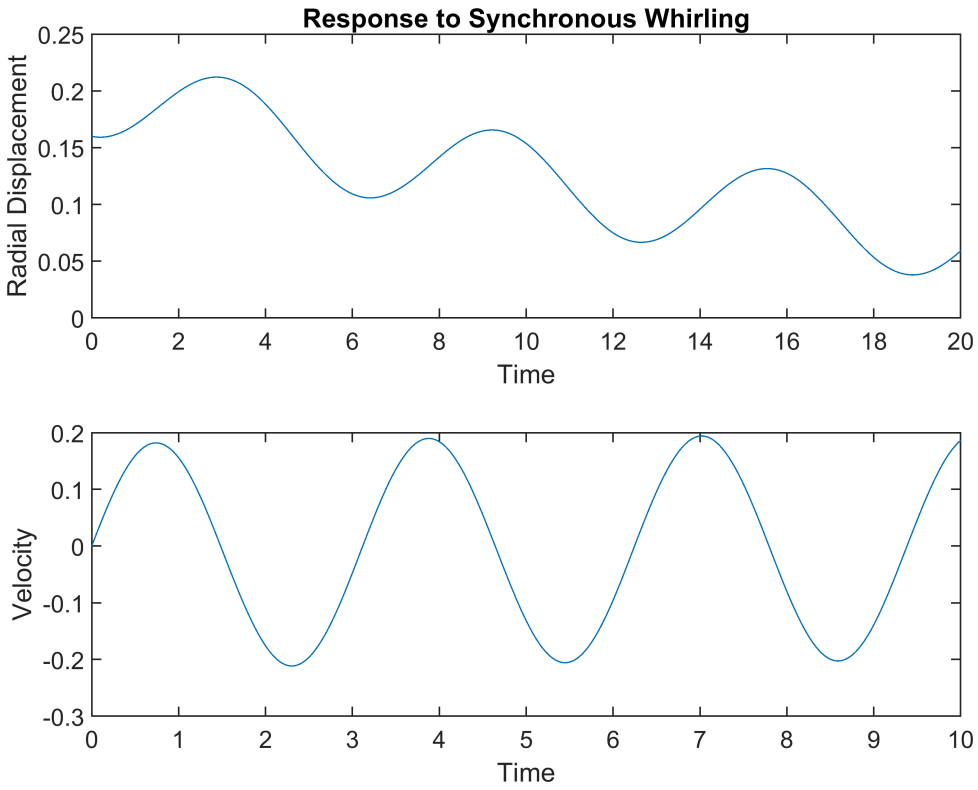
Actual displacement in case of Support motion (Thomson):

```
% w (omega) is the external frequency.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.1;Y = 5;w = 10;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ wn^2*Y*sin(w*t)+(2*zeta*wn)*w*Y*cos(w*t)
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Support Motion')
ylabel('Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



Synchronous Whirling (Check Again):

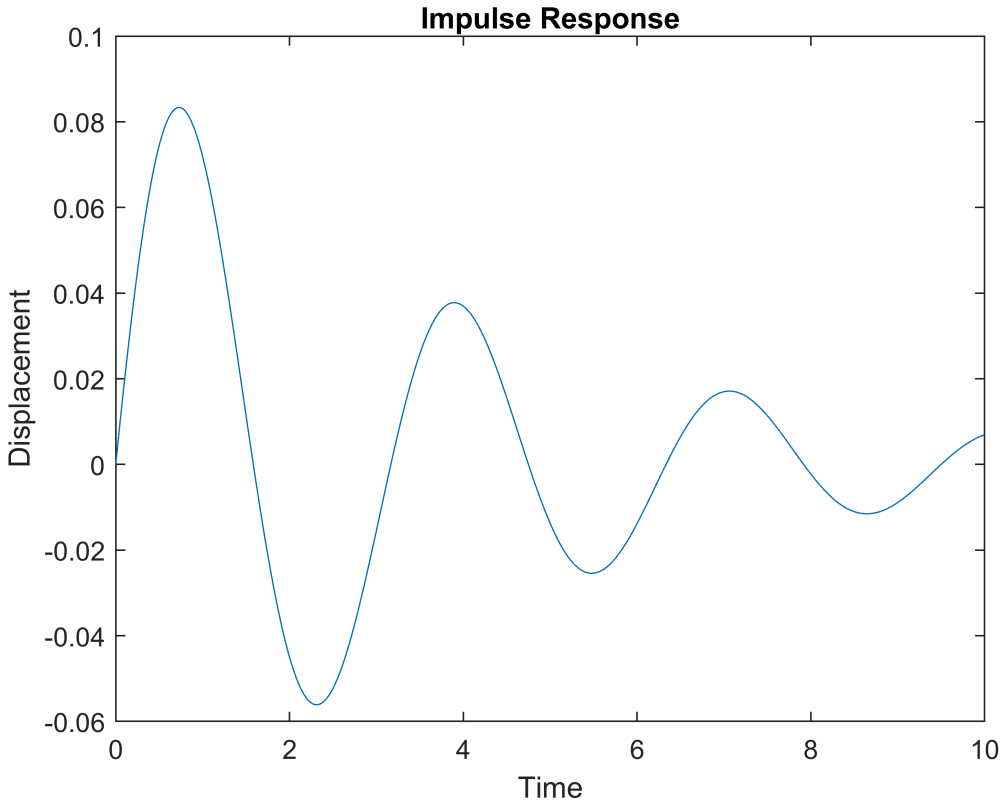
```
% e is the eccentricity.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.05;Y = 5;w = wn;e = 0.1;phi = 0;
fun = @(t,x) [-(zeta*wn)*x(1)+e*w*sin(w*t-phi)/2;...
              -(wn^2-w^2)*x(1) + (-2*zeta*wn)*x(2)+ e*w^2*cos(w*t-phi)];
[t,y] =  ode45(fun,0:0.01:10,[0.2;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Synchronous Whirling')
ylabel('Radial Displacement')
xlabel('Time')
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



Impulse Response:

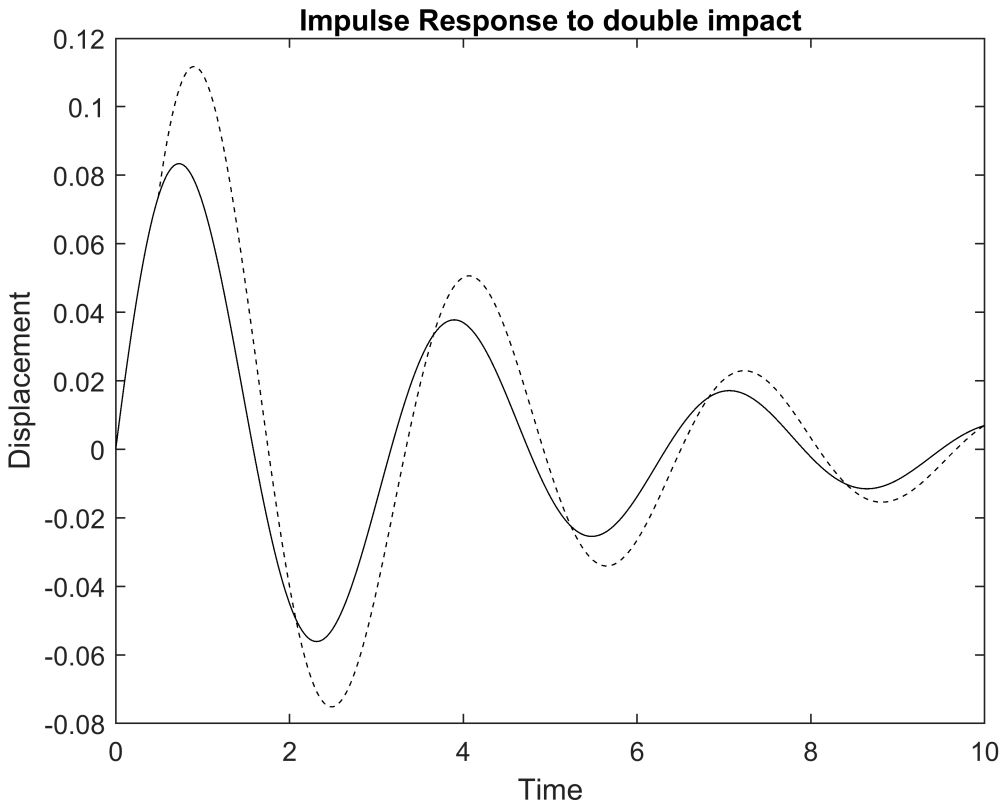
Impulse Response is calculated using Laplace Transform.

```
% Unit impulse is applied at t1.
% t is the usual time that starts at 0.
% Save the fuction unit_impulse.m in current directory.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.5/(2*sqrt(k*m));f0 = 0.2;
num = f0/m;
den = [1,2*zeta*wn,wn^2];
sys = tf(num,den);
t = 0:0.01:10;
y = impulse(sys,t);
figure
plot(t,y)
title('Impulse Response')
ylabel('Displacement')
xlabel('Time')
```



Double Impulse Response (Inman Example):

```
% First impulse is applied at time 0 and second impulse is applied at time
% 0.5 seconds. The response of the system to second impulse is just the
% time shifted version of the impulse at origin.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.5/(2*sqrt(k*m));f0 = 0.2;
num = f0/m;
den = [1,2*zeta*wn,wn^2];
sys = tf(num,den);
t = 0:0.01:10;
y = impulse(sys,t);
z = 0.5*circshift(y,49); % Second impulse amplitude is half of the first one
z(1:49) = 0;
figure
plot(t,y,'k',t,y+z,'--k')
title('Impulse Response to double impact')
ylabel('Displacement')
xlabel('Time')
```

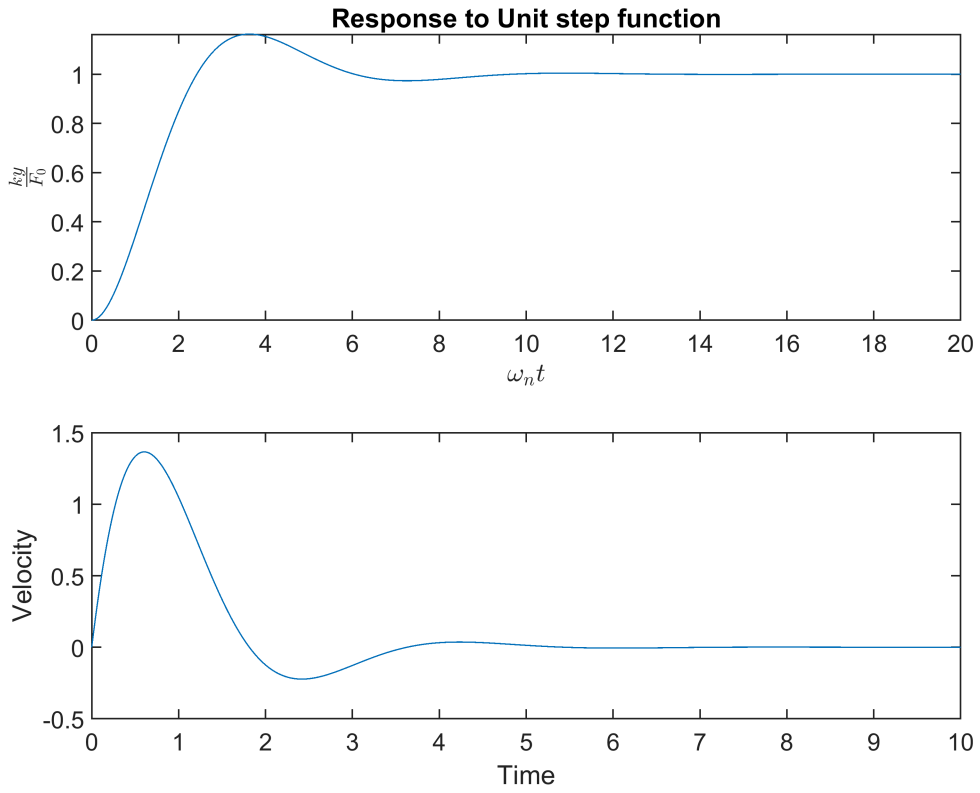


Transient Vibration:

Here we have used unit step function. The codes for unit step function and other functions can be found at the end of this document.

Response to unit step input:

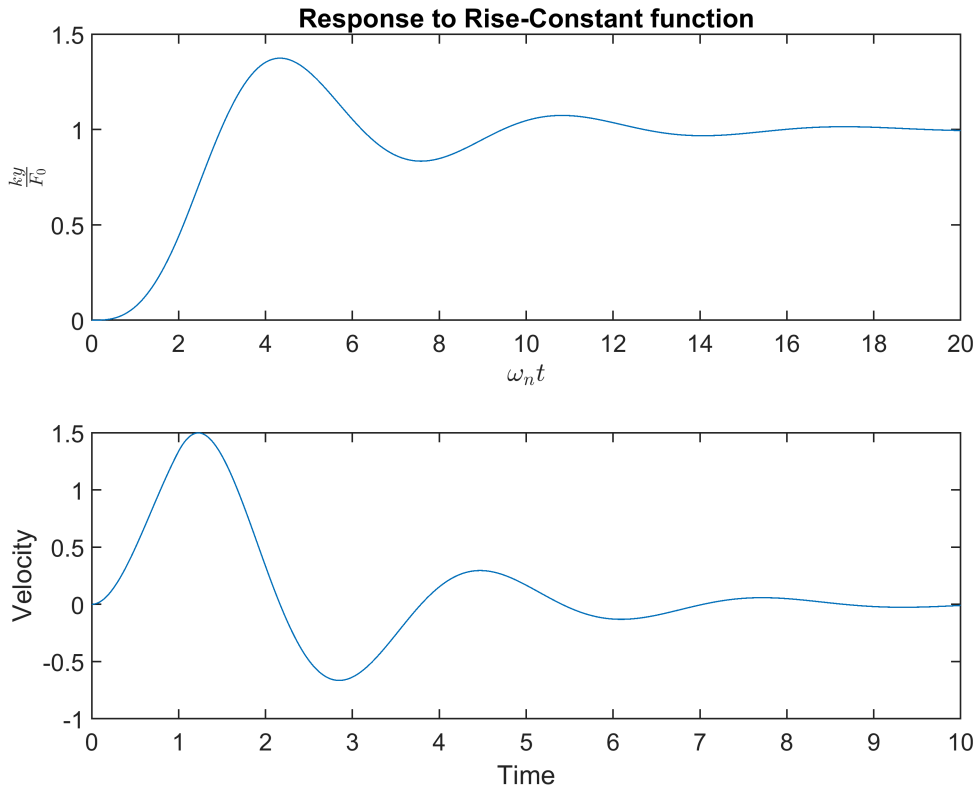
```
%  with step amplitude f0.
% Save the ustep.m function in current directory and then execute.
% By varying the damping factor we will get different responses.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.5;f0 = 5;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*ustep(t,0)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Unit step function')
ylabel('$\frac{ky}{F_0}$','Interpreter','latex','Fontsize',24)
xlabel('$\omega_nt$','Interpreter','latex','Fontsize',24)
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



Response to ramp-constant function:

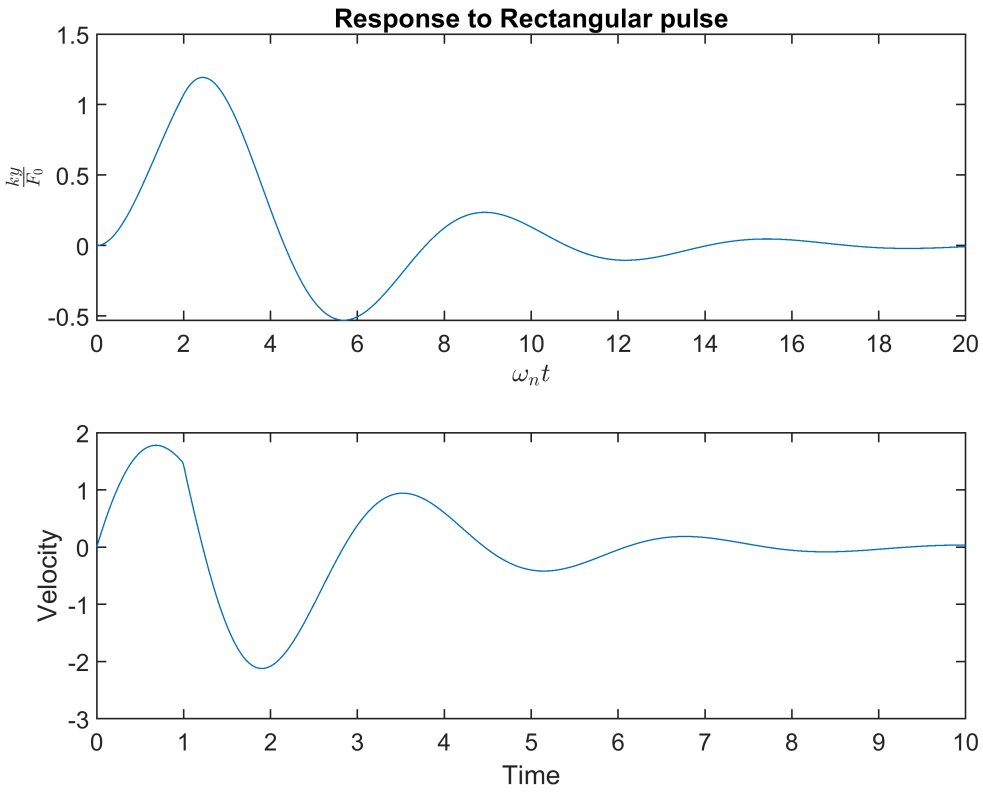
The function `rise_constant` used in the following code segment can be found at the end of this article. S

```
% Excitation force is constant after a rise time. The constant amplitude is f0.
% Save the rise_constant.m file in working directiory.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.25;f0 = 5;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*rise_constant(t,1)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Rise-Constant function')
ylabel('$\frac{ky}{F_0}$','Interpreter','latex','Fontsize',24)
xlabel('$\omega_nt$','Interpreter','latex','Fontsize',24)
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



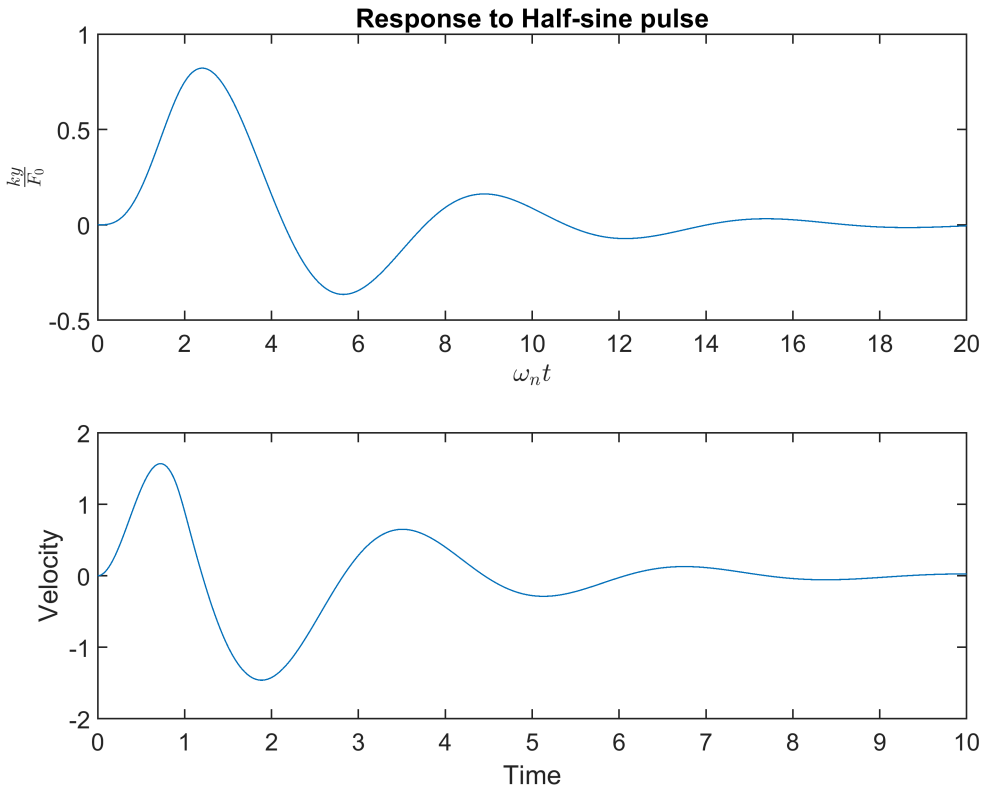
Response to rectangular pulse:

```
% The rectangular pulse stops at t1 and has an amplitude of f0.
% Save the function rect_pulse.m in current directory.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.25;f0 = 5;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*rect_pulse(t,1)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Rectangular pulse')
ylabel('$\frac{ky}{F_0}$','Interpreter','latex','Fontsize',24)
xlabel('$\omega_nt$','Interpreter','latex','Fontsize',24)
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



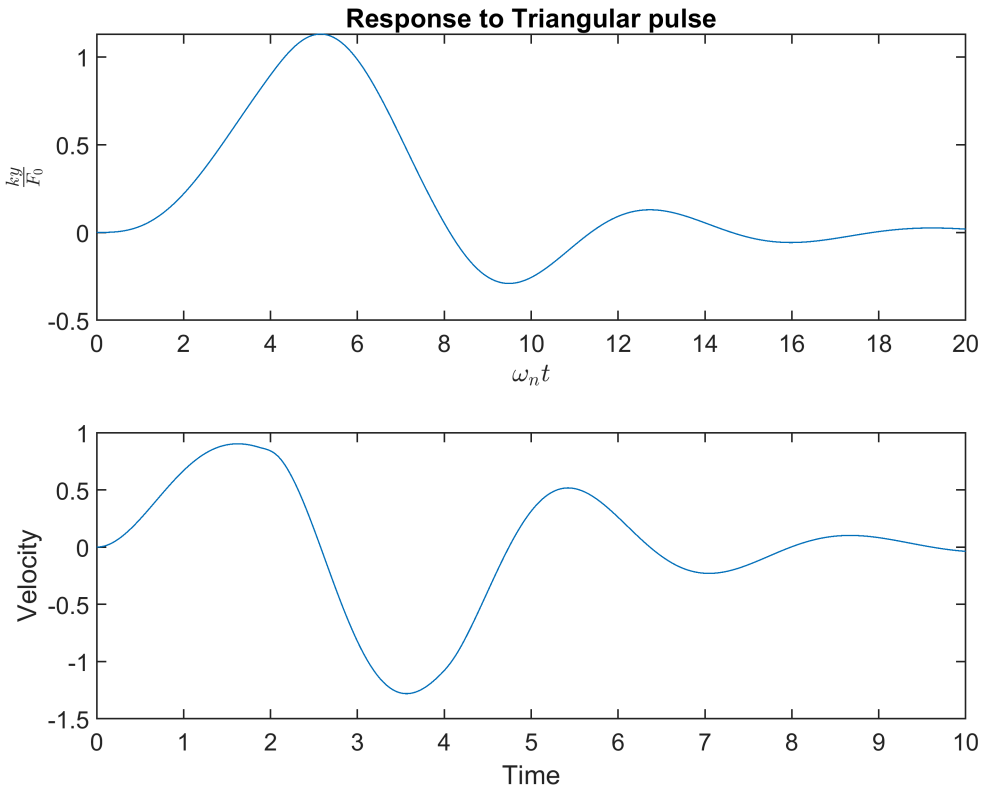
Response to Half-Sine Pulse:

```
% The amplitude of the half sine pulse is f0 and half period is t1.
% Save the half_sine.m file in current directory.
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.25;f0 = 5;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*half_sine(t,1)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Half-sine pulse')
ylabel('$\frac{ky}{F_0}$','Interpreter','latex','Fontsize',20)
xlabel('$\omega_nt$','Interpreter','latex','Fontsize',20)
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



Response to Triangular Pulse:

```
% The pulse starts at zero, reaches peak amplitude of 1 at time t1, and
% reaches 0 again at t2.
% t is the usual time that starts at 0.
% The function takes 3 aguments (t,t1,t2).
m = 1; k = 4; wn = sqrt(k/m); zeta = 0.25;f0 = 5;
fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*triangular_pulse(t,2,4)];
[t,y] =  ode45(fun,0:0.01:10,[0;0]);
figure
subplot(2,1,1)
plot(wn*t,(k/f0)*y(:,1))
title('Response to Triangular pulse')
ylabel('$\frac{ky}{F_0}$','Interpreter','latex','Fontsize',20)
xlabel('$\omega_nt$','Interpreter','latex','Fontsize',20)
subplot(2,1,2)
plot(t,y(:,2))
xlabel('Time')
ylabel('Velocity')
```



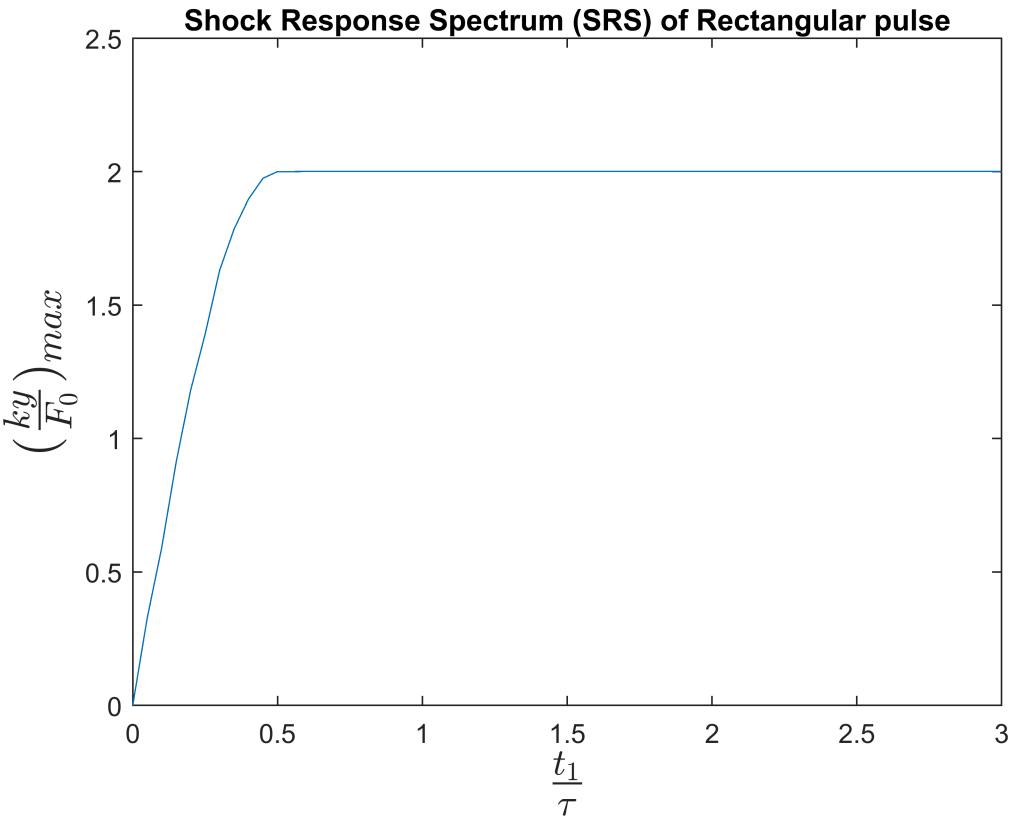
Shock Response Spectrum:

Let t_1 be the pulse duration. When the pulse duration t_1 is small compared to the natural period τ of the spring mass oscillator, the excitation is called a shock. In order to categorize all types of shock excitation, the SDOF underdamped oscillator is chosen as a standard. The shock response spectrum(SRS) is a plot of the maximum peak response of the SDOF oscillator as a function of the natural period of the oscillator. The maximum of the peaks, often labeled maximax, represents only a single point on the time response curve. It doesn't uniquely define the shock input because it is possible for two different shock pulses to have same maximum peak response. In spite of this limlitation, the SRS is a useful concept that is extensively used, especially for preliminary design.

Shock Response Spectrum (SRS) of Rectangular Pulse:

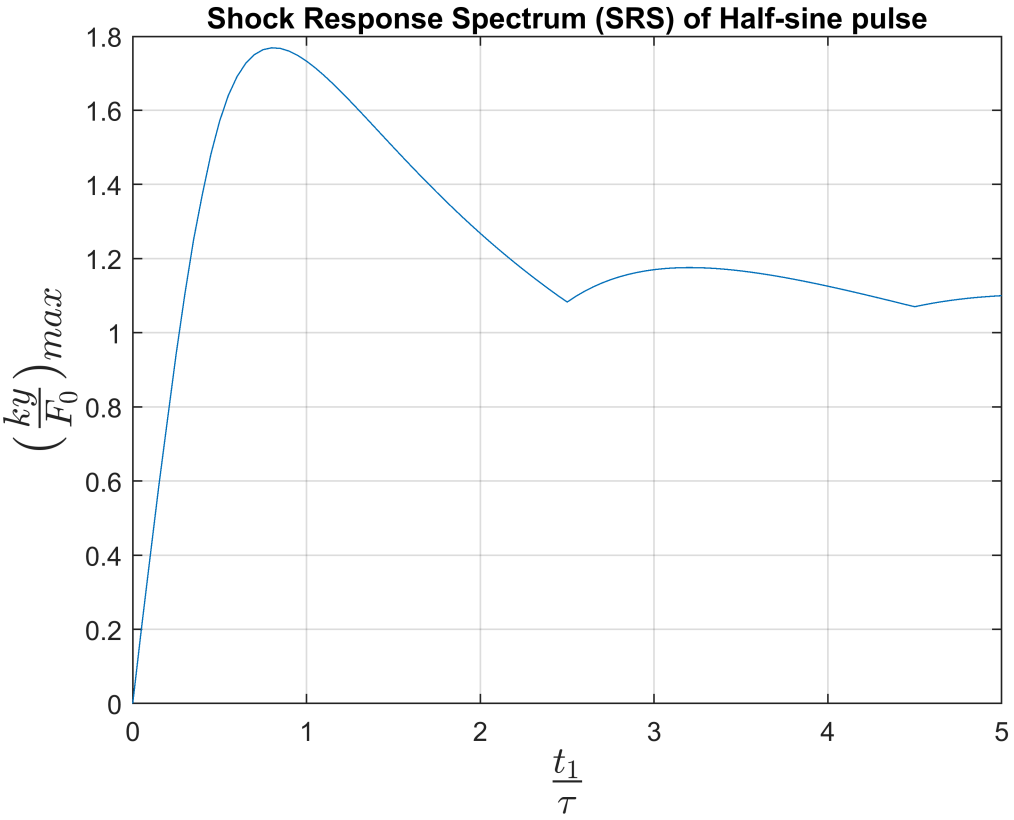
```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0;f0 = 1;
tau = (2*pi/wn); % Time period of undamped system
z = zeros(size(0:0.1:3));j = 0;
for i = 0:0.05:3
    fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*rect_pulse(t,tau*i)];
    [t,y] =  ode45(fun,0:0.01:12,[0;0]);
    z(j+1) = max((k/f0)*y(:,1));
    j = j+1;
end
figure
plot(0:0.05:3,z)
title('Shock Response Spectrum (SRS) of Rectangular pulse')
```

```
ylabel('$(\frac{ky}{F_0})_{max}$','Interpreter','latex','Fontsize',20)
xlabel('$\frac{t_1}{\tau}$','Interpreter','latex','Fontsize',20)
```



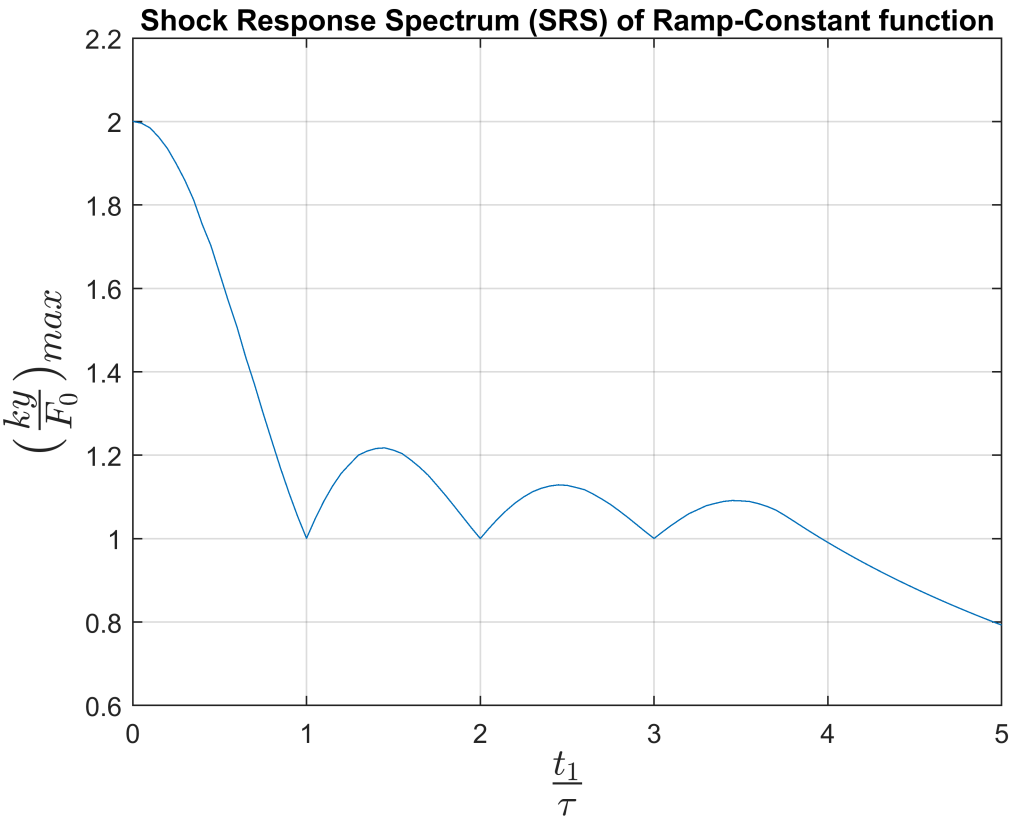
Shock Response Spectrum (SRS) of Half-Sine Pulse:

```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0;f0 = 1;
tau = (2*pi/wn); % Time period of undamped system
z = zeros(size(0:0.1:3));j = 0;
for i = 0:0.05:5
    fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*half_sine(t,tau*i)];
    [t,y] =  ode45(fun,0:0.01:12,[0;0]);
    z(j+1) = max((k/f0)*y(:,1));
    j = j+1;
end
figure
plot(0:0.05:5,z)
grid on
title('Shock Response Spectrum (SRS) of Half-sine pulse')
ylabel('$(\frac{ky}{F_0})_{max}$','Interpreter','latex','Fontsize',20)
xlabel('$\frac{t_1}{\tau}$','Interpreter','latex','Fontsize',20)
```



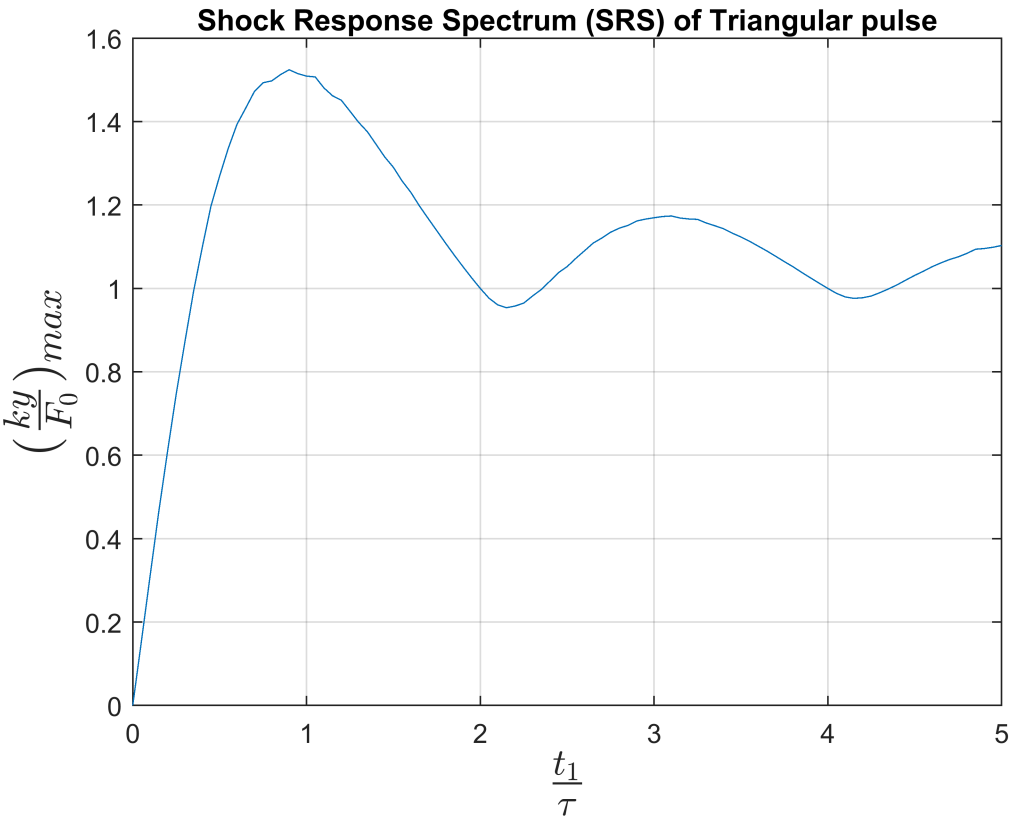
Shock Response Spectrum (SRS) of Ramp-Constant function:

```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0;f0 = 1;
tau = (2*pi/wn); % Time period of undamped system
z = zeros(size(0:0.1:3));j = 0;
for i = 0:0.05:5
    fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*rise_constant(t,tau*i)];
    [t,y] =  ode45(fun,0:0.01:12,[0;0]);
    z(j+1) = max((k/f0)*y(:,1));
    j = j+1;
end
figure
plot(0:0.05:5,z)
grid on
title('Shock Response Spectrum (SRS) of Ramp-Constant function')
ylabel('$(\frac{ky}{F_0})_{max}$','Interpreter','latex','Fontsize',20)
xlabel('$\frac{t_1}{\tau}$','Interpreter','latex','Fontsize',20)
```



Shock Response Spectrum (SRS) of Triangular Pulse:

```
m = 1; k = 4; wn = sqrt(k/m); zeta = 0;f0 = 1;
tau = (2*pi/wn); % Time period of undamped system
z = zeros(size(0:0.1:3));j = 0;
for i = 0:0.05:5
    fun = @(t,x) [x(2); (-wn^2)*x(1) + (-2*zeta*wn)*x(2)+ f0*triangular_pulse(t,tau*i/2,tau*i)]
    [t,y] =  ode45(fun,0:0.01:12,[0;0]);
    z(j+1) = max((k/f0)*y(:,1));
    j = j+1;
end
figure
plot(0:0.05:5,z)
grid on
title('Shock Response Spectrum (SRS) of Triangular pulse')
ylabel('$(\frac{ky}{F_0})_{max}$','Interpreter','latex','Fontsize',20)
xlabel('$\frac{t_1}{\tau}$','Interpreter','latex','Fontsize',20)
```



Different functions used in this article:

Unit step function:

```
function [y] = ustep(t,t1)
% Unit step function with two inputs and one output.
% t1 is the time from where unit step begins
% t is the usual time that starts from 0.
if t >= t1
    y = 1;
else
    y = 0;
end
end
```

Rise constant function:

```
function [y] = rise_constant(t,t1)
% The function rises linearly to the constant value with two inputs and one output.
% The function rises linearly till t1 and then remains constant at
% amplitude f0.
% t is the usual time that starts from 0.
% t1 is the time form where constant function begins.
if t >= t1
    y = 1;
```

```
else
    y = (1/t1)*t;
end
end
```

Rectangular pulse function:

```
function [y] = rect_pulse(t,t1)
% The function is a rectangular pulse. It starts at zero and ends at t1.
% It has an amplitude of f0.
% t is the usual time that starts from 0.
% t1 is the time where the pulse terminates.
if t > t1
    y = 0;
else
    y = 1;
end
end
```

Half-sine pulse function:

```
function [y] = half_sine(t,t1)
% The function is a half sine pulse. The peak amplitude of the pulse is 1.
% t is the usual time that starts from 0.
% t1 is the time where the pulse ends.
if t >= t1
    y = 0;
else
    y = sin(pi/t1*t);
end
end
```

Triangular pulse function:

```
function [y] = triangular_pulse(t,t1,t2)
% The function is a Triangular Pulse. It takes 3 arguments.
% It starts at zero, reaches peak at t1 and ends at t2.
% It has peak magnitude of 1.
% t is the usual time that starts from 0.
% t1 is the time where the pulse reaches its peak.
% t2 is the time where the pulse reaches zero again.
if t <= t1
    y = (1/t1)*t;
elseif (t>t1) && (t<=t2)
    y = (1/(t1-t2))*(t-t2);
else
    y = 0;
end
end
```

Last updated: 8th December, 2019.