```
%% DESCRIPTION
%
% This is a Script to solve the differential equation of a 2 DOF Vibration
% Absorber Damped
%
%% OUTPUT
%
% Formatted figure of the displacement of a 2 DOF Vibration Absorber and its
% animation.
%
%% VERSION & REFERENCE
%           Author: Shravan
%           Creation Date: 05 December 2020
%           Matlab Version: 2017a
%           Reference: Mechanical Vibrations by S.S. Rao, Chapter 9 .
%

%% Program
clear                                              % Delete Workspace
clc                                                % Clear Command Window
close all                                          % Close all figures

%% 1.) Definitions
%% 1.1) -Parameter definition
mass_body                 = 100;                        % Mass of the body [kg]
stiffness_body            = 50000;                        % Stiffness Coefficient of spring of body [N/m]
damping_body              = 0;                          % Damping coefficient of damper  of body  [Ns/m]

mass_absorber             = mass_body/20;                          % Mass of the absorber [kg]
stiffness_absorber        = mass_body*stiffness_body/mass_absorber;                    % Stiffness Coefficie
nt of spring of absorber [N/m]
damping_absorber          = 10000 ;                          % Damping coefficient of damper  of absorber  [Ns/
m]
% damping_absorber             = 2*sqrt(stiffness_body*mass_body) ;

time                      = 0:0.01:1;                     % Time [s]

x0_body                   = 3;                          % Initial Condition-displacement of body [m]
x_dot0_body               = 0;                          % Initial Condition-velocity of body [m/s]

x0_absorber               =2;                           % Initioal Condition-displacement of absorber [m].
x_dot0_absorber           = 0;                            % Initial Condition-velocity of absorber [m/s].

omega                     = 10;                         % Excited Frequency [Hz].
omega                     = 2*pi*omega;                     % Excited Frequency converted to angular frequency [rad/
sec].
force                     = 1000;                         % Excitation Force [N].

%% 2.) Computing
%% 2.1) Computing System Parameters
omega_body                = sqrt(stiffness_body/mass_body);          % Natural Frequency of Body [rad/sec].
omega_absorber            = sqrt(stiffness_absorber/mass_absorber);   % Natural Frequency of Absorber [rad/sec].
delta_static              = force/stiffness_body;                 % Static Deflection [m].
mew                       =  mass_absorber/mass_body;           % Mass Ratio
```

```
f                     = omega_absorber/omega_body;        % Ratio of Natural Frequency
omega_vector          = 0:1:100;                          % Omega Variable
g                     = omega_vector./omega_body ;        % Forced Frequency Ratio
cc                    = 2*mass_absorber*omega_body;       % Critical Damping Constant
z                     =  damping_absorber/cc;             % Damping Ratio

alpha11               = (2*z.*g).^2;
alpha22               = ( (g.^2/9   -(1)  + (mew.*g.^2).^2));
alpha33               = ( (mew*f^2*g.^2) - ( g.^2 -1).*(g.^2 - f^2)).^2;
alpha44               = ( g.^2 -f^2).^2;

%% 2.2) Computing Computation Parameters
alpha1                = stiffness_absorber -(mass_absorber*omega^2) + 1j*damping_absorber*omega;  % k2- (m2
*omega^2)+ i*c2*omega formula.
alpha2                = stiffness_body -(mass_body*omega^2) -(mass_absorber*omega^2);            % k1-(m1*om
ega^2) - (m2*omega^2)  formula.
alpha3                = (stiffness_absorber - (mass_absorber*omega^2));                          % k2- (m2*omega^2) for
mula.
alpha4                = (stiffness_body - (mass_body*omega^2));                                  % k1- (m1*omega^2) form
ula.

%% 2.3) Computing Steady State Amplitudes
X1                    = (alpha1*force)/( ((alpha4*alpha3)- (stiffness_absorber*omega^2*mass_absorber)) +1j*(ome
ga*damping_absorber*alpha2 ) );% Amplitude of body [m].
X2                    = X1*(stiffness_absorber + (1j*omega*damping_absorber)) / alpha1; % Amplitude of absorbe
r [m].

%% 2.4) Computing Displacement, Velocity and Acceleration
x_th_body             = real(X1*exp(1j*omega*time)*1000); % x0_body;           % Displacement of Body.
x_th_absorber         = real(X2*exp(1j*omega*time)*1000); % x0_absorber;       % Displacement of Absorber
.

v_th_body             = real(X1*1j*omega*exp(1j*omega*time));              % Velocity of Body.
v_th_absorber         = real(X2*1j*omega*exp(1j*omega*time));              % Velocity of Absorber.

a_th_body             = real(-X1*omega*omega*exp(1j*omega*time));          % Acceleration of Body.
a_th_absorber         = real(-X2*omega*omega*exp(1j*omega*time));          % Acceleration of Absorber.

%% 2.5) Computing Response
response_1 =sqrt(  (alpha11+alpha44)./(( alpha11.*alpha22) + alpha33));
response_2= sqrt(( alpha11 + f^4)./(( alpha11.*alpha22) + alpha33));

%% 3.) Plotting
%% 3.1) Initialize Figure
run('Initialize_Figure.m');                               % Run Initialize_Figure File.
hold on

%% 3.2) Draw Ground
run('Ground.m');                                          % Run Ground File.
hold on

%% 3.3) Draw Mass of Body
run('Mass_Body.m');                                       % Run  Mass_Body File
hold on
```

```
%% 3.4) Draw Spring of Body
run('Spring_Body.m');                                    % Run Spring_Body File.
hold on

%% 3.5) Draw Mass of Absorber
run('Mass_Absorber.m');                                  % Run Mass_Absorber File.
hold on

%% 3.6) Draw Spring of Absorber
run('Spring_Absorber.m');                                % Run Spring_Absorber File.
hold on

%% 3.7) Draw Cordinate System
run('Cordinate_System.m');                               % Run Spring_Absorber File.
hold on

% 3.8) Draw Damper
run('Damper.m');                                         % Run Damper File
hold on

% 3.9) Draw Arrow
run('Arrow.m');
hold on

%% 3.8) -Plot Animation and graph
% Initialise vectors
x_t_length_body = length(x_th_body);
x_t_length_absorber=length(x_th_absorber);

t_plot = NaN(1,x_t_length_body);
x_t_plot = NaN(1,x_t_length_body);
v_t_plot = NaN(1,x_t_length_body);

x_t_plot_absorber=NaN(1,x_t_length_absorber);
v_t_plot_absorber = NaN(1,x_t_length_absorber);


u = 1;                              % Counting variable
for k = 1 : x_t_length_body                  % Start Loop for Animation
   cla                              % clear last picture

   % Plot Graph
   t_plot(k) = time(u);                  % Build up time vector
   x_t_plot(k) = x_th_body(u);                  % Build up displacement vector
   v_t_plot(k) = v_th_body(u);                  % Build up velocitity vector
   x_t_plot_absorber(k)= x_th_absorber(u);
   v_t_plot_absorber(k)=v_th_absorber(u);
   set(graph_plot(1),'Parent',axes_graph(1), 'XData', t_plot, 'YData', x_t_plot);     % Set new Values to displacement
 graph
   set(graph_plot(2),'Parent',axes_graph(2), 'XData', t_plot, 'YData', v_t_plot);     % Set new Values to velocity grap
h

   set(graph_plot_ab(1),'Parent',axes_graph_ab(1), 'XData', t_plot, 'YData', x_t_plot_absorber);     % Set new Value
```

```
s to displacement graph
    set(graph_plot_ab(2),'Parent',axes_graph_ab(2), 'XData', t_plot, 'YData', v_t_plot_absorber);      % Set new Value
s to velocity graph

    % Plot ground system
    plotcube(axes_ani,dimension_g,position_g,clr_g)  % Plot ground

    % Plot ground mass
    position_m = [x_th_body(u) 0 0];                % Define new position of mass
    plotcube(axes_ani,dimension_m,position_m,clr_m)  % Plot mass at new position

    % Plot Absorber Mass
    position_m_ab = [x_th_absorber(u)+(7.5*delta) 0 0];                          % Initial position of the mass
    plotcube(axes_ani,dimension_m_ab,position_m_ab,clr_m_ab)                       % Initialise the mass

    % Plot Spring Body
    spring_foot = position_g(1) - dimension_g(1)/2;  % Position of the spring foot
    spring_head = x_th_body(u) + dimension_m(1)/2;        % Initial position of spring head
    x_pos_spring = phi_s/phi_max * (spring_head - spring_foot) + spring_foot;              % Calculate new x values fo
r spring
    plot3(axes_ani,x_pos_spring,y_pos_spring_1,z_pos_spring,'b','linewidth',lnwdth)          % Use plot3 function to p
lot spring
    plot3(axes_ani,x_pos_spring,y_pos_spring_2,z_pos_spring,'b','linewidth',lnwdth)          % Use plot3 function to p
lot spring
%
    % Plot Spring Absorber
    spring_foot = position_m_ab(1) - dimension_m_ab(1)/2;        % Position of the spring foot
    spring_head = x_th_body(u) + dimension_m(1)/2;        % Initial position of spring head
    x_pos_spring = phi_s/phi_max * (spring_head - spring_foot) + spring_foot;              % Calculate initial x value f
or spring
    plot3(axes_ani,x_pos_spring,y_pos_spring,z_pos_spring,'g','linewidth',lnwdth)          % Use plot3 function to plot
 spring

    % Plot Cordinate System
    plotcos(x_ar,variable_cos,clr_cos,lnwdth,fntsz,x_th_body(1),delta)
    plotcos2(x_ar,variable_cos_ab,clr_cos,lnwdth,fntsz,x_th_absorber(1)+(7.5*delta), delta);

    % Plot Damper
    damper_foot = position_m_ab(1) - dimension_m_ab(1)/2;              % Position of the damper foot
    damper_head = x_th_body(u) + dimension_m(1)/2;                     % Initial position of damper head
    plotdamper(stroke_length_max,damper_foot,damper_head,y_offset_d,clr_d,lnwdth,delta) % Plot damper

    % Plot Arrow
    arrow_head = x_th_body(u)- dimension_m(1)/2;
    arrow_foot = x_th_body(u) - dimension_m(1)/2 - max_length_a*sin(omega*time(u));
    plotarrow(arrow_foot,arrow_head,y_offset_a,clr_a,lnwdth)              % Initialise the ground

    % Set View Angle of Animation
    view(90,-90);                          % Rotate Animation

    % Plot title and label of Animation
    title(title_ani,'fontsize',fntsz)            % Title of Animation
    xlabel(xlabel_ani,'fontsize',fntsz)          % Label x-axis of Animation
```

```
    drawnow                                    % Update figures

    u = u +1;                                 % Increase counting variable by 1
end

fig1= figure('color',clr,'units',unts,'position',pos_fig,'WindowStyle','docked');
%plot(abs(omega_vector./omega_body),abs(response_0),'r','linewidth',2);
%hold on
plot(abs(g),abs(response_1),'k','linewidth',3);
xlabel('Omega/Omega1');
ylabel('X1/Delta st');
title('Response of the System');
ylim([0,50]);
```

```
%%%%%%%%%%%%%%%%%%%%%% FOR THE RUN FILES THE PROGRAMS ARE GIVEN BELOW- THEY ARE
 TO BE WRITTEN IN A NEW FILE%%%%%%%%%%%%%%%%%%%%%%%%%%%%

INITIALISE FIGURE FIGURE

%% DESCRIPTION
%
% This is a script to initialize the figures for an animation and a graph.
%
%% OUTPUT
%
% Formatted figure which can be used for an animation.
%
%% VERSION & REFERENCE
%           Author: Shravan
%           Creation Date: 04 December 2020
%           Matlab Version: 2017a
%           Reference: Mechanical Vibrations by S.S. Rao, Chapter 9 .
%

%% 1.) Definitions
%% 1.1) General Defnitions
clr = [235/255 237/255 237/255];                  % Background Color grey
unts = 'normalized';                             % Units for dimensions to normalized
lnwdth = 2;                                      % Linewidth 2
fntsz = 18;                                      % Fontsize 22
lnwdth1=1;
m=15; % A factor for animation plot.
n=2.25;
delta=abs(max(x_th_absorber));
delta1=x0_body;

%% 1.2) Positions, titles and labels
```

```
pos_fig = [0.01 0.1 0.98 0.8];                    % Position and dimension of figure
title_graph = 'Displacement and Velocity of Body v/s Time'; % Title of Body graph
title_graph_absorber='Displacement and Velocity of Absorber v/s Time'; % Title of Absorber Graph
title_ani = 'Vibration Absorber System';                  % Title of animation
xlabel_ani = 'Displacement x [mm]';                  % Name of x-axis of Animation
xlabel_graph = 'Time t [s]';                    % Name of x-axis of Graph
ylabel_graph{1} = 'Displacement x [mm]';             % Name of first y-axis of Graph
ylabel_graph{2} = 'Velocity v [m/s]';               % Name of second y-axis of Graph

%% 3.) Plot
%% 3.1) Body Graph
fig = figure('color',clr,'units',unts,'position',pos_fig,'WindowStyle','docked');   % Create a blank figure
subplot(2,2,2)                                                  % Divide figure into two subplots and select second plot
graph_plot = plot(1,1,1,1);                                          % Initialise graph at second position
set(graph_plot(1),'color', 'k','linewidth',lnwdth1);                               % Set Color and linewidth of first plot
set(gca, 'Color', clr)
set(graph_plot(2),'color', 'r','linewidth',lnwdth1);                               % Set Color and linewidth of second plot
axes_graph(1) = gca;                                            % Save first yaxis
set(axes_graph(1),'FontSize',fntsz);                                  % Set Fontsize of x and y-axes
axes_graph(2) = axes('Position',axes_graph(1).Position,...
   'YAxisLocation','right','YColor','r','Color','none','XTickLabel',[],'fontsize',fntsz);   % Create and save second yaxis
linkaxes([axes_graph(1) axes_graph(2) ],'x');                                       % Link both x Axes to each other
xlabel(axes_graph(1),xlabel_graph,'fontsize',fntsz)                                   % Label x-axis
ylabel(axes_graph(1),ylabel_graph{1},'fontsize',fntsz)                                  % Label y-axis
ylabel(axes_graph(2),ylabel_graph{2},'fontsize',fntsz)                                  % Label y-axis
title(title_graph,'fontsize',fntsz);                                       % Title of Graph
set(axes_graph(1),'Ydir','reverse')                                         % Invert y-axis
set(axes_graph(2),'Ydir','reverse')                                         % Invert y-axis
axes(axes_graph(1))
axes(axes_graph(2))
x_t_max_limit = (max(abs(x_th_body))+0.1*max(x_th_body));                              % Get Maximum of x_th_bod
y and add 5 percent
ylim(axes_graph(1),[-x_t_max_limit,x_t_max_limit]);                                   % Limit first y-axis
v_t_max_limit = (max(abs(v_th_body))+0.1*max(v_th_body));                              % Get Maximum of v_th_bod
y and add 5 percent
ylim(axes_graph(2),[-v_t_max_limit,v_t_max_limit]);                                   % Limit second y-axis
xlim(axes_graph(1),[time(1) time(end)]);                                        % Limit the time axis
grid on
grid minor

%% 3.2) Absorber Graph

subplot(2,2,4)                                                  % Divide figure into two subplots and select second plot
graph_plot_ab = plot(1,1,1,1);                                         % Initialise graph at second position
set(graph_plot_ab(1),'color', 'k','linewidth',lnwdth1);                              % Set Color and linewidth of first plot
set(gca, 'Color', clr)
set(graph_plot_ab(2),'color', 'm','linewidth',lnwdth1);                              % Set Color and linewidth of second plot
axes_graph_ab(1) = gca;                                           % Save first yaxis
set(axes_graph_ab(1),'FontSize',fntsz);                                 % Set Fontsize of x and y-axes
axes_graph_ab(2) = axes('Position',axes_graph_ab(1).Position,...
   'YAxisLocation','right','YColor','m','Color','None','XTickLabel',[],'fontsize',fntsz);   % Create and save second yaxi
s
linkaxes([axes_graph_ab(1) axes_graph_ab(2) ],'x');                                     % Link both x Axes to each other
xlabel(axes_graph_ab(1),xlabel_graph,'fontsize',fntsz)                                  % Label x-axis
```

```
ylabel(axes_graph_ab(1),ylabel_graph{1},'fontsize',fntsz)                    % Label y-axis
ylabel(axes_graph_ab(2),ylabel_graph{2},'fontsize',fntsz)                    % Label y-axis
title(title_graph_absorber,'fontsize',fntsz);                              % Title of Graph
set(axes_graph_ab(1),'Ydir','reverse')                                  % Invert y-axis
set(axes_graph_ab(2),'Ydir','reverse')                                  % Invert y-axis
axes(axes_graph_ab(1))
axes(axes_graph_ab(2))
x_t_max_limit_ab= (max(abs(x_th_absorber))+0.1*max(x_th_absorber));          % Get Maximum of x_th_
absorber and add 5 percent
v_t_max_limit_ab = (max(abs(v_th_absorber))+0.1*max(v_th_absorber));         % Get Maximum of v_th_
body and add 5 percent
ylim(axes_graph_ab(1),[-x_t_max_limit_ab,x_t_max_limit_ab]);                  % Limit first y-axis
ylim(axes_graph_ab(2),[-v_t_max_limit_ab,v_t_max_limit_ab]);                  % Limit first y-axis
xlim(axes_graph_ab(1),[time(1) time(end)]);
grid on
grid minor


%% 3.3) Animation Graph

subplot(1,2,1)                                        % Create and select first plot of figure
axes_ani = gca;                                        % Save current axes
set(axes_ani,'FontSize',fntsz);                            % Set Fontsize of Animation
set(gca, 'Color', clr)                                 % Set background color
set(axes_ani,'Xdir','reverse')                            % Invert y-axis
xlim([-m/2*delta m*delta])                           % Set the Limits of x-axis Animation (dependent of am
plitude)
ylim([-n*delta n*delta])                                        % Set the Limit of y-axis

title(title_ani,'fontsize',fntsz)                          % Set title of Animation
xlabel(xlabel_ani,'fontsize',fntsz)                          % Set label x-axis of Animation
axis('square')                                       % Axis lines with equal length
grid on
set(axes_ani,'xminorgrid','on','yminorgrid','on');
```

GROUND FILE

```
%% DESCRIPTION
%
% This is a script to draw the ground.
%
%% OUTPUT
%
% Ground is drawn in the animation.
%
%% VERSION & REFERENCE
%          Author: Shravan
%          Creation Date: 04 December 2020
%          Matlab Version: 2017a
%          Reference: Mechanical Vibrations by S.S. Rao, Chapter 9 .
%
```

```
%% 1.) Definitions
%% 1.1) General
dimension_g = [2*delta 3.75*delta 2];                    % Length, width and height of the ground
pos=m*delta-dimension_g(1)/2;
position_g = [pos 0 0];                 % Position of the ground depending on the minimum displacement of the ma
ss
clr_g = [0.9,0.45,0.1];                                   % Color of the ground

%% 3.) Plot
%% 3.) Draw ground
plotcube(axes_ani,dimension_g,position_g,clr_g)             % Initialise the ground


plotcube file

function plotcube(axis,dimension,position,clr_c)

%% DESCRIPTION
%
% Function plots a defined cube into the selected axis
%
%
%% INPUT
%       axis ... axis where you want the plot to appear
%       dimension ... length (x),width (y) ,height (z) of the cube
%       position ... position of the center of the cube
%       rotate ... rotate the cube around an axis
%       color ... Color of the cube
%       transparency ... Define transparencs of the cube
%
%% OUTPUT
%       Plot of a cube in a desired axis
%%% Programm
%% 1.) Definitions
% No definitions needed

%% 2.) Computing
%% 2.) -Calculate vertices and faces
x_c(1) = -dimension(1)/2;              % x-coordinate of the cube is half the length
x_c(2) = dimension(1)/2;             % x-coordinate of the cube is half the length
x_c(3) = dimension(1)/2;              % x-coordinate of the cube is half the length
x_c(4) = -dimension(1)/2;             % x-coordinate of the cube is half the length
x_c(5) = -dimension(1)/2;              % x-coordinate of the cube is half the length
x_c(6) = dimension(1)/2;             % x-coordinate of the cube is half the length
x_c(7) = dimension(1)/2;              % x-coordinate of the cube is half the length
x_c(8) = -dimension(1)/2;             % x-coordinate of the cube is half the length

y_c(1) = dimension(2)/2;              % x-coordinate of the cube is half the length
y_c(2) = dimension(2)/2;             % x-coordinate of the cube is half the length
y_c(3) = -dimension(2)/2;              % x-coordinate of the cube is half the length
y_c(4) = -dimension(2)/2;             % x-coordinate of the cube is half the length
y_c(5) = dimension(2)/2;              % x-coordinate of the cube is half the length
y_c(6) = dimension(2)/2;             % x-coordinate of the cube is half the length
y_c(7) = -dimension(2)/2;              % x-coordinate of the cube is half the length
```

```
y_c(8) = -dimension(2)/2;               % x-coordinate of the cube is half the length

z_c(1) = -dimension(3)/2;               % x-coordinate of the cube is half the length
z_c(2) = -dimension(3)/2;               % x-coordinate of the cube is half the length
z_c(3) = -dimension(3)/2;               % x-coordinate of the cube is half the length
z_c(4) = -dimension(3)/2;               % x-coordinate of the cube is half the length
z_c(5) = dimension(3)/2;                % x-coordinate of the cube is half the length
z_c(6) = dimension(3)/2;                % x-coordinate of the cube is half the length
z_c(7) = dimension(3)/2;                % x-coordinate of the cube is half the length
z_c(8) = dimension(3)/2;                % x-coordinate of the cube is half the length

vertices_c = [x_c(1) y_c(1) z_c(1);x_c(2) y_c(2) z_c(2);x_c(3) y_c(3) z_c(3);x_c(4) y_c(4) z_c(4);...
            x_c(5) y_c(5) z_c(5);x_c(6) y_c(6) z_c(6);x_c(7) y_c(7) z_c(7);x_c(8) y_c(8) z_c(8)];   % Define the eight
corners of the cube
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
faces_c = [4 3 2 1;2 3 7 6;3 4 8 7;1 2 6 5;7 8 5 6;1 4 8 5];            % Define the faces of the cube by four corner for
each face
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 2.) -Translate vertices
vertices_translated = [vertices_c(:,1)+position(1) vertices_c(:,2)+position(2) vertices_c(:,3)+position(3)];

%% 3.) Plot
patch(axis,'Vertices', vertices_translated, 'Faces', faces_c, 'FaceColor', clr_c,'EdgeColor', clr_c);
end
```

MASS BODY FILE

```
%% DESCRIPTION
%
% This is a script to draw the mass.
%
%% OUTPUT
%
% Mass is drawn in the animation.
%

%% 1.) Definitions
%% 1.) -General
dimension_m = [2*delta 2.5*delta 2];            % Length, width and height of the mass
position_m = [x_th_body(1) 0 0];                % Initial position of the mass
clr_m = 'r';                                    % %Color of the mass

%% 3.) Plot
%% 3.) -Draw mass
plotcube(axes_ani,dimension_m,position_m,clr_m)         % %Initialise the mass
```

MASS ABSORBER FILE

```
%% DESCRIPTION
```

```
%
% This is a script to draw the mass.
%
%% OUTPUT
%
% Mass is drawn in the animation.
%

%% 1.) Definitions
%% 1.) -General
dimension_m_ab = [1.5*delta 1.5*delta 0.2];                    % Length, width and height of the mass
position_m_ab = [x_th_absorber(1)+7.5*delta 0 0];                          % Initial position of the mass
clr_m_ab = 'm';                                                   % Color of the mass

%% 3.) Plot
%% 3.) -Draw mass
plotcube(axes_ani,dimension_m_ab,position_m_ab,clr_m_ab)                 % Initialise the mass
```

SPRING ABSORBER FILE

```
%% DESCRIPTION
%
% This is a script to initialise the spring.
%
%% OUTPUT
%
% Spring is drawn in the animation.
%
%% 1.) Definitions
%% 1.) -General

spring_number_windings = 24;                     % Number of spring windings
spring_radius = 0.1*delta;                        % Radius of spring radius
phi_max = 2*pi*spring_number_windings;           % Calculate the maximum angle of spring rotations
phi_s = 0:pi/50:phi_max;                         % Define a vector in order to calculate y and z position of the spring vertice
s
y_offset_1 =0.4*1.5*delta;                              % Spring y-offset
y_pos_spring = spring_radius * sin(phi_s) + y_offset_1;       % Calculate y position of spring vertices
z_pos_spring = spring_radius * cos(phi_s);                   % Calculate z position of spring vertices
spring_foot = position_m_ab(1) - dimension_m_ab(1)/2;         % Position of the spring foot
spring_head = x_th_body(1) + dimension_m(1)/2;         % Initial position of spring head

%% 3.) Plot
%% 3.) -Draw spring
x_pos_spring = phi_s/phi_max * (spring_head - spring_foot) + spring_foot;              % Calculate initial x value for s
pring
plot3(axes_ani,x_pos_spring,y_pos_spring,z_pos_spring,'g','linewidth',lnwdth)          % Use plot3 function to plot s
pring
```

SPRING BODY FILE

```
%% DESCRIPTION
%
% This is a script to initialise the spring.
%
%% OUTPUT
%
% Spring is drawn in the animation.
%
%% 1.) Definitions
%% 1.) -General

spring_number_windings = 24;                    % Number of spring windings
spring_radius = 0.1*delta;                      % Radius of spring radius
phi_max = 2*pi*spring_number_windings;          % Calculate the maximum angle of spring rotations
phi_s = 0:pi/50:phi_max;                        % Define a vector in order to calculate y and z position of the spring vertice
s
y_offset_1 = 0.55*2*delta;                      % Spring y-offset
y_offset_2 = -0.55*2*delta;
y_pos_spring_1 = spring_radius * sin(phi_s) + y_offset_1; % Calculate y position of spring vertices
y_pos_spring_2 = spring_radius * sin(phi_s) + y_offset_2; % Calculate y position of spring vertices
z_pos_spring = spring_radius * cos(phi_s);      % Calculate z position of spring vertices
spring_foot = position_g(1) - dimension_g(1)/2;  % Position of the spring foot
spring_head = x_th_body(1) + dimension_m(1)/2;         % Initial position of spring head

%% 3.) Plot
%% 3.) -Draw spring
x_pos_spring = phi_s/phi_max * (spring_head - spring_foot) + spring_foot;             % Calculate initial x value for s
pring
plot3(axes_ani,x_pos_spring,y_pos_spring_1,z_pos_spring,'b','linewidth',lnwdth)         % Use plot3 function to plot
 spring

plot3(axes_ani,x_pos_spring,y_pos_spring_2,z_pos_spring,'b','linewidth',lnwdth)         % Use plot3 function to plot
 spring
```

DAMPER FILE

```
%% DESCRIPTION
%
% This is a script to initialise the damper.
%
%% OUTPUT
%
% Damper is drawn in the animation.
%
%% 1.) Definitions
%% 1.) -General
clr_d = 'k';                                    % Color of the mass
y_offset_d = -0.4*1.5*delta;                                % Damper y-offset
damper_foot = position_m_ab(1) - dimension_m_ab(1)/2;              % Position of the damper foot
damper_head = x_th_body(1) + dimension_m(1)/2;                     % Initial position of damper head
stroke_length_max = abs(min(x_th_body + dimension_m(1)/2)) + damper_foot;          % Maximum stroke length
```

```
%% 3.) Plot
%% 3.) -Draw damper
plotdamper(stroke_length_max,damper_foot,damper_head,y_offset_d,clr_d,lnwdth,delta) % Plot damper
```

plotdamper file

```
function plotdamper(stroke_length_max,damper_foot,damper_head,y_offset_d,clr_d,lnwdth,delta)

%% DESCRIPTION
%
% Function plots a damper
%
%
%% INPUT
%       stroke_length_max ... Maximum stroke length from foot to head
%       damper_foot ... position of damper foot
%       damper_head ... position of damper head
%       y_offset_d ... y offset of the damper
%       clr_d ... Color of the damper
%       lnwdth ... Linewidth of the faces
%
%% OUTPUT
%       Plot of a damper
%
%% Programm
%% 1.) Definitions
% No definitions needed

%% 2.) Computing
%% 2.) -Calculate vertices and faces
x_d(1) = 0 + damper_foot;
x_d(2) = -stroke_length_max*0.02 + damper_foot;
x_d(3) = -stroke_length_max*0.02 + damper_foot;
x_d(4) = -stroke_length_max*0.02 + damper_foot;
% x_d(5) = -stroke_length_max*0.95 + damper_foot;
% x_d(6) = -stroke_length_max*0.95 + damper_foot;
x_d(5) = damper_head;
x_d(6) = damper_head;
x_d(10) = damper_head;
x_d(7) = x_d(10) + stroke_length_max*0.1;
x_d(8) = x_d(10) + stroke_length_max*0.1;
x_d(9) = x_d(10) + stroke_length_max*0.1;

y_d(1) = 0*delta + y_offset_d;
y_d(2) = 0.1*delta + y_offset_d;
y_d(3) = -0.1*delta + y_offset_d;
y_d(4) = 0*delta + y_offset_d;
y_d(5) = -0.1*delta + y_offset_d;
y_d(6) = 0.1*delta + y_offset_d;
y_d(7) = -0.09*delta + y_offset_d;
y_d(8) = 0.09*delta + y_offset_d;
y_d(9) = 0*delta + y_offset_d;
y_d(10) = 0*delta + y_offset_d;
```

```
z_d(1) = 0.1;
z_d(2) = -0.1;

vertices_d = [x_d(1) y_d(1) z_d(1);x_d(2) y_d(2) z_d(1);x_d(3) y_d(3) z_d(1);x_d(4) y_d(4) z_d(1);x_d(5) y_d(5) z
_d(1);...
            x_d(6) y_d(6) z_d(1);x_d(7) y_d(7) z_d(1);x_d(8) y_d(8) z_d(1);x_d(9) y_d(9) z_d(1);x_d(10) y_d(10) z_d
(1);...
            x_d(1) y_d(1) z_d(2);x_d(2) y_d(2) z_d(2);x_d(3) y_d(3) z_d(2);x_d(4) y_d(4) z_d(2);x_d(5) y_d(5) z_d(2
);...
            x_d(6) y_d(6) z_d(2);x_d(7) y_d(7) z_d(2);x_d(8) y_d(8) z_d(2);x_d(9) y_d(9) z_d(2);x_d(10) y_d(10) z_d
(2)];

faces = [1 4 14 11;2 3 13 12;3 5 15 13;2 6 16 12;7 8 18 17;9 10 20 19];

%% 3.) Plot
patch('Vertices', vertices_d, 'Faces', faces, 'FaceColor', clr_d,'EdgeColor', clr_d,'linewidth',lnwdth);
end
```

ARROW FILE

```
%% DESCRIPTION
%
% This is a script to draw the excitation arrow.
%
%% OUTPUT
%
% Excitation arrow is drawn in the animation.

%% 1.) Definitions
%% 1.) -General
max_length_a = 1.75*(delta);          % Define x length of the excitation arrow
clr_a = [0.1,0.1,0.1];                % Color of the mass
y_offset_a = 0;
arrow_head = x_th_body(1) - dimension_m(1)/2;
arrow_foot = x_th_body(1) - dimension_m(1)/2 - max_length_a;

%% 3.) Plot
%% 3.) -Draw ground
plotarrow(arrow_foot,arrow_head,y_offset_a,clr_a,lnwdth)          % Initialise the ground
```

plotarrow file

```
function plotarrow(arrow_foot,arrow_head,y_offset_a,clr_a,lnwdth)

%% DESCRIPTION
%
% Function plots a excitation arrow
%
```

```
%
%% INPUT
%       arrow_foot ... position of excitation arrow foot
%       arrow_head ... position of excitation arrow head
%       y_offset_a ... y offset of the excitation arrow
%       clr_a ... Color of the excitation arrow
%       lnwdth ... Linewidth of the faces
%
%% OUTPUT
%       Plot of a excitation arrow


%% 2.) Computing
%% 2.) -Calculate vertices and faces
x_a(1) = arrow_foot;
x_a(2) = arrow_foot;
x_a(3) = arrow_head - (arrow_head-arrow_foot)*0.3;
x_a(4) = arrow_head - (arrow_head-arrow_foot)*0.3;
x_a(5) = arrow_head - (arrow_head-arrow_foot)*0.3;
x_a(6) = arrow_head - (arrow_head-arrow_foot)*0.3;
x_a(7) = arrow_head;

y_a(1) = -0.05 + y_offset_a;
y_a(2) = 0.05 + y_offset_a;
y_a(3) = -0.2 + y_offset_a;
y_a(4) = -0.05 + y_offset_a;
y_a(5) = 0.05 + y_offset_a;
y_a(6) = 0.2 + y_offset_a;
y_a(7) = y_offset_a;

z_a(1) = -3;

vertices_d = [x_a(1) y_a(1) z_a(1);x_a(2) y_a(2) z_a(1);x_a(3) y_a(3) z_a(1);x_a(4) y_a(4) z_a(1);x_a(5) y_a(5) z_
a(1);...
            x_a(6) y_a(6) z_a(1);x_a(7) y_a(7) z_a(1);];

faces = [1 2 5 6 7 3 4];

%% 3.) Plot
patch('Vertices', vertices_d, 'Faces', faces, 'FaceColor', clr_a,'EdgeColor', clr_a,'linewidth',lnwdth);
end
```

CORDINATE SYSTEM FILE

```
%% DESCRIPTION
%
% This is a script to initialise the coordinate system.
%
%% OUTPUT
%
% Coordinate system is drawn in the animation.
%
%% 1.) Definitions
```

```
%% 1.) -General
x_ar = 2.5*(delta);              % Define x length of the arrow of the coordinate system
clr_cos = [0.6 0.1 0.1];                          % Color of the mass
variable_cos = 'X1';                    % Define the variable which is displayed at the coordinate system
variable_cos_ab='X2';

%% 3.) Plot
%% 3.) -Draw cos
plotcos(x_ar,variable_cos,clr_cos,lnwdth,fntsz,x_th_body(1),delta);
plotcos2(x_ar,variable_cos_ab,clr_cos,lnwdth,fntsz,x_th_absorber(1)+(7.5*delta),delta);
```

plotcos file

```
function plotcos(x_ar,variable_cos,clr_cos,lnwdth,fntsz,x_th_body,delta)

%% DESCRIPTION
%
% Function plots a coordinate system
%
%
%% INPUT
%       x_ar ... length of the arrow
%       variable_cos ... Variable of the coordinate system
%       clr_cos ... Color of the coordinate system
%       lnwdth ... Linewidth of the faces
%
%% OUTPUT
%       Plot of a coordinate system
%
%% Programm
%% 1.) Definitions
% No definitions needed

%% 2.) Computing
%% 2.) -Calculate vertices and faces
y_offset_cos = 0.8*2*delta;

x_cos(1) = x_th_body;
x_cos(2) = x_th_body;
x_cos(3) = x_th_body;
x_cos(4) = 0.6*x_ar+x_th_body;
x_cos(5) = 0.6*x_ar+x_th_body;
x_cos(6) = 1*x_ar+x_th_body;

y_cos(1) = -0.1*x_ar + y_offset_cos;
y_cos(2) = 0.1*x_ar + y_offset_cos;
y_cos(3) = 0 + y_offset_cos;
y_cos(4) = -0.1*x_ar + y_offset_cos;
y_cos(5) = 0.1*x_ar + y_offset_cos;
y_cos(6) = 0 + y_offset_cos;

z_cos(1) = 0.1;
z_cos(2) = -0.1;
```

```
vertices_cos = [x_cos(1) y_cos(1) z_cos(1);x_cos(2) y_cos(2) z_cos(1);x_cos(3) y_cos(3) z_cos(1);x_cos(4) y_cos(
4) z_cos(1);x_cos(5) y_cos(5) z_cos(1);x_cos(6) y_cos(6) z_cos(1);...
              x_cos(1) y_cos(1) z_cos(2);x_cos(2) y_cos(2) z_cos(2);x_cos(3) y_cos(3) z_cos(2);x_cos(4) y_cos(4) z_c
os(2);x_cos(5) y_cos(5) z_cos(2);x_cos(6) y_cos(6) z_cos(2)];

faces_cos = [1 2 8 7;3 6 12 9;4 6 12 10;5 6 12 11];

%% 3.) Plot
patch('Vertices', vertices_cos, 'Faces', faces_cos, 'FaceColor', clr_cos,'EdgeColor', clr_cos,'linewidth',lnwdth);


text(x_th_body, 2*delta,variable_cos,'Color',clr_cos,'FontSize',fntsz);
end
```

plotcos2 file

```
function plotcos2(x_ar,variable_cos,clr_cos,lnwdth,fntsz,x_th_absorber,delta)

%% DESCRIPTION
%
% Function plots a coordinate system
%
%
%% INPUT
%       x_ar ... length of the arrow
%       variable_cos ... Variable of the coordinate system
%       clr_cos ... Color of the coordinate system
%       lnwdth ... Linewidth of the faces
%
%% OUTPUT
%       Plot of a coordinate system
%
%% Programm
%% 1.) Definitions
% No definitions needed

%% 2.) Computing
%% 2.) -Calculate vertices and faces
y_offset_cos = 0.8*2*delta;

x_cos(1) = x_th_absorber;
x_cos(2) = x_th_absorber;
x_cos(3) = x_th_absorber;
x_cos(4) = 0.6*x_ar+x_th_absorber;
x_cos(5) = 0.6*x_ar+x_th_absorber;
x_cos(6) = 1*x_ar+x_th_absorber;

y_cos(1) = -0.1*x_ar + y_offset_cos;
y_cos(2) = 0.1*x_ar + y_offset_cos;
y_cos(3) = 0 + y_offset_cos;
y_cos(4) = -0.1*x_ar + y_offset_cos;
y_cos(5) = 0.1*x_ar + y_offset_cos;
y_cos(6) = 0 + y_offset_cos;
```

```
z_cos(1) = 0.1;
z_cos(2) = -0.1;

vertices_cos = [x_cos(1) y_cos(1) z_cos(1);x_cos(2) y_cos(2) z_cos(1);x_cos(3) y_cos(3) z_cos(1);x_cos(4) y_cos(
4) z_cos(1);x_cos(5) y_cos(5) z_cos(1);x_cos(6) y_cos(6) z_cos(1);...
            x_cos(1) y_cos(1) z_cos(2);x_cos(2) y_cos(2) z_cos(2);x_cos(3) y_cos(3) z_cos(2);x_cos(4) y_cos(4) z_c
os(2);x_cos(5) y_cos(5) z_cos(2);x_cos(6) y_cos(6) z_cos(2)];

faces_cos = [1 2 8 7;3 6 12 9;4 6 12 10;5 6 12 11];

%% 3.) Plot
patch('Vertices', vertices_cos, 'Faces', faces_cos, 'FaceColor', clr_cos,'EdgeColor', clr_cos,'linewidth',lnwdth);


text(x_th_absorber ,2*delta,variable_cos,'Color',clr_cos,'FontSize',fntsz);
end
```