AE332-MODELLING AND ANALYSIS LAB

SESSION 5

Impact and Friction

Aditya Kumar Shahi

SC21B005

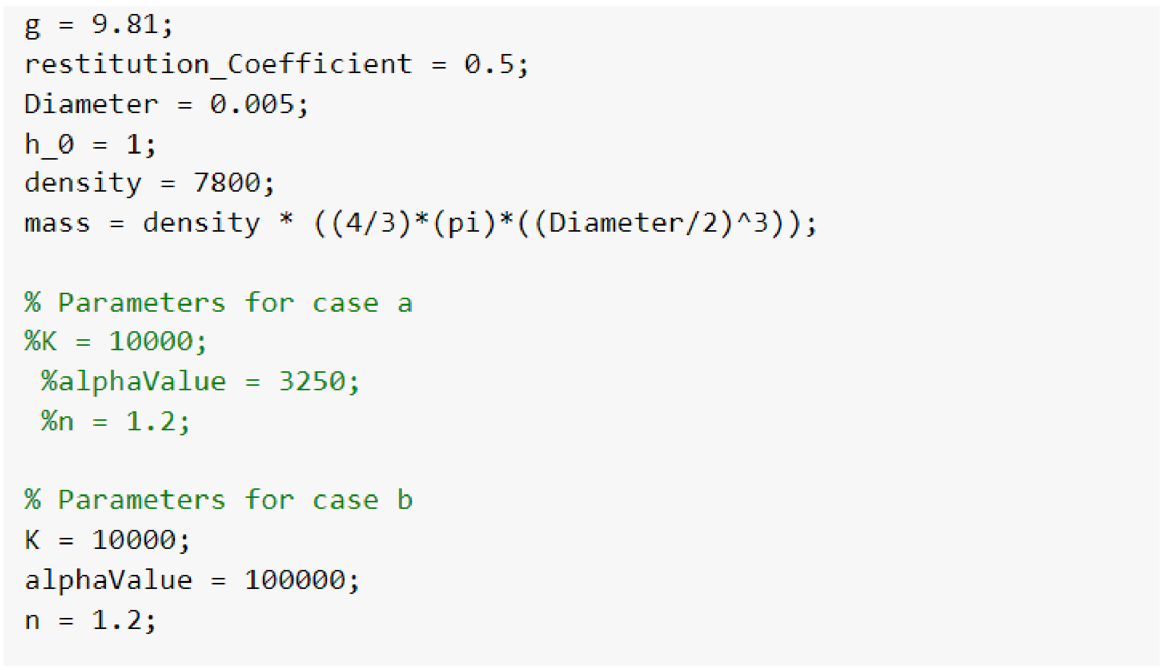
Q1) Determine the Hunt-Crossley parameters K, n, α and ADAMS parameters K, n, Cmax, δT , to give it (nearly) the same coefficient of restitution in the first bounce.

Part1: Hunt-Crossley Model

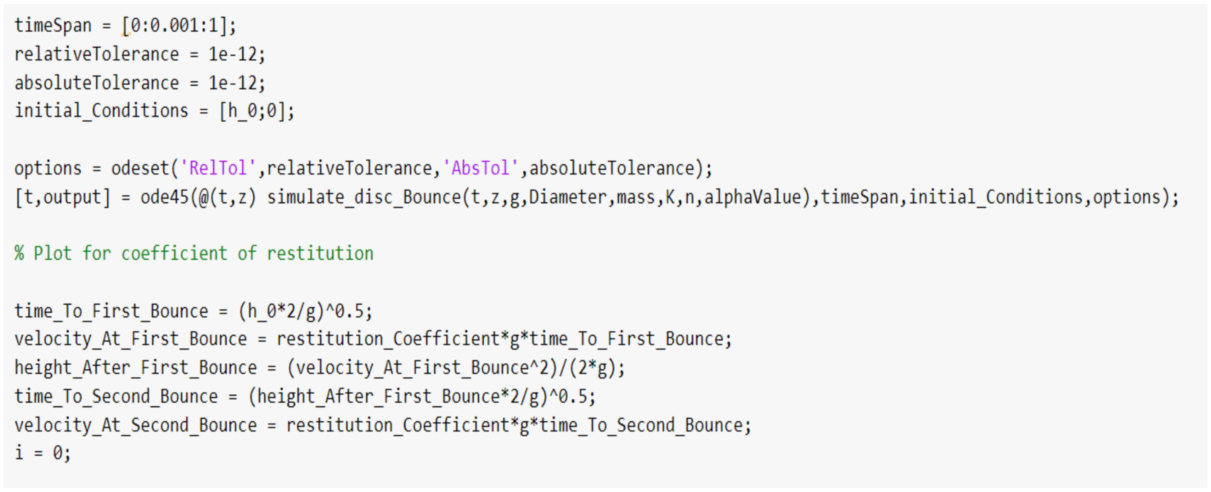
Function:

```
function res = simulate_disc_Bounce(~,z,g,Diameter,mass,K,n,alphaValue)

    delta = (Diameter/2) - z(1);
    res(1) = z(2);

    if delta > 0
        R = K*(delta^n) + alphaValue*(delta^n)*(-z(2));
    else
        R = 0;
    end

    res(2) = R/mass - g;
    res = res';
end
```

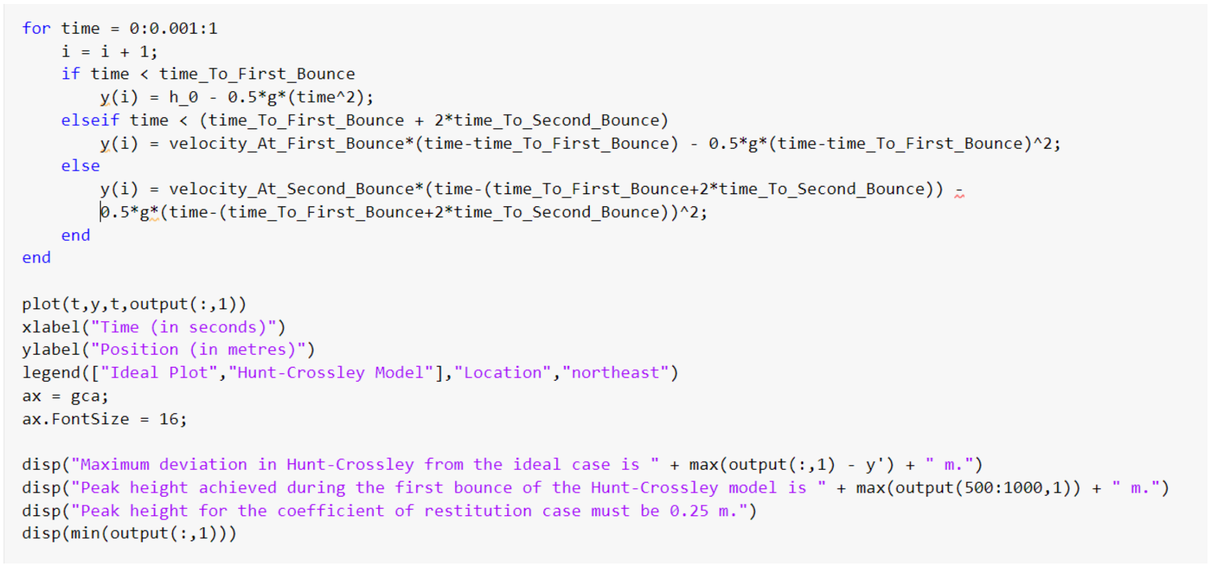
Code:



```
g = 9.81;
restitution_Coefficient = 0.5;
Diameter = 0.005;
h_0 = 1;
density = 7800;
mass = density * ((4/3)*(pi)*((Diameter/2)^3));

% Parameters for case a
%K = 10000;
 %alphaValue = 3250;
 %n = 1.2;

% Parameters for case b
K = 10000;
alphaValue = 100000;
n = 1.2;
```

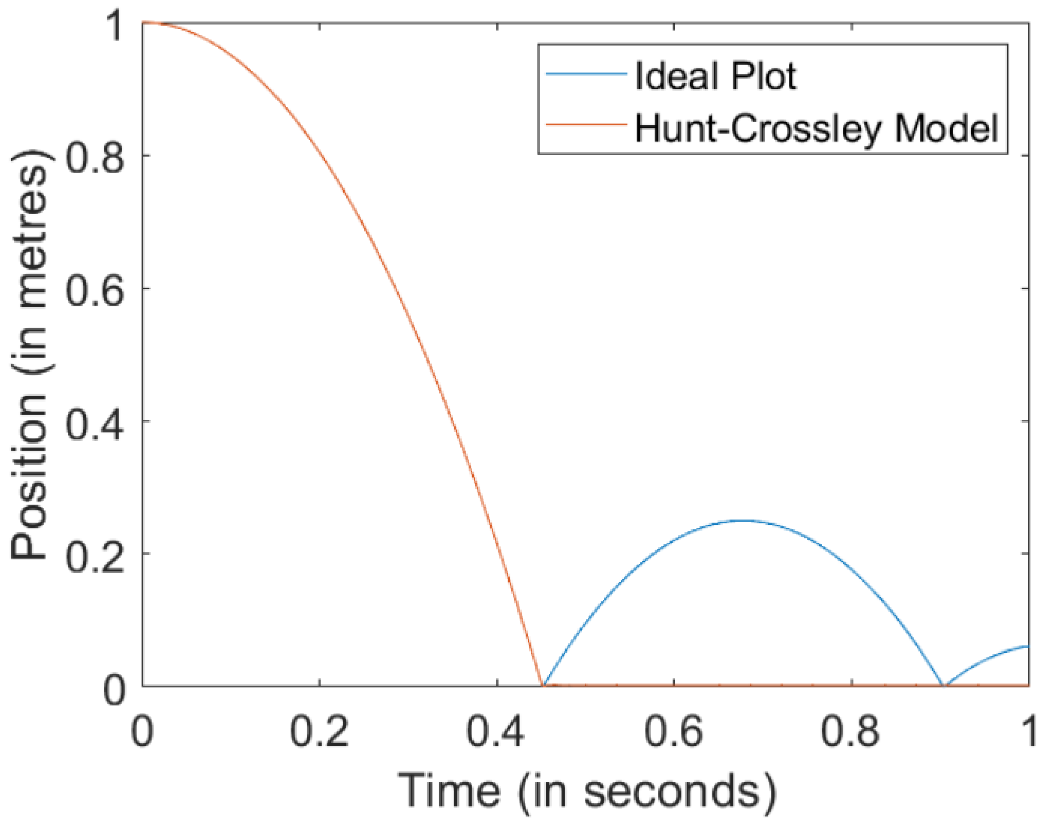


```
timeSpan = [0:0.001:1];
relativeTolerance = 1e-12;
absoluteTolerance = 1e-12;
initial_Conditions = [h_0;0];

options = odeset('RelTol',relativeTolerance,'AbsTol',absoluteTolerance);
[t,output] = ode45(@(t,z) simulate_disc_Bounce(t,z,g,Diameter,mass,K,n,alphaValue),timeSpan,initial_Conditions,options);

% Plot for coefficient of restitution

time_To_First_Bounce = (h_0*2/g)^0.5;
velocity_At_First_Bounce = restitution_Coefficient*g*time_To_First_Bounce;
height_After_First_Bounce = (velocity_At_First_Bounce^2)/(2*g);
time_To_Second_Bounce = (height_After_First_Bounce*2/g)^0.5;
velocity_At_Second_Bounce = restitution_Coefficient*g*time_To_Second_Bounce;
i = 0;
```



```
for time = 0:0.001:1
    i = i + 1;
    if time < time_To_First_Bounce
        y(i) = h_0 - 0.5*g*(time^2);
    elseif time < (time_To_First_Bounce + 2*time_To_Second_Bounce)
        y(i) = velocity_At_First_Bounce*(time-time_To_First_Bounce) - 0.5*g*(time-time_To_First_Bounce)^2;
    else
        y(i) = velocity_At_Second_Bounce*(time-(time_To_First_Bounce+2*time_To_Second_Bounce)) -
        0.5*g*(time-(time_To_First_Bounce+2*time_To_Second_Bounce))^2;
    end
end

plot(t,y,t,output(:,1))
xlabel("Time (in seconds)")
ylabel("Position (in metres)")
legend(["Ideal Plot","Hunt-Crossley Model"],"Location","northeast")
ax = gca;
ax.FontSize = 16;

disp("Maximum deviation in Hunt-Crossley from the ideal case is " + max(output(:,1) - y') + " m.")
disp("Peak height achieved during the first bounce of the Hunt-Crossley model is " + max(output(500:1000,1)) + " m.")
disp("Peak height for the coefficient of restitution case must be 0.25 m.")
disp(min(output(:,1)))
```

Graph:



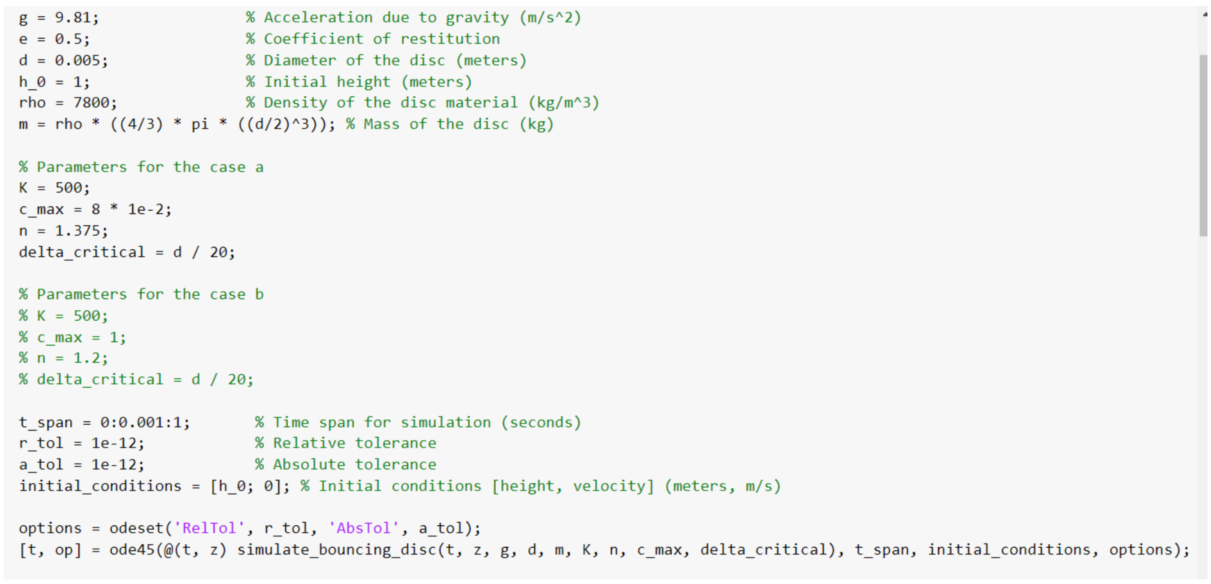
```
Maximum deviation in Hunt-Crossley from the ideal case is 0.0023897 m.
Peak height achieved during the first bounce of the Hunt-Crossley model is 0.0025289 m.
Peak height for the coefficient of restitution case must be 0.25 m.
    0.0021
```

Part 2: ADAMS model

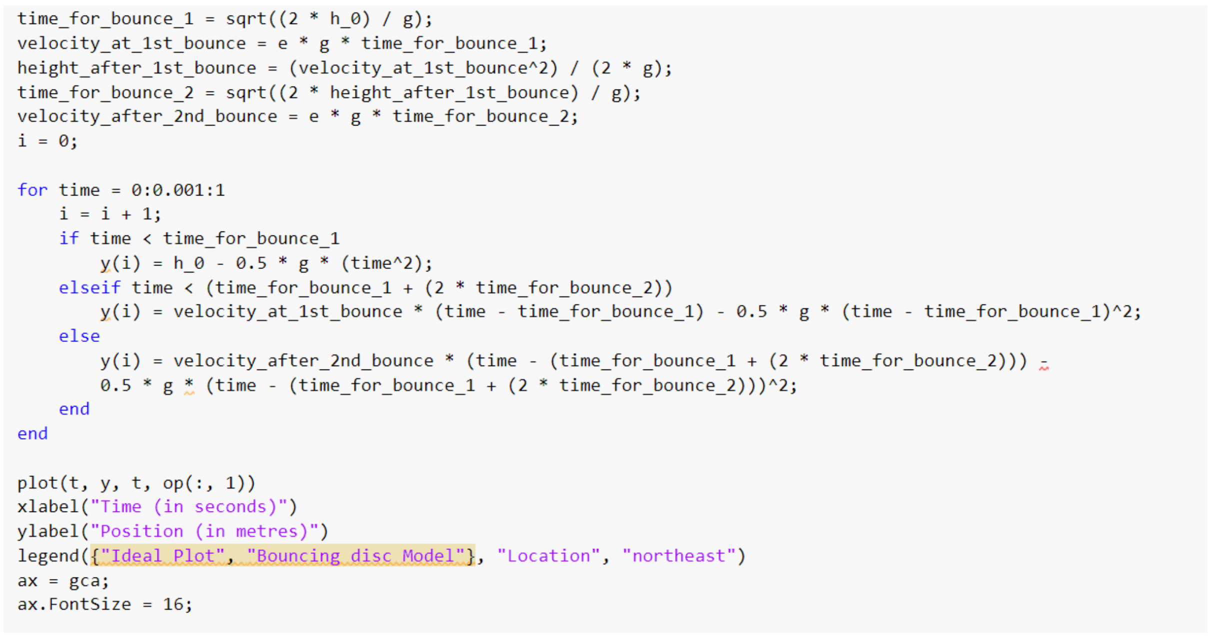
Function:

```
function res = simulate_bouncing_disc(~, z, g, d, m, K, n, c_max, delta_critical)

    delta = (d/2) - z(1);
    res(1) = z(2);

    if delta > 0
        if delta < delta_critical
            c = (c_max / 2) * ((3 * delta / delta_critical) - ((delta / delta_critical)^3));
        else
            c = c_max;
        end

        R = K * (delta^n) + c * (-z(2));
    else
        R = 0;
    end

    res(2) = R / m - g;
    res = res';
end
```

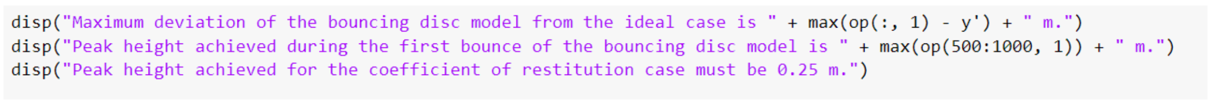
Code:



```
g = 9.81;                % Acceleration due to gravity (m/s^2)
e = 0.5;                 % Coefficient of restitution
d = 0.005;               % Diameter of the disc (meters)
h_0 = 1;                 % Initial height (meters)
rho = 7800;              % Density of the disc material (kg/m^3)
m = rho * ((4/3) * pi * ((d/2)^3)); % Mass of the disc (kg)

% Parameters for the case a
K = 500;
c_max = 8 * 1e-2;
n = 1.375;
delta_critical = d / 20;

% Parameters for the case b
% K = 500;
% c_max = 1;
% n = 1.2;
% delta_critical = d / 20;

t_span = 0:0.001:1;       % Time span for simulation (seconds)
r_tol = 1e-12;            % Relative tolerance
a_tol = 1e-12;            % Absolute tolerance
initial_conditions = [h_0; 0]; % Initial conditions [height, velocity] (meters, m/s)

options = odeset('RelTol', r_tol, 'AbsTol', a_tol);
[t, op] = ode45(@(t, z) simulate_bouncing_disc(t, z, g, d, m, K, n, c_max, delta_critical), t_span, initial_conditions, options);
```

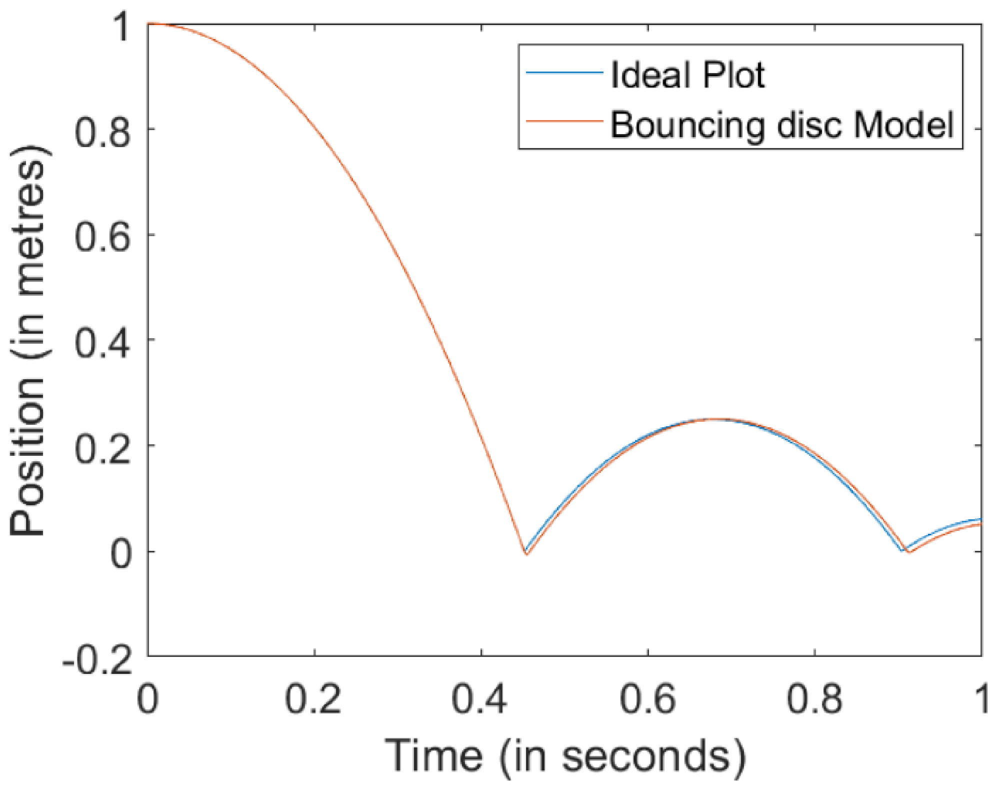


```
time_for_bounce_1 = sqrt((2 * h_0) / g);
velocity_at_1st_bounce = e * g * time_for_bounce_1;
height_after_1st_bounce = (velocity_at_1st_bounce^2) / (2 * g);
time_for_bounce_2 = sqrt((2 * height_after_1st_bounce) / g);
velocity_after_2nd_bounce = e * g * time_for_bounce_2;
i = 0;

for time = 0:0.001:1
    i = i + 1;
    if time < time_for_bounce_1
        y(i) = h_0 - 0.5 * g * (time^2);
    elseif time < (time_for_bounce_1 + (2 * time_for_bounce_2))
        y(i) = velocity_at_1st_bounce * (time - time_for_bounce_1) - 0.5 * g * (time - time_for_bounce_1)^2;
    else
        y(i) = velocity_after_2nd_bounce * (time - (time_for_bounce_1 + (2 * time_for_bounce_2))) -
        0.5 * g * (time - (time_for_bounce_1 + (2 * time_for_bounce_2)))^2;
    end
end

plot(t, y, t, op(:, 1))
xlabel("Time (in seconds)")
ylabel("Position (in metres)")
legend({"Ideal Plot", "Bouncing disc Model"}, "Location", "northeast")
ax = gca;
ax.FontSize = 16;
```



```
disp("Maximum deviation of the bouncing disc model from the ideal case is " + max(op(:, 1) - y') + " m.")
disp("Peak height achieved during the first bounce of the bouncing disc model is " + max(op(500:1000, 1)) + " m.")
disp("Peak height achieved for the coefficient of restitution case must be 0.25 m.")
```

Graph:



```
Maximum deviation of the bouncing disc model from the ideal case is 0.015262 m.
Peak height achieved during the first bounce of the bouncing disc model is 0.25052 m.
Peak height achieved for the coefficient of restitution case must be 0.25 m.
```

Q 2) Simulate a tennis ball bouncing on the ground, in planar motion. Assume the diameter to be 6.7 cm, and mass to be 58 gms. The coefficient of friction between ball and ground can be assumed to be 0.6, and coefficient of resistution when dropped from around 1 m to be 0.8 for the first bounce. Simulate the behaviour of the ball when thrown forward with (a) top spin, (b) under spin, and (c) no spin.

Function:

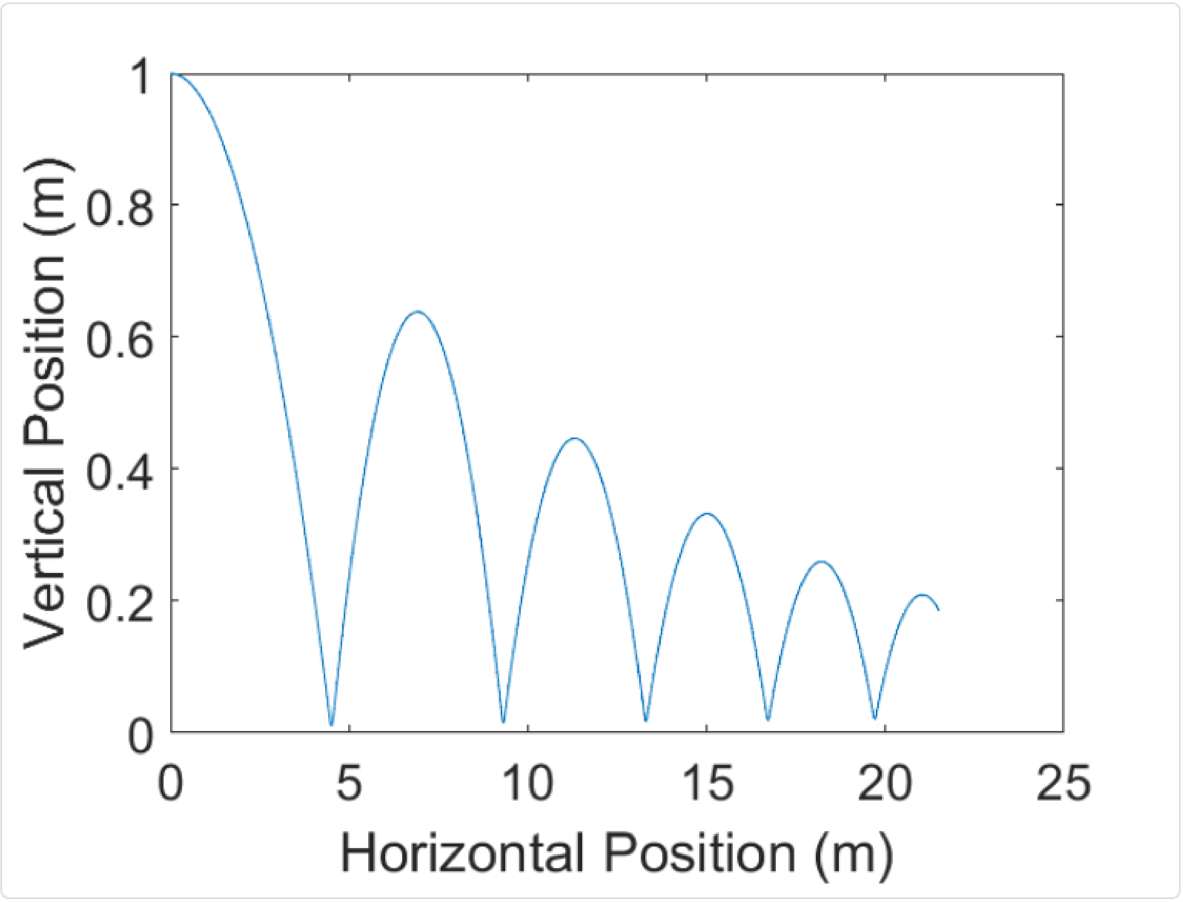
```
function res = f1(~,z,g,diameter,m,K,n,alpha,mu_max,I,s_dot)

    delta = (diameter/2) - z(2);
    res(1) = z(4);
    res(2) = z(5);
    res(3) = z(6);

    if delta > 0
        R = K*(delta^n) + alpha*(delta^n)*(-z(5));

        sdot = z(4) + (diameter*z(6)/2);
        a = (3*pi/4)/s_dot;

        mu = (mu_max*2/pi)*atan(a*sdot);

    else
        R = 0;
        mu = 0;
    end

    res(4) = -mu*R/m;
    res(5) = R/m - g;
    res(6) = -mu*R*(diameter/2)/I;
    res = res';
end
```

Code:

```
g = 9.81;
e = 0.8;
diameter = 0.067;
h_0 = 1;
m = 0.058;
I = (m/2)*(diameter/2)^2;

K = 6300;
alpha = 550;
n = 1.285;
mu_max = 0.6;
s_dot = 0.04;

%topspin
% spin = -5000;
% v = 3;

%backspin
% spin = 5000;
% v = 8;

%no spin
spin = 0;
v = 10;
```

```
t_span = [0:0.001:3];
rtol = 1e-12;
atol = 1e-12;
initial_conditions = [0;h_0;0;v;0;spin];

options = odeset('RelTol',rtol,'AbsTol',atol);
[t,op] = ode45(@(t,z) f1(t,z,g,diameter,m,K,n,alpha,mu_max,I,s_dot),t_span,initial_conditions,options);

plot(op(:,1),op(:,2))

xlabel("Horizontal Position (m)")
ylabel("Vertical Position (m)")
ax=gca;
ax.FontSize = 18;
disp("The minimum altitude achieved by the COM is "+ min(op(:,2)) +" m.")
disp("Peak height achieved during the first bounce of Hunt-Crossley model is "+ max(op(500:3000,2)) +" m.")
disp("The spin initially is "+ op(1,6) + " rad/s and after simulation, due to frictional losses it becomes "+
op(end,6) ...+ " rad/s.")
```

Graph:



```
The minimum altitude achieved by the COM is 0.011494 m.
Peak height achieved during the first bounce of Hunt-Crossley model is 0.63869 m.
The spin initially is 0 rad/s and after simulation, due to frictional losses it becomes -199.005 rad/s.
```