AE332 – Modeling and Analysis Lab

Name: Shingala Vaidik ID : SC21B054

PROBLEM **1**: Simulate a ball bouncing on the floor (purely one-dimensional motion), using the above two models. (a) Consider a steel ball of 0.005 m diameter bouncing with a coefficient of restitution of 0.5, when dropped from a height of 1 m, with zero initial velocity. 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. Plot the x vs t for the three cases (restitution, H-C, and ADAMS), for 1.0 s. (b) Determine the parameters of impact to give zero coefficient of restitution for the above ball, when released from rest from a height of 1 m. Include in the report, the values of parameters for both the above cases, and also the atol and rtol used. Also include the plot mentioned in (a) and a plot of the ball's motion in (b), simulated for 0.5 s.

1. Consider a steel ball of 0.005 m diameter bouncing with a coefficient of restitution of 0.5 when dropped from a height of 1 m, with zero initial velocity.

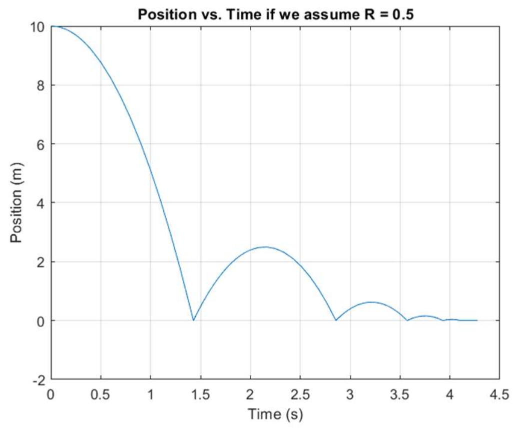
Matlab code:

```
close all;
clear;
clc
% Set up ODE options
options = odeset('events', @myEventFunc, 'refine', 10);
% Initialize time and state vectors
[t, y] = ode45(@myODEFunction, [0 10], [10; 0], options);
timePlot = t;
statePlot = y;
% Simulate for 9 more iterations
for iteration = 1:9
    [t, y] = ode45(@myODEFunction, [timePlot(end) 100], [0; -statePlot(end)*0.5],
options);
    timePlot = [timePlot; t];
    statePlot = [statePlot; y];
end
% Plot the results
figure;
```

```
plot(timePlot, statePlot(:, 1))
xlabel('Time (s)');
ylabel('Position (m)');
title('Position vs. Time if we assume R = 0.5');
grid on;
```

```
function dsdt = myODEFunction(~, s)
    gravity = 9.8;
    dsdt = [s(2); -gravity];
end
% Define the event function
function [value, isTerminal, direction] = myEventFunc(~, s)
    value = s(1);
    isTerminal = 1;
    direction = 0;
end
```

Output:



2. **(a) Determine the Hunt-Crossley parameters K, n, α (b) Determine the parameters of impact to give zero coefficient of restitution for the above ball.**

(a) After assuming some values, simulating, and comparing with the ideal model, the Hunt-Crossley parameters are found to be as follows: K = 10,000, n = 1.2, and α = 3,250.

Matlab code:

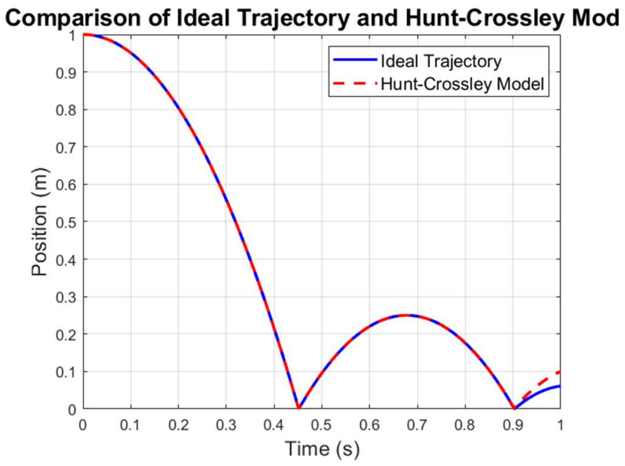
```
clc; clear;
% Constants
gravity = 9.81;
coefficientOfRestitution = 0.5;
sphereDiameter = 0.005;
initialHeight = 1;
```

```
density = 7800;
mass = density * ((4/3)*(pi)*((sphereDiameter/2)^3));
timeRange = 0:0.001:1;
relativeTolerance = 1e-12;
absoluteTolerance = 1e-12;
initialConditions = [initialHeight; 0];
timeToBounce1 = (initialHeight * 2 / gravity)^0.5;
terminalVelocity = coefficientOfRestitution * gravity * timeToBounce1;
height1 = (terminalVelocity^2) / (2 * gravity);
timeToBounce2 = (height1 * 2 / gravity)^0.5;
terminalVelocity2 = coefficientOfRestitution * gravity * timeToBounce2;
% Parameters for Case A
k = 10000;
alpha = 3250;
n = 1.2;
% Parameters for Case B
% k = 10000;
% alpha = 100000;
% n = 1.2;
options = odeset('RelTol', relativeTolerance, 'AbsTol', absoluteTolerance);
[time, output] = ode45(@(t, z) HC(t, z, gravity, sphereDiameter, mass, k, n,
alpha), timeRange, initialConditions, options);
% Plot for coefficient of restitution
count = 0;
idealPosition=zeros;
for currentTime = 0:0.001:1
    count = count + 1;
    if currentTime < timeToBounce1
        idealPosition(count) = initialHeight - 0.5 * gravity * (currentTime^2);
    elseif currentTime < (timeToBounce1 + 2 * timeToBounce2)
        idealPosition(count) = terminalVelocity * (currentTime - timeToBounce1) -
0.5 * gravity * (currentTime - timeToBounce1)^2;
    else
        idealPosition(count) = terminalVelocity2 * (currentTime - (timeToBounce1 +
2 * timeToBounce2)) - 0.5 * gravity * (currentTime - (timeToBounce1 + 2 *
timeToBounce2))^2;
    end
end
plot(time, idealPosition, 'b-', time, output(:, 1), 'r--', 'LineWidth', 2)
xlabel('Time (s)', 'FontSize', 14);
ylabel('Position (m)', 'FontSize', 14);
legend('Ideal Trajectory', 'Hunt-Crossley Model', 'Location', 'northeast',
'FontSize', 12);
title('Comparison of Ideal Trajectory and Hunt-Crossley Model', 'FontSize', 16);
grid on;
% Display results
disp('Simulation Results:');
disp(['Maximum Deviation of Modified Model from Ideal Case: ',
num2str(max(output(:, 1) - idealPosition')), ' meters.']);
disp(['Peak Height During the First Bounce of the Modified Model: ',
num2str(max(output(500:1000, 1))), ' meters.']);
disp('Peak Height for the Coefficient of Restitution Case Must Be 0.25 meters.');
disp(['Minimum Position Reached by Hunt-Crossley Model: ', num2str(min(output(:,
1))), ' meters.']);
```

```
function dzdt = HC(~, z, gravity, sphereDiameter, mass, k,n,alpha)

    delta = (sphereDiameter/2) - z(1);
    dzdt(1) = z(2);

    if delta > 0
        force = k * (delta^n) + alpha * (delta^n) * (-z(2));
    else
        force = 0;
    end

    dzdt(2) = force / mass - gravity;
    dzdt = dzdt';
end
```

Output:

```
Simulation Results:
Maximum Deviation of Modified
Model from Ideal Case: 0.038655
meters.
Peak Height During the First
Bounce of the Modified Model: 0.25
meters.
Peak Height for the Coefficient of
Restitution Case Must Be 0.25
meters.
Minimum Position Reached by Hunt-
Crossley Model: 0.0016141 meters.
```



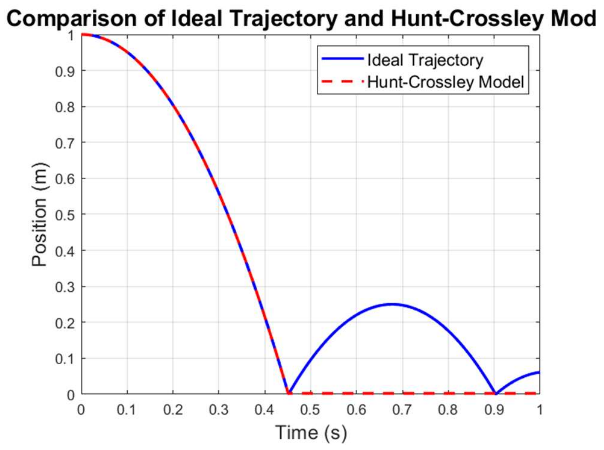
(b) Determine the parameters of impact to give zero coefficient of restitution After assuming some values, simulating, and comparing with the ideal model, the Hunt-Crossley parameters are found to be as follows: K = 10,000, n = 1.2, and α = 100,000.

Matlab code:

```
% Parameters for Case B

k = 10000;
alpha = 100000;
n = 1.2;
```

Output:

```
Simulation Results:
Maximum Deviation of Modified
Model from Ideal Case: 0.0023897
meters.
Peak Height During the First
Bounce of the Modified Model:
0.0025289 meters.
Peak Height for the Coefficient
of Restitution Case Must Be 0.25
meters.
Minimum Position Reached by Hunt-
Crossley Model: 0.0021118 meters.
```



3. (a) **Determine the ADAMS parameters K, n, Cmax, δT (b) Determine the parameters of impact to give zero coefficient of restitution for the above ball.**

(a) After assuming some values, simulating, and comparing with the ideal model, the ADAMS parameters are found to be as follows: K = 500, n = 1.375, `cmax = 8x1e-2` and δT = d/20.

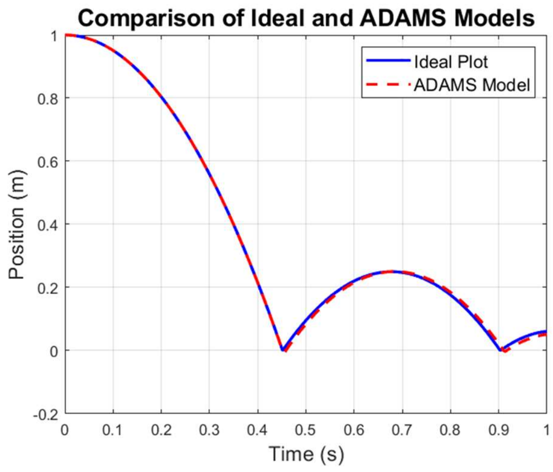
Matlab code:

```
clear;
clc;

% Constants

gravity = 9.81;
coefficientOfRestitution = 0.5;
sphereDiameter = 0.005;
initialHeight = 1;
density = 7800;
mass = density * ((4/3)*(pi)*((sphereDiameter/2)^3));

timeRange = 0:0.001:1;

relativeTolerance = 1e-12;
absoluteTolerance = 1e-12;
initialConditions = [initialHeight; 0];

timeToBounce1 = (initialHeight * 2 / gravity)^0.5;

terminalVelocity = coefficientOfRestitution * gravity * timeToBounce1;
height1 = (terminalVelocity^2) / (2 * gravity);
timeToBounce2 = (height1 * 2 / gravity)^0.5;
terminalVelocity2 = coefficientOfRestitution * gravity * timeToBounce2;

% Parameters for Case A

k = 500;
cmax = 8*1e-2;
n = 1.375;
criticalDelta = sphereDiameter/20;


% Parameters for Case B
% k = 500;
% cmax = 1;
% n = 1.2;
% criticalDelta = sphereDiameter/20;
options = odeset('RelTol', relativeTolerance, 'AbsTol', absoluteTolerance);
[t, output] = ode45(@(t, z) adamsModel(t, z, gravity, sphereDiameter, mass, k,
n, cmax, criticalDelta), timeRange, initialConditions, options);
% Plot for coefficient of restitution
count = 0;
idealPosition=zeros;
for time = 0:0.001:1
    count = count + 1;
    if time < timeToBounce1
        idealPosition(count) = initialHeight - 0.5 * gravity * (time^2);
    elseif time < (timeToBounce1 + 2 * timeToBounce2)
        idealPosition(count) = terminalVelocity * (time - timeToBounce1) - 0.5
* gravity * (time - timeToBounce1)^2;
    else
        idealPosition(count) = terminalVelocity2 * (time - (timeToBounce1 + 2
* timeToBounce2)) - 0.5 * gravity * (time - (timeToBounce1 + 2 *
timeToBounce2))^2;
    end
end


% Plot the results
figure;
plot(t, idealPosition, 'b-', t, output(:, 1), 'r--', 'LineWidth', 2);
xlabel('Time (s)', 'FontSize', 14);
ylabel('Position (m)', 'FontSize', 14);
legend('Ideal Plot', 'ADAMS Model', 'Location', 'northeast', 'FontSize', 12);
title('Comparison of Ideal and ADAMS Models', 'FontSize', 16);
grid on;
% Display results
disp('Simulation Results:');
disp(['Maximum Deviation of Modified Model from Ideal Case: ',
num2str(max(output(:, 1) - idealPosition')), ' meters.']);
disp(['Peak Height During the First Bounce of the Modified Model: ',
num2str(max(output(500:1000, 1))), ' meters.']);
disp('Peak Height for the Coefficient of Restitution Case Must Be 0.25
meters');
disp(['Minimum Position Reached by ADAMS Model: ', num2str(min(output(:, 1))),
' meters.']);
```

```
function dzdt = adamsModel(~, z, gravity, sphereDiameter, mass, k, n, cmax,
criticalDelta)
    delta = (sphereDiameter/2) - z(1);
    dzdt(1) = z(2);
    if delta > 0
        if delta < criticalDelta
            c = (cmax/2) * ((3 * delta / criticalDelta) - ((delta /
criticalDelta)^3));
        else
            c = cmax;
        end

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

    dzdt(2) = R / mass - gravity;
    dzdt = dzdt';
end
```

Output:

```
Simulation Results:
Maximum Deviation of Modified
Model from Ideal Case: 0.015262
meters.
Peak Height During the First
Bounce of the Modified Model:
0.25052 meters.
Peak Height for the Coefficient of
Restitution Case Must Be 0.25
meters.
Minimum Position Reached by ADAMS
Model: -0.0061762 meters.
```



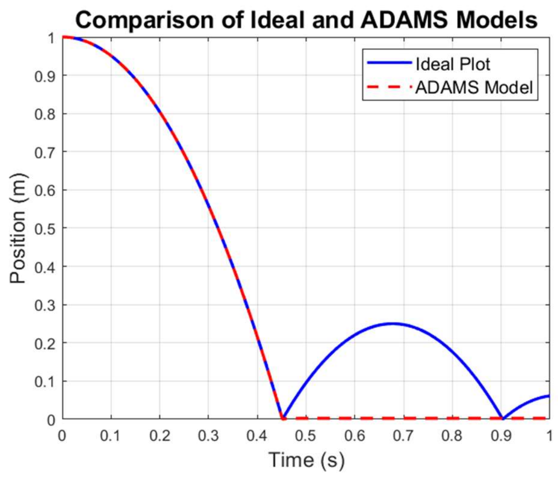
(b) Determine the parameters of impact to give zero coefficient of restitution After assuming some values, simulating, and comparing with the ideal model, the ADAMS parameters are found to be as follows: K = 500, n = 1, `cmax = 1` and δT = d/20.

Matlab code:

```
% Parameters for Case B
k = 500;
cmax = 1;
n = 1.2;
criticalDelta = sphereDiameter/20;
```

Output:

```
Simulation Results:
Maximum Deviation of Modified
Model from Ideal Case: 0.0023271
meters.
Peak Height During the First
Bounce of the Modified Model:
0.0024318 meters.
Peak Height for the Coefficient
of Restitution Case Must Be 0.25
meters.
Minimum Position Reached by ADAMS
Model: 0.00052071 meters.
```



<u>PROBELM 2:</u> 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.

Matlab code:

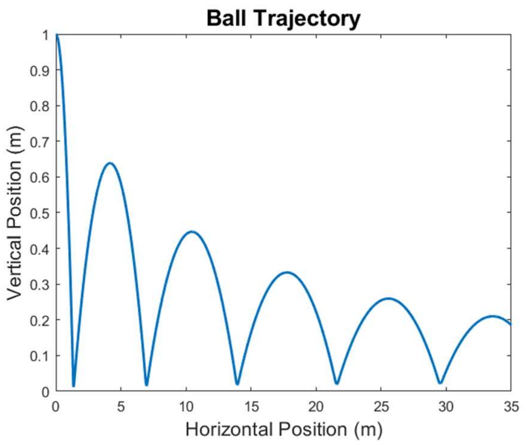
```matlab
clc;
clear;

% Constants

gravity = 9.81;
coefficientOfRestitution = 0.8;
diameter = 0.067;
initialHeight = 1;
mass = 0.058;
momentOfInertia = (mass / 2) * (diameter / 2)^2;

% Hunt-Crossley Model Parameters

k = 6300;
alpha = 550;
n = 1.285;
frictionCoefficientMax = 0.6;
slidingVelocityCritical = 0.04;

% Initial Conditions

% Set the appropriate values for topspin, backspin, or no spin
% topspin: spin = -5000, v = 3
% backspin: spin = 5000, v = 8
% no spin: spin = 0, v = 10
spin = 0;
```

```
initialVelocity = 10;


timeRange = 0:0.001:3;
relativeTolerance = 1e-12;
absoluteTolerance = 1e-12;
initialConditions = [0; initialHeight; 0; initialVelocity; 0; spin];
% Solve the ODE
options = odeset('RelTol', relativeTolerance, 'AbsTol', absoluteTolerance);
[t, op] = ode45(@(t, z) ballMotionODE(t, z, gravity, diameter, mass, k, n,
alpha, frictionCoefficientMax, momentOfInertia, slidingVelocityCritical),
timeRange, initialConditions, options);
% Plot the trajectory
figure;
plot(op(:,1), op(:,2), 'LineWidth', 2);
xlabel('Horizontal Position (m)', 'FontSize', 14);
ylabel('Vertical Position (m)', 'FontSize', 14);
title('Ball Trajectory', 'FontSize', 16);
% Display results
minimumAltitude = min(op(:,2));
peakHeight = max(op(500:3000,2));
initialSpin = initialConditions(6);
finalSpin = op(end,6);
disp('Simulation Results:');
disp(['Minimum Altitude Achieved by the Center of Mass: ',
num2str(minimumAltitude), ' meters']);
disp(['Peak Height During the First Bounce (Hunt-Crossley Model): ',
num2str(peakHeight), ' meters']);
disp(['Initial Spin: ', num2str(initialSpin), ' rad/s']);
disp(['Final Spin (After Frictional Losses): ', num2str(finalSpin), '
rad/s']);
```

```
function dzdt = ballMotionODE(~, z, gravity, diameter, mass, k, n, alpha,
frictionCoefficientMax, momentOfInertia, slidingVelocityCritical)
    delta = (diameter / 2) - z(2);
    dzdt(1) = z(4);
    dzdt(2) = z(5);
    dzdt(3) = z(6);
    if delta > 0
        R = k * (delta^n) + alpha * (delta^n) * (-z(5));

        slidingVelocity = z(4) + (diameter * z(6) / 2);
        a = (3 * pi / 4) / slidingVelocityCritical;
        frictionCoefficient = (frictionCoefficientMax * 2 / pi) * atan(a *
slidingVelocity);
    else
        R = 0;
        frictionCoefficient = 0;
    end


    dzdt(4) = -frictionCoefficient * R / mass;
    dzdt(5) = R / mass - gravity;
    dzdt(6) = -frictionCoefficient * R * (diameter / 2) / momentOfInertia;
    dzdt = dzdt';
end
```

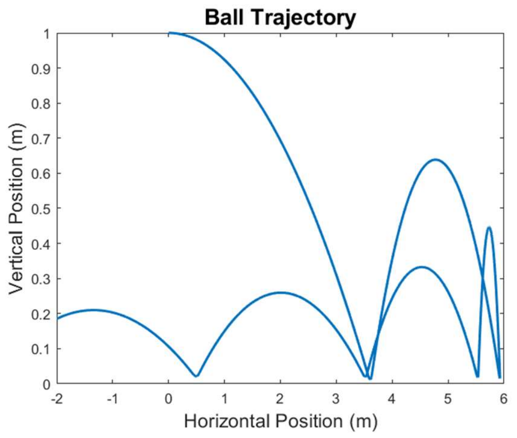
(a) Top spin:

```
Simulation Results:
Minimum Altitude Achieved by the
Center of Mass: 0.011494 meters
Peak Height During the First Bounce
(Hunt-Crossley Model): 0.63869 meters
Initial Spin: -5000 rad/s
Final Spin (After Frictional Losses):
-3970.674 rad/s
```



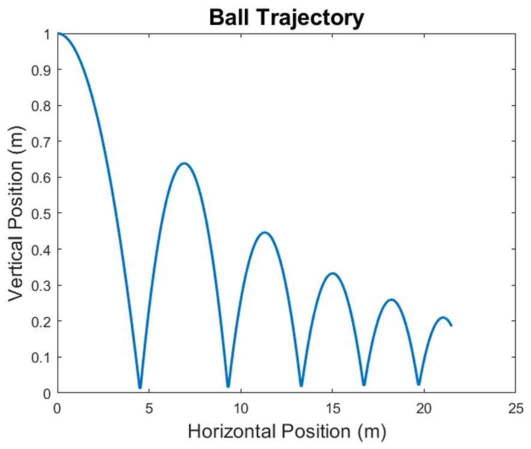
(b) Back spin:

```
Simulation Results:
Minimum Altitude Achieved by the
Center of Mass: 0.011494 meters
Peak Height During the First Bounce
(Hunt-Crossley Model): 0.63869 meters
Initial Spin: 5000 rad/s
Final Spin (After Frictional Losses):
3970.6679 rad/s
```



(c) No spin:

```
Simulation Results:
Minimum Altitude Achieved by the
Center of Mass: 0.011494 meters
Peak Height During the First Bounce
(Hunt-Crossley Model): 0.63869 meters
Initial Spin: 0 rad/s
Final Spin (After Frictional Losses):
-199.005 rad/s
```


Ball Trajectory
Vertical Position (m)
Horizontal Position (m)