```matlab
function res = PingPongIteration10()
%%  INITIAL VALUES
    %Constants
    C = .5;                 %Coefficient of drag
    A = .00125663706;       %Cross-sectional area (m^2)
    p = 1.3;                %Air density
    g = -9.81;              %Acceleration due to gravity (m/s^2)
    m = .145;               %Mass (kg)
    r = .02;                %Radius (m)
    I = 6.045e-7;           %Moment of inertia (kg*m^2)
    energyLoss = .9;        %Energy loss coefficient

    %Ball Characteristics
    xi = 0;                 %Initial x position
    yi = .8;                %Initial y position
    Vxi = 5;                %Initial x velocity
    Vyi = 2;                %Initial y velocity

    ey = .77;            % coefficient of restitution (vertical)
    ex = -.3;            % coefficient of restitution (horizontal)
    a = 2/5;             % 2/5 for uniform spheres

    %Table Characteristics in meters
    ground = 0;
    tableLength = 2.74;
    tableWidth = 1.525;
    tableHeight = .76;
    netHeight = .1525;
    tableX = [ground, tableLength];
    tableY = [tableHeight, tableHeight];
    netX = [tableLength/2, tableLength/2];
    netY = [tableHeight, tableHeight+netHeight];

%%  ODE45 EVENTS
    %Stop ode45 when the ball hits the table height
    options =  odeset('Events', @events);
    function [value, isterminal, direction] = events(t,W)
        value = W(2)-.76
        isterminal = 1;
        direction = -1;
    end

%% ODE INTEGRATION plots
    function res = odewithspin(spin)
        [T0 , M0 ] = ode45(@PingPongSimWithMag, linspace(0,2), [xi, yi, Vxi, Vyi, spin], options
        );   %Before bounce

        %calculate stuff after bounce
        vx1 = M0(end,3);
        vy1 = M0(end,4);
        w1 = M0(end,5);

        vy2 = -ey*vy1;
        vx2 = ((1-a*ex)*vx1 + a*(1+ex)*r*w1)/(1+a);
```

```
        w2 = ((1+ex)*vx1 + a*(1+ex)*r*w1)/(r*(1+a));

        [T01, M01] = ode45(@PingPongSimWithMag, linspace(0,2), [M0(end, 1), M0(end,2),vx2,vy2,w2
        ],options); % after bounce
        pT = [T0; T01+.4265];
        px = [M0(:,1); M01(:,1)];
        py = [M0(:,2); M01(:,2)];
        pVx = [M0(:,3); M01(:,3)];
        pVy = [M0(:,4); M01(:,4)];
        pSpin = [M0(:,5); M01(:,5)];
        res = [pT, px, py, pVx, pVy, pSpin];
    end

    function res = odewithspinandyvelocity(spin, VelYi)
        [T0 , M0 ] = ode45(@PingPongSimWithMag, linspace(0,2), [xi, yi, Vxi, VelYi, spin],
        options);   %Before bounce

        %calculate stuff after bounce
        vx1 = M0(end,3);
        vy1 = M0(end,4);
        w1 = M0(end,5);

        vy2 = -ey*vy1;
        vx2 = ((1-a*ex)*vx1 + a*(1+ex)*r*w1)/(1+a);
        w2 = ((1+ex)*vx1 + a*(1+ex)*r*w1)/(r*(1+a));

        [T01, M01] = ode45(@PingPongSimWithMag, linspace(0,2), [M0(end, 1), M0(end,2),vx2,vy2,w2
        ],options); % after bounce
        pT = [T0; T01+.4265];
        px = [M0(:,1); M01(:,1)];
        py = [M0(:,2); M01(:,2)];
        pVx = [M0(:,3); M01(:,3)];
        pVy = [M0(:,4); M01(:,4)];
        pSpin = [M0(:,5); M01(:,5)];
        res = [px(end)];
    end

    function res = odewithspinandxvelocity(spin, VelXi)
        [T0 , M0 ] = ode45(@PingPongSimWithMag, linspace(0,2), [xi, yi, VelXi, Vyi, spin],
        options);   %Before bounce

        %calculate stuff after bounce
        vx1 = M0(end,3);
        vy1 = M0(end,4);
        w1 = M0(end,5);

        vy2 = -ey*vy1;
        vx2 = ((1-a*ex)*vx1 + a*(1+ex)*r*w1)/(1+a);
        w2 = ((1+ex)*vx1 + a*(1+ex)*r*w1)/(r*(1+a));

        [T01, M01] = ode45(@PingPongSimWithMag, linspace(0,2), [M0(end, 1), M0(end,2),vx2,vy2,w2
        ],options); % after bounce
        pT = [T0; T01+.4265];
        px = [M0(:,1); M01(:,1)];
```

```
        py = [M0(:,2); M01(:,2)];
        pVx = [M0(:,3); M01(:,3)];
        pVy = [M0(:,4); M01(:,4)];
        pSpin = [M0(:,5); M01(:,5)];
        res = [px(end)];
    end

    function res = odewithyvelocity(VelYi)
        spin = 500;
        [T0 , M0 ] = ode45(@PingPongSimWithMag, linspace(0,2), [xi, yi, Vxi, VelYi, spin],
        options);   %Before bounce

        %calculate stuff after bounce
        vx1 = M0(end,3);
        vy1 = M0(end,4);
        w1 = M0(end,5);
        disp(M0(end,2))

        vy2 = -ey*vy1;
        vx2 = ((1-a*ex)*vx1 + a*(1+ex)*r*w1)/(1+a);
        w2 = ((1+ex)*vx1 + a*(1+ex)*r*w1)/(r*(1+a));

        [T01, M01] = ode45(@PingPongSimWithMag, linspace(0,2), [M0(end, 1), M0(end,2),vx2,vy2,w2
        ],options); % after bounce
        pT = [T0; T01+.4265];
        px = [M0(:,1); M01(:,1)];
        py = [M0(:,2); M01(:,2)];
        pVx = [M0(:,3); M01(:,3)];
        pVy = [M0(:,4); M01(:,4)];
        pSpin = [M0(:,5); M01(:,5)];
        res = [pT, px, py, pVx, pVy, pSpin];
    end

%     p1=odewithspin(628);    %top
%     p2=odewithspin(0);      %no
%     p3=odewithspin(-628);   %back

%%  RUN THIS
    clf
    %spinAndYVelocityEffect()
    %spinAndXVelocityEffect()
    %spinEffect()
    %plotAllCharacteristics(p1)
    %plotAllCharacteristics(p2)
    %plotPosition()
    plotVaryingYVelocity()
    %plotSpin()
    %plotConservation(p1(:,1), p1(:,2), p1(:,3), p1(:,4), p1(:,5))
    %plotConservation(p2(:,1), p2(:,2), p2(:,3), p2(:,4), p2(:,5))

%%  PLOTTING
    function res = plotAllCharacteristics(characteristics)
        T = characteristics(:, 1);
        X = characteristics(:, 2);
```

```
    Y = characteristics(:, 3);
    Vx = characteristics(:, 4);
    Vy = characteristics(:, 5);
    spin = characteristics(:, 6);

    %Y Position vs. Time
    subplot(3, 1, 1)
    hold on
    plot(T, Y, 'linewidth', 2)
    size = 12;
    title('Ping Pong Ball Position', 'fontsize', size+1)
    xlabel('X Position (m)', 'fontsize', size)
    ylabel('Y Position (m)', 'fontsize', size)

    %Spin
    subplot(3, 1, 2)
    hold on
    plot(T, spin, 'g', 'linewidth', 2)
    title('Ping Pong Ball Spin', 'fontsize', size+1)
    xlabel('Time(s)', 'fontsize', size)
    ylabel('Amount of Spin (rad/s)', 'fontsize', size)

    %Energy
    V1 = sqrt(Vx.^2 + Vy.^2);
    grav_potential = -m * g * Y;
    kinetic = .5 * m * V1.^2;
    total = grav_potential + kinetic;

    subplot(3, 1, 3)
    hold on
    plot(T, [grav_potential kinetic total], 'linewidth', 2)
    title('Energy Conservation: With Drag, With Bounce', 'fontsize', size+1)
    xlabel('Time', 'fontsize', size)
    ylabel('Energy', 'fontsize', size)
    legend('gravity', 'kinetic', 'total', 'location', 'NORTHWEST')
end

function res = plotPosition()
    hold on
    plot(p1(:,2),p1(:,3), 'r', 'linewidth', 2)            %Top Spin
    plot(p2(:,2),p2(:,3), 'b', 'linewidth', 2)            %No Spin
    plot(p3(:,2),p3(:,3), 'g', 'linewidth', 2)            %Back Spin

    plot(tableX, tableY, 'm', 'linewidth', 2)            %Table
    plot(netX, netY, 'm', 'linewidth', 2)                %Net

    size = 12;
    title('Table Tennis Ball Trajectory with Varying Spin', 'fontsize', size+1)
    xlabel('X Position (m)', 'fontsize', size)
    ylabel('Y Position (m)', 'fontsize', size)
    legend('Top Spin', 'No Spin', 'Back Spin')
    xlim([0, 4.5])
    ylim([.75, 1.05])
end
```

```
    function res = plotVaryingYVelocity()
        yplot1 = odewithyvelocity(15);
%         yplot2 = odewithyvelocity(15);
%         yplot3 = odewithyvelocity(20);

        hold on
        size = 12;
        plot(yplot1(:, 2), yplot1(:, 3), 'b', 'linewidth', 2)
%         plot(yplot2(:, 2), yplot2(:, 3), 'g', 'linewidth', 2)
%         plot(yplot3(:, 2), yplot3(:, 3), 'r', 'linewidth', 2)
        title('Ping Pong Ball Positions With Varying Y Velocity')
        xlabel('X Position (m)', 'fontsize', size)
        ylabel('Y Position (m)', 'fontsize', size)
        legend('1 m/s Y Velocity', '2 m/s Y Velocity', '3 m/s Y Velocity')
    end

    function res = plotSpin()
        figure
        hold on
        plot(p1(:,2),p1(:,6), 'r', 'linewidth', 2)           %Top Spin
        plot(p2(:,2),p2(:,6), 'b', 'linewidth', 2)           %No Spin
        plot(p3(:,2),p3(:,6), 'g', 'linewidth', 2)           %Back Spin

        size = 12;
        title('Table Tennis Ball Spin', 'fontsize', size+1)
        xlabel('Time(s)', 'fontsize', size)
        ylabel('Amount of Spin (rad/s)', 'fontsize', size)
        legend('Top Spin', 'No Spin', 'Back Spin')
    end

    function plotConservation(T, X, Y, Vx, Vy)
        figure
        hold on

        %w = rps * 2 * pi;
        V1 = sqrt(Vx.^2 + Vy.^2);
        grav_potential = -m * g * Y;
        kinetic = .5 * m * V1.^2;
        %rotational = -.5 * I * w.^2;

        size = 12;
        total = grav_potential + kinetic;
        plot(T, [grav_potential kinetic total], 'linewidth', 2)
        legend('Gravity', 'Kinetic', 'Total')
        xlabel('Time', 'fontsize', size)
        ylabel('Energy', 'fontsize', size)
        title('Energy Conservation', 'fontsize', size+1)
    end

    function spinEffect()
        noSpin = odewithspin(0);
        default = noSpin(2, end);
        for i=1:(628*2)
```

```
            current = odewithspin(i-628);
            xFinal(i) = current(end, 2);
            spinAmount(i) = i-628;
        end

        disp(xFinal)
        size = 12;
        plot(spinAmount, xFinal, '-', 'linewidth', 2)
        title('Effects of Spin on Delta X', 'fontsize', size+1)
        xlabel('Spin (rad/s)', 'fontsize', size)
        ylabel('Total X Distance (m)', 'fontsize', size)
    end

    function spinAndYVelocityEffect()
        yVelocity = linspace(0, 40, 75);
        spinAmount = linspace(-628, 628, 75);
        efficiency = 0;
        for j=1:length(yVelocity)
            for i=1:length(spinAmount)
                efficiency(j, i) = odewithspinandyvelocity(spinAmount(i), yVelocity(j));
            end
        end

        size = 12;
        pcolor(spinAmount, yVelocity, efficiency)
        %contourf(spinAmount, yVelocity, efficiency)
        colorbar()
        title('Effects of Spin and Initial Y Velocity on Delta X', 'fontsize', size+1)
        xlabel('Amount of Spin (rad/s)')
        ylabel('Initial Y Velocity (m/s)', 'fontsize', size)
    end

    function spinAndXVelocityEffect()
        xVelocity = linspace(0, 5, 75);
        spinAmount = linspace(-628, 628, 75);
        efficiency = 0;
        for j=1:length(xVelocity)
            for i=1:length(spinAmount)
                efficiency(j, i) = odewithspinandxvelocity(spinAmount(i), xVelocity(j));
            end
        end

        size = 12;
        pcolor(spinAmount, xVelocity, efficiency)
        %contourf(spinAmount, yVelocity, efficiency)
        colorbar()
        title('Effects of Spin and Initial X Velocity on Delta X', 'fontsize', size+1)
        xlabel('Amount of Spin (rad/s)')
        ylabel('Initial X Velocity (m/s)', 'fontsize', size)
    end

%%  SIMULATION
    function res = PingPongSimWithMag(t, W)
        x = W(1);
```

```
    y = W(2);
    Vx = W(3);
    Vy = W(4);
    %spinsPerSecond = W(5);
    energyLosses = .9;
    %spin = spinsPerSecond*2*pi;         %Convert rads/s to rotations/s
    spin = W(5);                        %no need to convert now
    dXdt = Vx;
    dYdt = Vy;

    %angles
    angleV = atan(Vy/Vx);
    angleMagnus = angleV + (pi/2);

    %FORCES
    fMagnus = -pi*p^2*r^3*spin;                                             %Force of spin
    aMagnus = fMagnus/m;                                                    %Acceleration due
    to spin
    aMagX = aMagnus*cos(angleMagnus);
    aMagY = aMagnus*sin(angleMagnus);

    fDragX = -((C*A*p*(Vx^2+Vy^2))/2)*(Vx/((sqrt(Vx^2 + Vy^2))));        %Force of drag in x
    direction
    aDragX = fDragX/m;                                                      %X acceleration due
    to drag
    fDragY = -((C*A*p*(Vx^2+Vy^2))/2)*(Vy/((sqrt(Vx^2 + Vy^2))));        %Force of drag in y
    direction
    aDragY = fDragY/m;                                                      %Y acceleration due
    to drag

    %spin = spin*energyLosses;                                               %Affect of energy
    loss
    %outspin = spin/(2*pi)*energyLosses;

    dspindt = -10;
    %disp(['aMagX: ', num2str(aMagX,'%.3f'), ' aMagnus: ', num2str(aMagnus,'%.3f')])

    %energyLosses

    dVxdt = aDragX + aMagX;
    dVydt = aDragY + aMagY + g;
    res = [dXdt; dYdt; dVxdt; dVydt; dspindt];
end

function check_conservation(T, X, Y, Vx, Vy, rps)
    % Checks that the system conserves energy.
    % T: time vector
    % X: x position
    % Y: y position
    % Vx: x velocity
    % Vy: y velocity

    w = rps * 2 * pi;
    V1 = sqrt(Vx.^2 + Vy.^2);
```

```
        grav_potential = -m * g * Y;
        kinetic = .5 * m * V1.^2;
        rotational = -.5 * I * w.^2;

        size = 12;
        total = grav_potential + kinetic + rotational;
        plot(T, [grav_potential kinetic rotational total], 'linewidth', 2)
        legend('gravity', 'kinetic', 'rotational', 'total')
        xlabel('Time', 'fontsize', size)
        ylabel('Energy', 'fontsize', size)
        title('Energy Conservation: No Drag', 'fontsize', size+1)
        %ylim([0,4.5])
    end
end
```