Table of Contents


```
% Relativistic Electromagnetic Simulation of Two Circuits
% This code models two circuits with current, including relativistic effects
% for the propagation of electromagnetic fields and the resulting forces.

clear all;
close all;
clc;
```

Physical constants

```
c = 299792458;           % Speed of light in vacuum (m/s)
mu_0 = 4*pi*1e-7;        % Vacuum permeability (H/m)
epsilon_0 = 8.85e-12;    % Vacuum permittivity (F/m)
e = 1.602e-19;           % Elementary charge (C)
m_e = 9.11e-31;          % Electron mass (kg)
```

Simulation parameters

```
dt = 1e-12;              % Time step (s)
total_time = 1e-8/13;       % Total simulation time (s)
num_steps = ceil(total_time/dt);

% Circuit parameters
d = 0.1;                 % Distance between circuits (m)
circuit_radius = 0.01;   % Radius of the circular circuits (m)
circuit_segments = 100;  % Number of segments to discretize each circuit
current_magnitude = 1;   % Current magnitude (A)
wire_radius = 0.001;     % Wire radius (m)
n_electrons = 100;       % Number of electrons to visualize per circuit

% Initialize time and arrays
t = 0;
times = zeros(1, num_steps);
field_reached_middle = false;
switch_time = d/(2*c);   % Time when field reaches middle (s)
```

Create circuit geometries

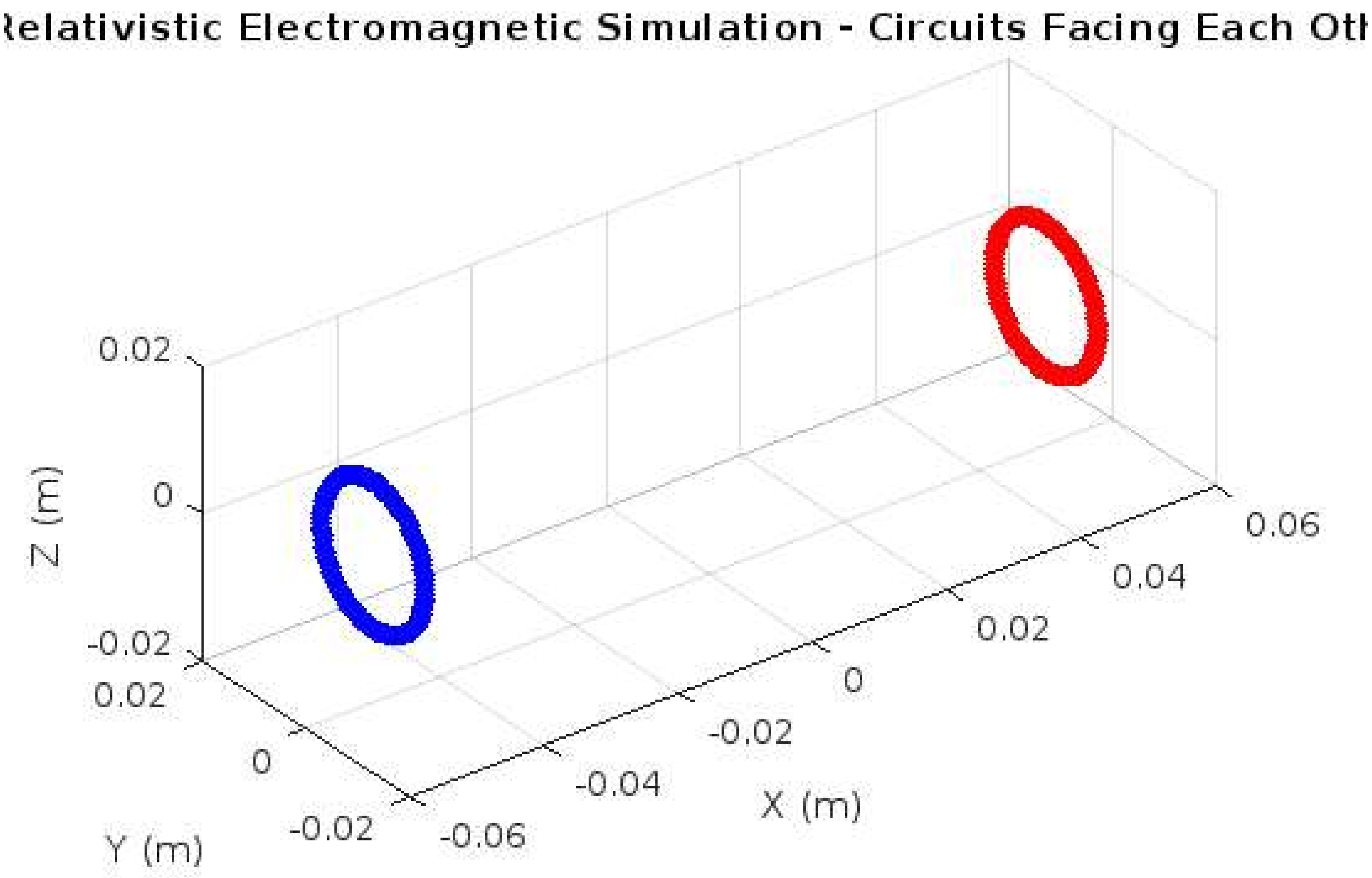
Circuit 1 (left circuit) - Now in YZ plane

```
theta = linspace(0, 2*pi, circuit_segments);
circuit1_x = -d/2 * ones(size(theta));  % All points at x = -d/2
circuit1_y = circuit_radius * cos(theta);
circuit1_z = circuit_radius * sin(theta);
circuit1_positions = [circuit1_x; circuit1_y; circuit1_z]';
circuit1_segments = [circuit1_positions, circshift(circuit1_positions, -1)];
circuit1_segments(end,:) = [];

% Circuit 2 (right circuit) - Now in YZ plane
circuit2_x = d/2 * ones(size(theta));   % All points at x = d/2
circuit2_y = circuit_radius * cos(theta);
circuit2_z = circuit_radius * sin(theta);
circuit2_positions = [circuit2_x; circuit2_y; circuit2_z]';
circuit2_segments = [circuit2_positions, circshift(circuit2_positions, -1)];
circuit2_segments(end,:) = [];
```

Setup visualization

```
figure('Position', [100, 100, 1200, 800]);
view(3);
grid on;
axis equal;
xlabel('X (m)');
ylabel('Y (m)');
zlabel('Z (m)');
title('Relativistic Electromagnetic Simulation - Circuits Facing Each Other');
xlim([-d/2-circuit_radius, d/2+circuit_radius]);
ylim([-2*circuit_radius, 2*circuit_radius]);
zlim([-2*circuit_radius, 2*circuit_radius]);
hold on;

% Plot circuits
circuit1_handle = plot3(circuit1_x, circuit1_y, circuit1_z, 'b', 'LineWidth',
2);
circuit2_handle = plot3(circuit2_x, circuit2_y, circuit2_z, 'r', 'LineWidth',
2);

% Initialize magnetic field visualization
[Y, Z] = meshgrid(linspace(-circuit_radius*1.5, circuit_radius*1.5, 20), ...
                  linspace(-circuit_radius*1.5, circuit_radius*1.5, 20));
X = zeros(size(Y));
Bx = zeros(size(Y));
By = zeros(size(Y));
Bz = zeros(size(Y));
quiver_handle = quiver3(X, Y, Z, Bx, By, Bz, 'g');

% Initialize electron visualization
electron1_positions = zeros(n_electrons, 3);
electron2_positions = zeros(n_electrons, 3);
for i = 1:n_electrons
    theta_e = 2*pi*i/n_electrons;
    electron1_positions(i,:) = [-d/2, circuit_radius * cos(theta_e),
circuit_radius * sin(theta_e)];
```

```
    electron2_positions(i,:) = [d/2, circuit_radius * cos(theta_e),
circuit_radius * sin(theta_e)];
end
electron1_handle = scatter3(electron1_positions(:,1),
electron1_positions(:,2), electron1_positions(:,3), 'b', 'filled');
electron2_handle = scatter3(electron2_positions(:,1),
electron2_positions(:,2), electron2_positions(:,3), 'r', 'filled');

% Field propagation visualization - Fixed initialization
% Initialize field_propagation as an empty array first
field_propagation = gobjects(1,2);

% Create the surface objects with valid coordinates
field_propagation(1) = surf([-d/2 -d/2; -d/2 -d/2], ...
                            [0 0; 0 0], ...
                            [0 0; 0 0], ...
                            'FaceColor', [0, 0.7, 0], ...   % Only RGB
                            'FaceAlpha', 0.2, ...            % Separate alpha
                            'EdgeColor', 'none', ...
                            'Visible', 'off');

field_propagation(2) = surf([d/2 d/2; d/2 d/2], ...
                            [0 0; 0 0], ...
                            [0 0; 0 0], ...
                            'FaceColor', [0, 0, 0.7], ...    % Only RGB
                            'FaceAlpha', 0.2, ...             % Separate alpha
                            'EdgeColor', 'none', ...
                            'Visible', 'off');

% Force visualization
force_arrow1 = quiver3(0, 0, 0, 0, 0, 0, 'b', 'LineWidth', 2, 'MaxHeadSize',
0.5);
force_arrow2 = quiver3(0, 0, 0, 0, 0, 0, 'r', 'LineWidth', 2, 'MaxHeadSize',
0.5);

% Text for information display
% Text for information display - positioned to avoid overlap
current_info = text(-d/2, 2*circuit_radius, 0, '', 'FontSize', 8,
'HorizontalAlignment', 'left');
time_info = text(0, 2*circuit_radius, 0, '', 'FontSize', 8,
'HorizontalAlignment', 'center');
field_info = text(d/2, 2*circuit_radius, 0, '', 'FontSize', 8,
'HorizontalAlignment', 'right');
force_info = text(-d/2, 3*circuit_radius, 0, '', 'FontSize', 8,
'HorizontalAlignment', 'left');
```



Main simulation loop

```
for step = 1:num_steps
    % Update time
    t = step * dt;
    times(step) = t;

    % Determine current direction in circuit 1
    if t > switch_time
        circuit1_current = -current_magnitude;
        if ~field_reached_middle
            field_reached_middle = true;
            disp(['Field reached middle at t = ', num2str(t), ' s']);
        end
    else
        circuit1_current = current_magnitude;
    end
    circuit2_current = current_magnitude;  % Circuit 2 always has same
direction

    % Calculate relativistic factor for electron drift
    v_drift1 = circuit1_current / (pi * wire_radius^2 * e * n_electrons);
    v_drift2 = circuit2_current / (pi * wire_radius^2 * e * n_electrons);
    gamma1 = 1/sqrt(1-(v_drift1/c)^2);
    gamma2 = 1/sqrt(1-(v_drift2/c)^2);

    % Update electron positions (with relativistic corrections)
    for i = 1:n_electrons
        theta_e = 2*pi*i/n_electrons + t*v_drift1/circuit_radius;
```

```
        electron1_positions(i,:) = [-d/2, circuit_radius * cos(theta_e),
circuit_radius * sin(theta_e)];

        theta_e = 2*pi*i/n_electrons + t*v_drift2/circuit_radius;
        electron2_positions(i,:) = [d/2, circuit_radius * cos(theta_e),
circuit_radius * sin(theta_e)];
    end

    % Calculate magnetic field at grid points
    for i = 1:size(Y, 1)
        for j = 1:size(Y, 2)
            % Now we're visualizing the field in the middle XY plane
            observation_point = [0, Y(i,j), Z(i,j)];

            % Initialize field components
            B = [0, 0, 0];

            % Calculate retarded time (time for field to propagate from
source to observation point)
            for k = 1:size(circuit1_segments, 1)
                segment = circuit1_segments(k,:);
                source_point = segment(1:3);
                dl = segment(4:6) - segment(1:3);

                r = observation_point - source_point;
                r_mag = norm(r);

                % Skip if observation point is too close to source to avoid
numerical issues
                if r_mag < 1e-6
                    continue;
                end

                % Calculate retarded time
                t_ret = t - r_mag/c;

                % Only contribute if field has had time to propagate
                if t_ret > 0
                    % Determine current at retarded time
                    if t_ret > switch_time
                        I_ret = -current_magnitude;
                    else
                        I_ret = current_magnitude;
                    end

                    % Biot-Savart law with relativistic corrections
                    dB = mu_0/(4*pi) * I_ret * cross(dl, r/r_mag) / r_mag^2;

                    % Relativistic correction factor
                    rel_factor = 1; % Simplified relativistic factor

                    B = B + dB * rel_factor;
                end
            end
```

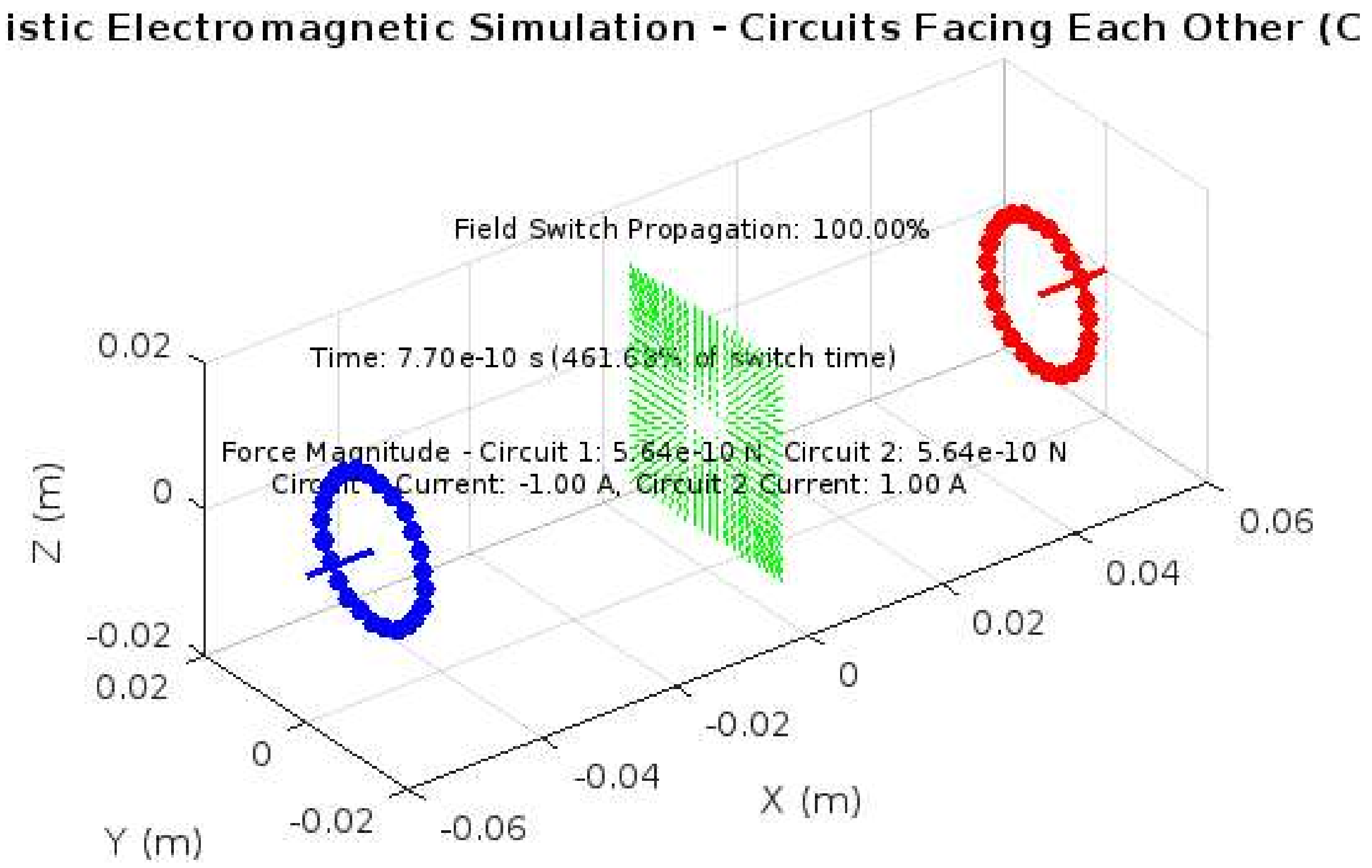
```
        % Circuit 2 contribution (similar calculations)
        for k = 1:size(circuit2_segments, 1)
            segment = circuit2_segments(k,:);
            source_point = segment(1:3);
            dl = segment(4:6) - segment(1:3);

            r = observation_point - source_point;
            r_mag = norm(r);

            if r_mag < 1e-6
                continue;
            end

            t_ret = t - r_mag/c;

            if t_ret > 0
                I_ret = current_magnitude; % Circuit 2 current is constant
                dB = mu_0/(4*pi) * I_ret * cross(dl, r/r_mag) / r_mag^2;
                rel_factor = 1; % Simplified relativistic factor
                B = B + dB * rel_factor;
            end
        end

        % Store field components
        Bx(i,j) = B(1);
        By(i,j) = B(2);
        Bz(i,j) = B(3);
    end
end

% Calculate force on each circuit (Lorentz force)
F_on_circuit1 = [0, 0, 0];
F_on_circuit2 = [0, 0, 0];

% Force on Circuit 1 due to Circuit 2
for k = 1:size(circuit1_segments, 1)
    segment = circuit1_segments(k,:);
    source_point = segment(1:3);
    dl = segment(4:6) - segment(1:3);

    % Calculate magnetic field at this point from Circuit 2
    B_at_point = [0, 0, 0];

    for m = 1:size(circuit2_segments, 1)
        segment2 = circuit2_segments(m,:);
        source_point2 = segment2(1:3);
        dl2 = segment2(4:6) - segment2(1:3);

        r = source_point - source_point2;
        r_mag = norm(r);

        if r_mag < 1e-6
            continue;
```

```
        end

        t_ret = t - r_mag/c;

        if t_ret > 0
            I_ret = current_magnitude;
            dB = mu_0/(4*pi) * I_ret * cross(dl2, r/r_mag) / r_mag^2;
            B_at_point = B_at_point + dB;
        end
    end

    % Calculate force element (I × dl × B)
    dF = circuit1_current * cross(dl, B_at_point);
    F_on_circuit1 = F_on_circuit1 + dF;
end

% Force on Circuit 2 due to Circuit 1 (similar calculation)
for k = 1:size(circuit2_segments, 1)
    segment = circuit2_segments(k,:);
    source_point = segment(1:3);
    dl = segment(4:6) - segment(1:3);

    B_at_point = [0, 0, 0];

    for m = 1:size(circuit1_segments, 1)
        segment1 = circuit1_segments(m,:);
        source_point1 = segment1(1:3);
        dl1 = segment1(4:6) - segment1(1:3);

        r = source_point - source_point1;
        r_mag = norm(r);

        if r_mag < 1e-6
            continue;
        end

        t_ret = t - r_mag/c;

        if t_ret > 0
            if t_ret > switch_time
                I_ret = -current_magnitude;
            else
                I_ret = current_magnitude;
            end

            dB = mu_0/(4*pi) * I_ret * cross(dl1, r/r_mag) / r_mag^2;
            B_at_point = B_at_point + dB;
        end
    end

    dF = circuit2_current * cross(dl, B_at_point);
    F_on_circuit2 = F_on_circuit2 + dF;
end
```

```
    % Visualize electromagnetic field propagation
    if t <= switch_time*1.5
        % Calculate propagation radius based on speed of light
        radius = c * t;

        % For circuit 1 - propagating along +X direction
        [Y_circle1, Z_circle1, X_circle1] = cylinder(radius, 50);
        X_circle1 = X_circle1 * 0.005 - d/2;  % Thin disk at x = -d/2

        % For circuit 2 - propagating along -X direction
        [Y_circle2, Z_circle2, X_circle2] = cylinder(radius, 50);
        X_circle2 = X_circle2 * 0.005 + d/2;  % Thin disk at x = d/2

        % Update the field propagation visualization
        set(field_propagation(1), 'XData', X_circle1, 'YData', Y_circle1,
'ZData', Z_circle1, 'Visible', 'on');
        set(field_propagation(2), 'XData', X_circle2, 'YData', Y_circle2,
'ZData', Z_circle2, 'Visible', 'on');
    end

    % Update visualization
    set(electron1_handle, 'XData', electron1_positions(:,1), 'YData',
electron1_positions(:,2), 'ZData', electron1_positions(:,3));
    set(electron2_handle, 'XData', electron2_positions(:,1), 'YData',
electron2_positions(:,2), 'ZData', electron2_positions(:,3));
    set(quiver_handle, 'XData', X, 'YData', Y, 'ZData', Z, 'UData', Bx,
'VData', By, 'WData', Bz);

    % Update force arrows
    F1_center = [-d/2, 0, 0];
    F2_center = [d/2, 0, 0];
    F1_scale = 0.05 * F_on_circuit1 / (norm(F_on_circuit1) + eps);
    F2_scale = 0.05 * F_on_circuit2 / (norm(F_on_circuit2) + eps);

    set(force_arrow1, 'XData', F1_center(1), 'YData', F1_center(2), 'ZData',
F1_center(3), ...
        'UData', F1_scale(1), 'VData', F1_scale(2), 'WData', F1_scale(3));
    set(force_arrow2, 'XData', F2_center(1), 'YData', F2_center(2), 'ZData',
F2_center(3), ...
        'UData', F2_scale(1), 'VData', F2_scale(2), 'WData', F2_scale(3));

    % Update text information
    set(current_info, 'String', sprintf('Circuit 1 Current: %.2f A, Circuit 2
Current: %.2f A', circuit1_current, circuit2_current));
    set(time_info, 'String', sprintf('Time: %.2e s (%.2f%% of switch time)',
t, t/switch_time*100));
    set(field_info, 'String', sprintf('Field Switch Propagation: %.2f%%',
min(t/switch_time*100, 100)));
    set(force_info, 'String', sprintf('Force Magnitude - Circuit 1: %.2e N,
Circuit 2: %.2e N', norm(F_on_circuit1), norm(F_on_circuit2)));

    drawnow;

    % Optional: Capture frame for video
```

```
    frames(step) = getframe(gcf);

    % Display progress
    if mod(step, ceil(num_steps/10)) == 0
        disp(['Simulation progress: ', num2str(step/num_steps*100), '%']);
    end
end

% Optional: Create video
firstFrame = frames(1).cdata;
targetSize = size(firstFrame);

video = VideoWriter('relativistic_circuits.avi', 'Motion JPEG AVI');
video.FrameRate = 30;
open(video);
for i = 1:length(frames)
    currentFrame = frames(i).cdata;
    if ~isequal(size(currentFrame), targetSize)
        resizedFrame = imresize(currentFrame, [targetSize(1), targetSize(2)]);
        tempFrame = frames(i);
        tempFrame.cdata = resizedFrame;
        writeVideo(video, tempFrame);
    else
        writeVideo(video, frames(i));
    end
end
close(video);


% Final display
title('Relativistic Electromagnetic Simulation - Circuits Facing Each Other
(Complete)');
disp('Simulation complete.');
```

```
Simulation progress: 10%
Simulation progress: 20%
Field reached middle at t = 1.67e-10 s
Simulation progress: 30%
Simulation progress: 40%
Simulation progress: 50%
Simulation progress: 60%
Simulation progress: 70%
Simulation progress: 80%
Simulation progress: 90%
Simulation progress: 100%
Simulation complete.
```



Published with MATLAB® R2024b