```m
function dxdt = quartercar_ode(t, x, mu, ms, ks, kt, cs, ct, amplitude, omega)
    % State variables
    zu = x(1);
    zs = x(2);
    zu_dot = x(3);
    zs_dot = x(4);

    % Road input
    vehicle_speed = 20;  % m/s
    x = vehicle_speed * t;
    wavelength = 5;

    zr = amplitude * sin(omega * x).* (x < wavelength);
    zr_dot = amplitude * omega * vehicle_speed *cos(omega * x).* (x < wavelength);


    % Equations of motion
    zu_ddot = (1/mu) * ( - (cs + ct)*zu_dot + cs*zs_dot - (ks + kt)*zu + ks*zs + ct*zr_dot + kt*zr );
    zs_ddot = (1/ms) * ( cs*(zu_dot - zs_dot) + ks*(zu - zs) );

    % State derivatives
    dxdt = [zu_dot; zs_dot; zu_ddot; zs_ddot];
end


%% Quarter Car Model - ode23 solution
clc; clear; close all;

% Parameters
mu = 300;      % unsprung mass (kg)
ms = 2500;     % sprung mass (kg)
ks = 250000;   % suspension stiffness (N/m)
kt = 800000;  % tire stiffness (N/m)
cs = 20000;     % suspension damping (N*s/m)
ct = 2000;        % tire damping (N*s/m), often zero

% Natural Frequency Calculation
fn_sprung = 1/(2*pi) * sqrt(ks/ms);
fprintf('Sprung Mass Natural Frequency: %.2f Hz\n', fn_sprung);

% Damping Ratio Calculation
zeta = cs / (2*sqrt(ks*ms));
fprintf('Damping Ratio: %.3f\n', zeta);


% Road input (Sinusoidal Bump)
vehicle_speed = 20;  % m/s
wavelength = 5;      % m
amplitude = 0.05;    % m
omega = 2*pi/wavelength;



% Time span
tspan = [0:0.001:10];  % 10 seconds simulation
dt = 0.001;

% Initial conditions [zu, zs, zu_dot, zs_dot]
x0 = [0 0 0 0];

```

```
% Define ODE function
f = @(t, x) quartercar_ode(t, x, mu, ms, ks, kt, cs, ct, amplitude, omega);

% Solve using ode23
[t, X] = ode23(f, tspan, x0);

% Extract states
zu = X(:,1);
zs = X(:,2);
zu_dot = X(:,3);
zs_dot = X(:,4);

% Plot results
figure;
plot(t, zu, 'b', t, zs, 'r', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Displacement (m)');
legend('Unsprung mass (z_u)', 'Sprung mass (z_s)');
title('Dynamic Response of Quarter Car Suspension (ODE23)');
grid on;



% Recalculate accelerations from differential equations
zs_ddot = zeros(size(t));
for i = 1:length(t)
    zr = amplitude * sin(omega * t(i));
    zr_dot = amplitude * omega * cos(omega * t(i));
    zu_ddot = (1/mu) * ( - (cs + ct)*zu_dot(i) + cs*zs_dot(i) - (ks + kt)*zu(i) + ks*zs(i) +
ct*zr_dot + kt*zr );
    zs_ddot(i) = (1/ms) * ( cs*(zu_dot(i) - zs_dot(i)) + ks*(zu(i) - zs(i)) );
end

% Compute RMS acceleration of sprung mass
rms_acc_sprung = sqrt(mean(zs_ddot.^2));

% Display result
fprintf('RMS acceleration of sprung mass = %.4f m/s^2\n', rms_acc_sprung);

% Plot responses
figure;
plot(t, zu, 'b', t, zs, 'r', 'LineWidth', 1.5);
xlabel('Time (s)'); ylabel('Displacement (m)');
legend('z_u (unsprung)', 'z_s (sprung)');
title('Quarter-Car Model Response (ODE23)');
grid on;

plot(t, zs_dot, 'r', 'LineWidth', 1.5);
xlabel('Time (s)'); ylabel('Velocity (m/s)');
title('Sprung Mass Velocity');
grid on;

plot(t, zs_ddot, 'r', 'LineWidth', 1.5);
xlabel('Time (s)'); ylabel('Acceleration (m/s^2)');
title(sprintf('Sprung Mass Acceleration (RMS = %.3f m/s^2)', rms_acc_sprung));
grid on;


% Suspension deflection
susp_deflection = zs - zu;
max_susp_deflection = max(abs(susp_deflection));
```

```
121
122 % Display results
123 fprintf('Maximum suspension deflection = %.4f m\n', max_susp_deflection);
124 figure;
125
126 plot(t, susp_deflection, 'm', 'LineWidth', 1.5);
127 xlabel('Time (s)'); ylabel('Deflection (m)');
128 title(sprintf('Suspension Deflection (Max = %.4f m)', max_susp_deflection));
129 grid on;
130
131
132 % --- Frequency range (rad/s) ---
133 omega = logspace(0, 3, 1000);  % 1 to 1000 rad/s
134 Ts = zeros(size(omega));       % Sprung mass transmissibility
135 Tu = zeros(size(omega));       % Unsprung mass transmissibility
136
137 % --- Loop through frequencies ---
138 for i = 1:length(omega)
139     w = omega(i);
140
141     % Complex dynamic stiffness matrix
142     A = [(-mu*w^2 + 1j*w*(cs+ct) + (ks+kt)),   (-1j*w*cs - ks);
143          (-1j*w*cs - ks),                      (-ms*w^2 + 1j*w*cs + ks)];
144
145     % Road excitation (unit amplitude)
146     B = [(1j*w*ct + kt); 0];
147
148     % Solve for Zu and Zs
149     Z = A\B;
150
151     Zu = Z(1);  % Unsprung mass displacement
152     Zs = Z(2);  % Sprung mass displacement
153
154     Tu(i) = abs(Zu);   % Unsprung transmissibility
155     Ts(i) = abs(Zs);   % Sprung transmissibility
156 end
157
158 % --- Plot Transmissibility ---
159 figure;
160 semilogx(omega/(2*pi), Ts, 'r-', 'LineWidth', 1.8); hold on;
161 semilogx(omega/(2*pi), Tu, 'b--', 'LineWidth', 1.8);
162 grid on;
163 xlabel('Frequency (Hz)');
164 ylabel('Transmissibility (|Z / Z_r|)');
165 title('Transmissibility of Sprung and Unsprung Mass');
166 legend('Sprung Mass (Z_s/Z_r)', 'Unsprung Mass (Z_u/Z_r)', 'Location', 'best');
167
168
169 %-------------------------------------------------------------------------------
170 %%for random road response
171 % 3.3 Random Road Profile (ISO 8608 Class B)
172 % PSD parameters for Class B road
173 Gq_n0 = 64e-6;   % Road roughness coefficient for Grade B road (m^3)
174 n0 = 0.1;        % Reference spatial frequency (1/m)
175 w = 2;           % Waviness
176 v = vehicle_speed;  % Vehicle speed (m/s)
177
178 % Convert to time-domain parameters
179 alpha = 2 * pi * n0;   % Filter parameter (rad/m)
180 sigma_w = sqrt(2 * pi * v * Gq_n0 * n0^w);  % White noise std deviation
181
```

```
% Time vector
t = 0:dt:10;

% Generate white noise
w_t = sigma_w * randn(size(t));

% Filtered white noise to get random road displacement q(t)
zr = zeros(size(t));
for i = 2:length(t)
    zr(i) = zr(i-1) + dt * (w_t(i) - alpha * v * zr(i-1));
end

% Road velocity (derivative)
zr_dot = [0 diff(zr)/dt];

% Plot random road profile
figure;
plot(t, zr, 'y', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Road displacement z_r (m)');
title('Random Road Surface (ISO 8608 - Grade B)');
grid on;
```