```
% ME 303 - Zhao Pan
% Programmers: Shaan B, Zubair H, Mirza M, Dharmik R, Milind K
% Date: 10th March, 2025

clc; % clear command window

% Table 1 Parameters
m = 1400;   % Vehicle mass (kg)
a = 1.14;   % Distance from centre of mass to the front axle (m)
b = 1.33;   % Distance from centre of mass to the rear axle (m)
Iz = 2420;  % Yaw inertia (kg·m^2)
dt = 0.01;  % Δt (grid spacing)
t = 0:dt:5; % Simulate for 5 seconds
x0 = [0; 0]; % Initial conditions
delta = 0.1; % Step steering input

% Use 100 km/h to determine oversteer or understeer
u = 100 / 3.6; % Convert km/h to m/s

% Define different tire stiffness cases (N/rad)
tire_cases = {
    {'Default', 25000, 21000},
    {'Increased Front Stiffness', 30000, 21000},
    {'Increased Rear Stiffness', 25000, 26000},
    {'Reduced Front Stiffness', 20000, 21000},
    {'Reduced Rear Stiffness', 25000, 16000}
};

% Pre-allocate the storage for trajectory data
X_data = zeros(length(tire_cases), length(t)); % X positions
Y_data = zeros(length(tire_cases), length(t)); % Y positions

% Loop through each tire stiffness scenario and run RK4
for idx = 1:length(tire_cases)
    % Extract stiffness values for each case
    case_name = tire_cases{idx}{1};
    Cf = tire_cases{idx}{2};
    Cr = tire_cases{idx}{3};

    % System of ODEs
    A = [- (Cf + Cr) / (m * u), - (a * Cf - b * Cr) / (m * u) - u;
       - (a * Cf - b * Cr) / (Iz * u), - (a^2 * Cf + b^2 * Cr) / (Iz * u)];
    % Define A as a 2x2 matrix using the given parameters

    B = [Cf / m; a * Cf / Iz]; % Define B as a 2x1 matrix

    % Pre-allocate storage
    x_rk4 = zeros(2, length(t)); % Define vector for state variables
    x_rk4(:,1) = x0; % Set first point to initial conditions
    psi = zeros(1, length(t)); % Yaw angle
    X = zeros(1, length(t)); % X position
```

```
    Y = zeros(1, length(t)); % Y position

    % RK4 solver (same as before)
    for i = 1:length(t)-1
      k1 = A * x_rk4(:,i) + B * delta;
      k2 = A * (x_rk4(:,i) + 0.5 * dt * k1) + B * delta;
      k3 = A * (x_rk4(:,i) + 0.5 * dt * k2) + B * delta;
      k4 = A * (x_rk4(:,i) + dt * k3) + B * delta;
      x_rk4(:,i+1) = x_rk4(:,i) + (dt/6) * (k1 + 2*k2 + 2*k3 + k4);

      % Integrate yaw rate to get heading angle
      psi(i+1) = psi(i) + dt * x_rk4(2,i);

      % Integrate velocity equations to get X and Y positions
      X(i+1) = X(i) + dt * (u * cos(psi(i)) - (x_rk4(1,i) ...
            + a * x_rk4(2,i)) * sin(psi(i)));
      Y(i+1) = Y(i) + dt * ((x_rk4(1,i) + a * x_rk4(2,i)) * cos(psi(i)) ...
            + u * sin(psi(i)));
      % 2 lines above refined and corrected with the use of LLM as prev
    end

    % Store trajectory data
    X_data(idx,:) = X;
    Y_data(idx,:) = Y;
end

% Plot X-Y Vehicle Trajectory for each tire stiffness case
figure; % Open new figure
hold on; % Keep multiple plots
grid on; % Enable grid for clarity

for idx = 1:length(tire_cases) % Run loop to plot each stiffness case
    plot(X_data(idx,:), Y_data(idx,:), 'LineWidth', 1.5, ...
         'DisplayName', tire_cases{idx}{1});
end

legend('Location', 'Best', 'Interpreter', 'Latex'); % Add a legend
xlabel('X Position (m)', 'Interpreter', 'Latex'); % X-axis label
ylabel('Y Position (m)', 'Interpreter', 'Latex'); % Y-axis label
axis equal; % Keep x and y axes equal when zooming in and out
hold off;
```