```m
%% prob_4.m
%
% this script is used to answer problem 4
%
% - written by: Dimitri Lezcano

%% Set-up parameters
x_0 = zeros(5,1); % initial state
Q = diag([5, 5, 0.01, 0.1, 0.1]);
R = diag([0.5, 0.1]);
tf = 5; % final time
ud = [0; 0];

%% Compute A and B matrices
x_0d = compute_xd(0);
A_d = compute_A(x_0d);
B_d = compute_B(x_0); % grab B since it is a constant


%% Optimize the lqr problem to get the K matrix
[K, S, e] = lqr(A_d, B_d, Q, R, 0);

%% Determine the trajectory
[t, x] = ode45(@(t, x) dynamics(t, x, K), [0,tf], x_0);

x = x'; % reshape to 5 x N matrix
xd = compute_xd(t); % desired trajectory

%% Get the control
u = zeros(length(ud), length(t));
for i = 1:length(t)
    t_i = t(i);
    x_i = x(:,i);

    u(:,i) = control_law(t_i, x_i, K);

end

%% Plotting
fig = figure(1);
% plot the trajectories
subplot(2,2,1);
plot(t, x(1:2,:)); hold on;
plot(t, xd(1:2,:)); hold off;
legend('x_1', 'x_2', 'x_{d,1}', 'x_{d,2}', 'location', 'best');
xlabel('t'); ylabel('p_i');
title('trajectories vs. time')

% plot the 2-d Trajectories
subplot(2,2,2);
plot(x(1,:), x(2,:), 'DisplayName', 'executed'); hold on;
plot(xd(1,:), xd(2,:), 'DisplayName', 'desired'); hold off;
legend('location', 'best');
xlabel('p_x'); ylabel('p_y');
title('2-D trajectories');

% plot the control
subplot(2,2,[3 4]);
plot(t, u);
xlabel('t'); ylabel('u');
title('u(t)');

%% Saving the figure
fig_save = 'prob_4.jpg';
saveas(fig, fig_save);
fprintf('Saved figure: %s\n\n', fig_save);

%% Functions
% function for computing A matrix
```

```
function A = compute_A(x)
    v = x(4); % velocity
    th = x(3); % theta
    delta = x(5); % delta

    A = zeros(5);
    % Set values
    A(1, 3) = - v * sin(th);
    A(2, 3) = v * cos(th);

    A(1,4) = cos(th);
    A(2,4) = sin(th);
    A(3,4) = tan(delta);

    A(3, 5) = v*sec(delta)^2;

end

% function for computing B matrix
function B = compute_B(x)
    B = zeros(5,2);

    % set values
    B(4,1) = 1;
    B(5,2) = 1;

end

% compute the desired trajectory
function xd = compute_xd(t)
    t = reshape(t, 1, []);
    xd = [t; 2*t; atan(2)*ones(size(t)); sqrt(5)*ones(size(t)); zeros(size(t))];

end

% the system dynamics
function dx = dynamics(t, x, K)
    v = x(4); % velocity
    th = x(3); % theta
    delta = x(5); % delta

    u = control_law(t, x, K);
    dx = [v*cos(th); v*sin(th); v*tan(delta); u];


end

% the control law
function u = control_law(t, x, K)
    xd = compute_xd(t);
    u = -K * (x - xd);

end

```