```
%% Problem 2
clc; close all;

rng(0,"v4");

% SIM variables
dt=0.1;
dtMeas=0.1;
tNextMeas = dtMeas;
meas_available = 0;
tMax = 16;  % Max time, s
step = floor(dtMeas/dt);
makeplot = 1;
rho0 =0.0034; krho=22000; g=32.2; r1=1000; r2=100;

% EKF Parameters
options = odeset('RelTol',1e-3,'AbsTol',1e-6);

var_init_uncertainty = [500; 20000; 250000];
var_meas_noise = 4;
var_process_noise = [0; 0; 0];

R = diag(var_meas_noise); % mxm Measurement noise matrix
P = diag(var_init_uncertainty); % nxn Covariance matrix
Q = diag(var_process_noise); % nxn
m = length(R);
n = length(P);

x0_true = [100100; -5650; 3000];
x0_est = [100000; -6000; 2600];

% For storing process and t=0 values
saveVars = {"T", "X_true", "X_est", "Z_true", "Z_est", "P_est", "P_plot", "K_plot",↙
"step","P_lim","K_lim", "L_lim","info", "makeplot"};

T = 0:dt:tMax;
t_length = length(T);
X_true = nan(n,t_length); % True state vectors (n x steps)
X_est = nan(n,t_length); % Estimate state vectors (n x steps)
Z_true = nan(m,t_length); % True measurement vectors (m x steps)
Z_est = nan(m,t_length); % Estimate measurement vectors (m x steps)
P_est = nan(n,n,t_length); % Estimate variance vectors (n x n steps)
P_plot= nan(n,t_length);
K_plot = nan(n,t_length);

xest  = x0_est;
xtrue = x0_true;
for i = 1:t_length
    t = T(i);
```

```
    if t>=tNextMeas
        tNextMeas = t+dtMeas;
        meas_available=1;
    else
        meas_available=0;
     end

    if meas_available == 1
        X_true(:,i) = xtrue;
        X_est(:,i)  = xest;
        Z_true(:,i) = h(X_true(:,i),r1,r2) + sqrt(R).*randn(m,1);
        Z_est(:,i) =  h(X_est(:,i),r1,r2);
        P_est(:,:,i) = P;
        P_plot(:,i) = diag(P_est(:,:,i));

        % Gain Matrix
        H = jacobian_h(xest,r1,r2);
        K = P_est(:,:,i)*H.'/(H*P_est(:,:,i)*H.' + R);

        % States updated with measurement information
        X_est(:,i) = X_est(:,i) + K*(Z_true(:,i) - Z_est(:,i));

        % Covariance matrix updated with measurement information
        P_est(:,:,i) = (eye(n) - K*H)*P_est(:,:,i);
        K_plot(:,i) = K;
        P_plot(:,i) = diag(P_est(:,:,i));
    else
        X_true(:,i) = xtrue;
        X_est(:,i)  = xest;
        Z_est(:,i) = nan(m,1);
        P_est(:,:,i) = P;
        K_plot(:,i) = nan(n,1);
        P_plot(:,i) = diag(P_est(:,:,i));
    end

    t = t+dt;

    % STATE propagation
    [~,xtrue]  = ode45(@f,[0 dt],X_true(:,i),options,rho0,krho,g);
    xtrue = xtrue(end,:)';

    PHI = expm(jacobian_f(X_est(:,i),rho0,krho)*dt);
    [~,x] = ode45(@f,[0 dt],X_est(:,i),options,rho0,krho,g);
    xest = x(end,:)';
    P = PHI*P_est(:,:,i)*PHI' + Q;

end
clearvars('-except',saveVars{:})

%--------------------------------------------------------------------------
```

```
function H = jacobian_h(x,r1,r2)

    H=[(x(1)-r2)/sqrt(r1^2+(x(1)-r2)^2),0,0];
end

%---------------------------------------------------------------------------
function F = jacobian_f(x,rho0,krho)
    F=zeros(3,3);
    F(1,:)=[0,1,0];
    F(2,:)=[-rho0/krho*exp(-x(1)/krho)*(x(2))^2/(2*x(3)), rho0*exp(-x(1)/krho)*x(2)/x(3),...
            -rho0*exp(-x(1)/krho)*(x(2))^2/(2*(x(3))^2)];
    F(3,:)=[0,0,0];
end

%---------------------------------------------------------------------------
function z=h(x,r1,r2)
     z=sqrt(r1^2+(x(1,:)-r2).^2);
end

%---------------------------------------------------------------------------
function dx = f(~,x,rho0,krho,g)
    dx = zeros(3,1);
    dx(1) = x(2);
    dx(2) = (rho0*exp(-x(1)/krho).*(x(2))^2)/(2*x(3)) - g;
    dx(3) = 0;
end

if makeplot
    %% Error plots
    figure()
    hold on
    plot(T,X_est(1,:)-X_true(1,:), 'r-');
    plot(T, sqrt(P_plot(1,:)),'r--',T,- sqrt(P_plot(1,:)),'r--')
    title('Altitude Estimation')
    xlabel('Time (s)')
    ylabel('Altitude Error')
    ylim([-10 10])
    hold off

    figure()
    hold on
    plot(T,X_est(3,:)-X_true(3,:), 'r');
    plot(T, sqrt(P_plot(3,:)),'r--',T,- sqrt(P_plot(3,:)),'r--')
    title('Drag Estimation')
    xlabel('Time (s)')
    ylabel('Drag Error')
    hold off
end
```