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

% rng(0,"v4");
set(groot,'defaultTextInterpreter','latex')

% SIM variables
tMax = 60;  % Max time, s
dt=0.1;
dtMeas=0.1;
tNextMeas = dtMeas;
meas_available = 0;
tlength = floor(tMax/dt) + 1; % length of vectors
N = [20, 0];
E = [0, 20];
% EKF Parameters

var_init_uncertainty = [0; 0; 0; 0];
var_meas_noise = [1; 1];
var_process_noise = [0; 0; 4; 4];

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

m = length(R);
n = length(P);

x0_true = [0; 0; 50; 50];

F = [1 0 dt 0;
     0 1 0 dt;
     0 0 1  0;
     0 0 0 1];

% For storing process and t=0 values
saveVars = {"T", "X_true", "X_est", "Z_true", "Z_est", "P_est", "P_plot", "step","↙
P_lim","K_lim", "L_lim","info", "makeplot"};
xNames={'$\hat{N}, m$', '$\hat{E}, m$', '$\hat{\dot{N}}, m/s$', '$\hat{\dot{E}},↙
m/s$'};

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);
```

```
xest = x0_true;
xtrue = x0_true;
ztrue = h(xtrue,N,E) + sqrt(R)*randn(m,1);
zest = h(xest,N,E);

H = jacobian_h(x0_true,N,E);
A = eye(n) - F;
[P_lim,K_lim,L_lim,info] = dare(A,H',Q,R,[],[]);

for i = 1:tlength
    % Store current time info
    t = T(i);
    X_true(:,i) = xtrue;
    X_est(:,i)  = xest;
    Z_true(:,i) = ztrue;
    Z_est(:,i) =  zest;
    P_est(:,:,i) = P;
    P_plot(:,i) = diag(P);

    % Propagate foward
    t = t+dt;

    % Sim update
    xtrue = F*xtrue +  sqrt(Q)*randn(n,1);

    % Propagation of previous state estimate to current time
    L = eye(n);

    P = F*P*F' + L*Q*L';
    xest = F*xest;
    ztrue = h(xtrue,N,E) + sqrt(R)*randn(m,1);
    zest =  h(xest,N,E);

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

    if meas_available == 1

        % Gain Matrix
        H = jacobian_h(xest,N,E);
        M = eye(m);
        K = (P*H.')/(H*P*H.' + M*R*M');

        % States updated with measurement information
        xest = xest + K*(ztrue - zest);
```

```
        % Covariance matrix updated with measurement information
        P = (eye(n) - K*H)*P;
    end
end

    figure()
tiledlayout()
for i = 1:2
    nexttile
    hold on
    plot(T,X_est(i,:)-X_true(i,:), 'r-');
    plot(T, sqrt(P_plot(i,:)),'b-',T,- sqrt(P_plot(i,:)),'b-')
    xlabel('Time (s)')
    title(xNames(i))
end
    figure()
tiledlayout()
for i = 3:4
    nexttile
    hold on
    plot(T,X_est(i,:)-X_true(i,:), 'r-');
    plot(T, sqrt(P_plot(i,:)),'b-',T,- sqrt(P_plot(i,:)),'b-')
    xlabel('Time (s)')
    title(xNames(i))
end
clearvars('-except',saveVars{:})

%---------------------------------------------------------------------------
function H = jacobian_h(x,N,E)

    H=[(x(1)-N(1))/sqrt((x(1)-N(1))^2 + (x(2)-E(1))^2), (x(2)-E(1))/sqrt((x(1)-N(1))^2 ↙
+ (x(2)-E(1))^2), 0, 0;
       (x(1)-N(2))/sqrt((x(1)-N(2))^2 + (x(2)-E(2))^2), (x(2)-E(2))/sqrt((x(1)-N(2))^2 ↙
+ (x(2)-E(2))^2), 0, 0];
end

%---------------------------------------------------------------------------
function z=h(x,N,E)
    z = [sqrt((x(1)-N(1))^2 + (x(2)-E(1))^2);
         sqrt((x(1)-N(2))^2 + (x(2)-E(2))^2)];
end
```