Assignment 10 Development of a tracking filter of a moving object when measurements and motion models are in different coordinate system

Group 6:

- **Andrei Shumeiko**
- **Ayush Gupta**
- **Olga Klyagina**
- **Simona Nitti**


```
clc
clear
close all
```

Task 1: Generate a true trajectory X

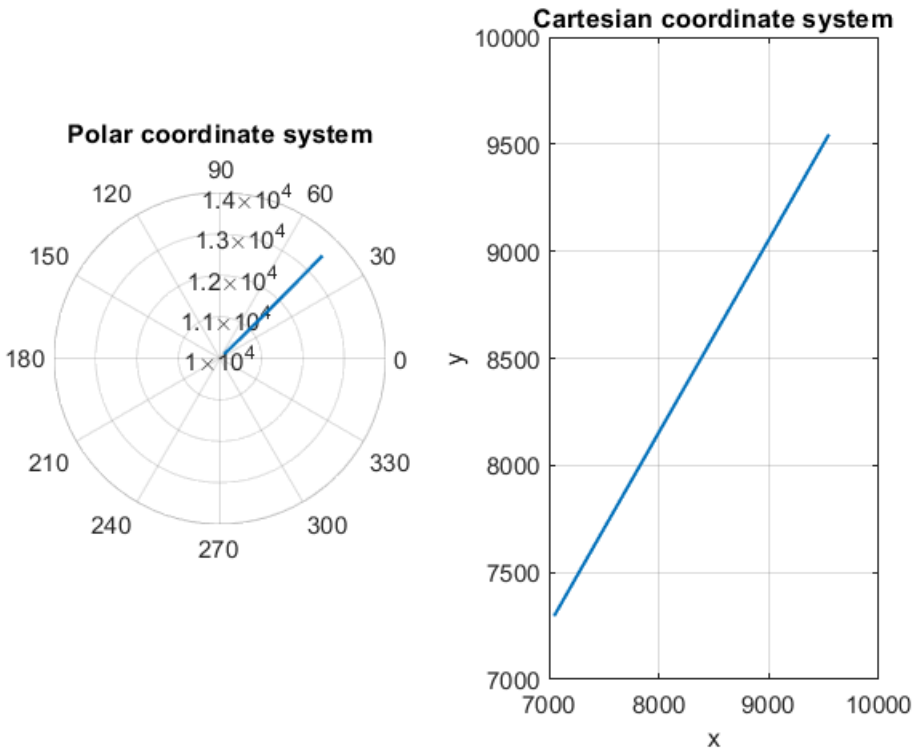
```
n = 26; % Size of trajectory
T = 2;  % Interval between measurements

%Initial components of velocity
V = zeros(2,n);
V(1,1)= -50; % V_x
V(2,1)= -45; % V_y

% Initial coordinates
x = zeros(2,n);
x(1,1)= 13500/sqrt(2); % x
x(2,1)= 13500/sqrt(2); % y

for i = 2:n
    x(1,i) = x(1,i-1)+V(1,i-1)*T;
    x(2,i) = x(2,i-1)+V(2,i-1)*T;
    V(1,i) = V(1,i-1);
    V(2,i) = V(2,i-1);
end
```

Task 2: Generate also true values of range and azimuth

```
D = sqrt(x(1,:).^2 + x(2,:).^2);  % true range
beta = atan(x(1,:)./x(2,:));      % true azimuth

figure
sgtitle('Fig.1: True trajectory')
subplot(1,2,1)
```

```
polarplot(beta,D,'linewidth',1.2)
title('Polar coordinate system')
rlim([10e3 14e3])

subplot(1,2,2)
plot(x(1,:),x(2,:),'linewidth',1.2)
title('Cartesian coordinate system')
xlabel('x')
ylabel('y')
grid on
```

Fig.1: True trajectory



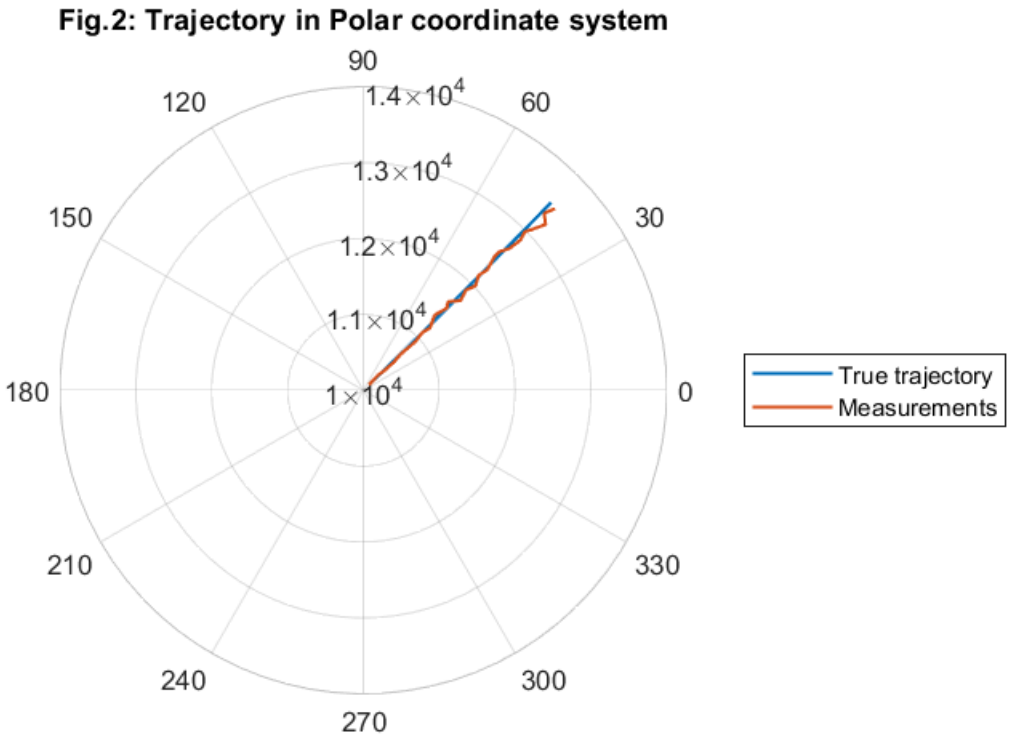
Task 3: Generate measurements of range and azimuth

```
var_D = 20^2;
var_beta = 0.02^2;

eta = randn(2,n);
eta(1,:) = eta(1,:)*sqrt(var_D);    % eta_D
eta(2,:) = eta(2,:)*sqrt(var_beta); % eta_beta

D_meas = D + eta(1,:);
beta_meas = beta + eta(2,:);
```

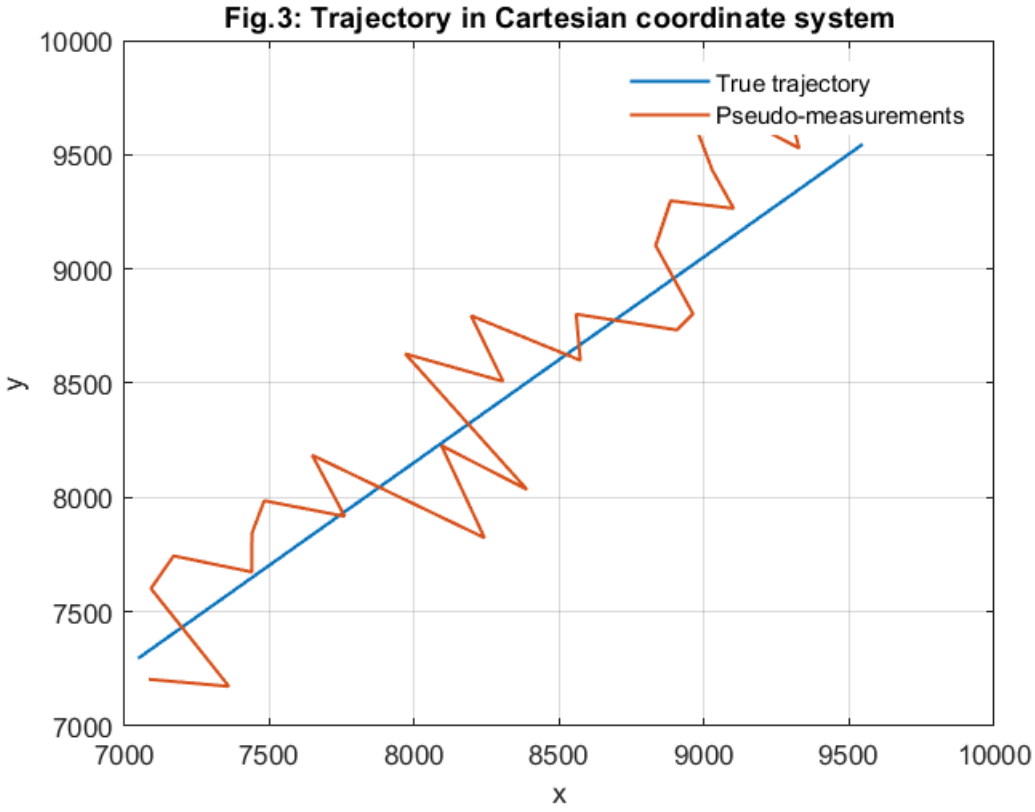
Task 4: Get pseudo-measurements of coordinates x and y

```
x_meas = D_meas.*sin(beta_meas);
y_meas = D_meas.*cos(beta_meas);
```

```
figure
polarplot(beta,D,'linewidth',1.2)
hold on
polarplot(beta_meas,D_meas,'linewidth',1.2)
title('Fig.2: Trajectory in Polar coordinate system')
rlim([10e3 14e3])
legend('True trajectory','Measurements')
```




```
figure
plot(x(1,:),x(2,:),'linewidth',1.2)
hold on
plot(x_meas,y_meas,'linewidth',1.2)
title('Fig.3: Trajectory in Cartesian coordinate system')
xlabel('x')
ylabel('y')
grid on
legend('True trajectory','Pseudo-measurements')
```



Task 5: Create the measurement vector z from pseudo-measurements of coordinates x and y

```
z = [x_meas;y_meas];
```

Task 6: Initial conditions for Kalman filter algorithm

```
X = zeros(4,n);
X(:,1) = [40000 -20 40000 -20]'; % Initial filtered estimate of state vector
P = 1e10*eye(4);                 % Initial filtration error covariance matrix

polar_f(1,1) = sqrt(X(1,1)^2+X(3,1)^2); % Initial filtered estimate of D
polar_f(2,1) = atan(X(1,1)/X(3,1));     % Initial filtered estimate of beta
```

Task 7: Create the transition matrix and observation matrix

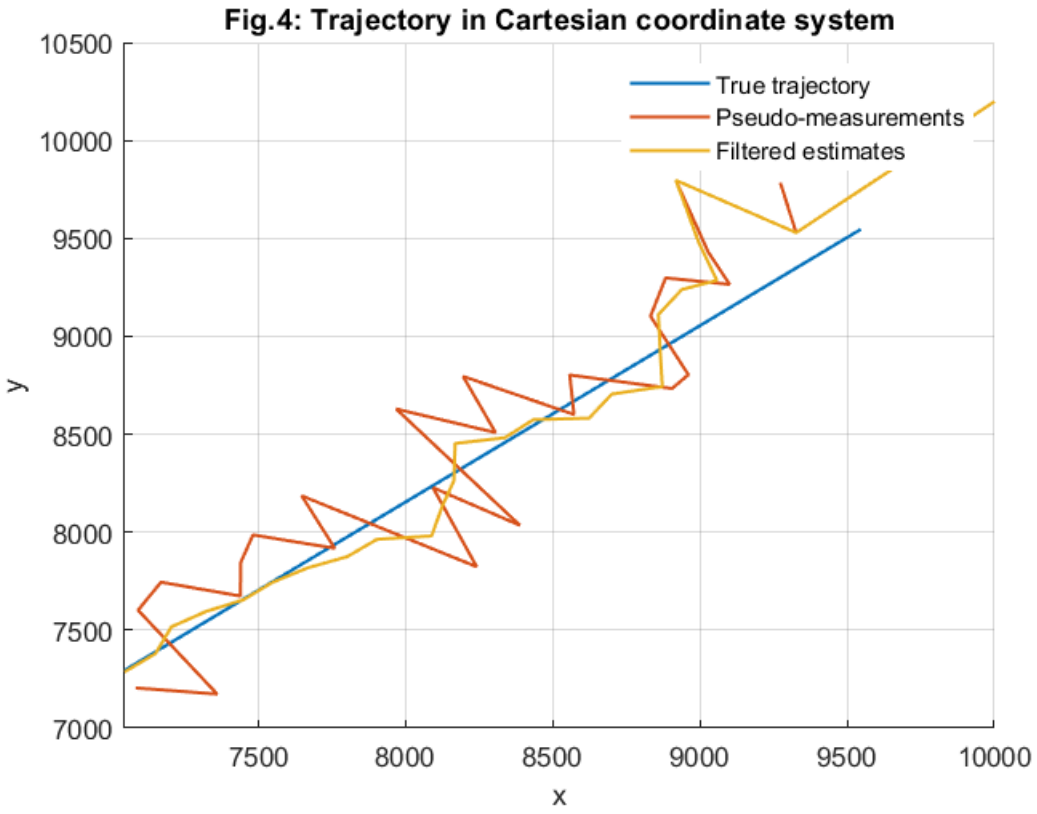
```
% Transition matrix
phi = eye(4)+ T*diag(ones(3,1),1);
phi(2,3)=0;

% Observation matrix
H = eye(4);
H(2:2:4,:) = [];
```

Task 8: Create the measurement error covariance matrix R

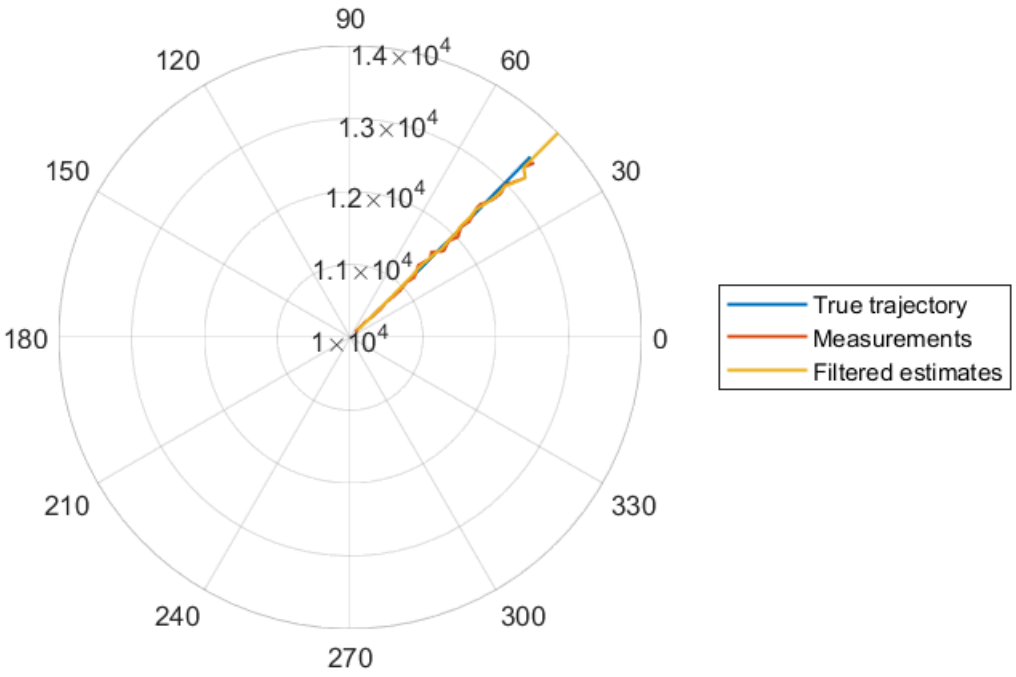
```
j = 1;
for i = 1:n
    R(1,j:j+1) = [var_D*(sin(beta_meas(i)))^2+(D_meas(i))^2*(cos(beta_meas(i)))^2*var_beta,...
        sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta)];
    R(2,j:j+1) = [sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta),...
        var_D*(cos(beta_meas(i)))^2+(D_meas(i))^2*(sin(beta_meas(i)))^2*var_beta ];
    j = j + 2;
end
```

Task 9: Develop Kalman filter algorithm

```
K = zeros(4,2*n);
k = zeros(1,n);
X_pred = zeros(4,n);
polar_e = zeros(2,n);
j = 1;
for i = 2:n

    % Prediction
    X_pred(:,i) = phi*X(:,i-1);
    P_pred = phi*P*phi';
    polar_e(1,i) = sqrt(X_pred(1,i)^2+X_pred(3,i)^2);   % extrapolated estimate of D
    polar_e(2,i) = atan(X_pred(1,i)/X_pred(3,i));  % extrapolated estimate of beta

    % Filtration
    K(:,j:j+1) = P_pred*H'*inv(H*P_pred*H'+R(:,j:j+1));
    k(i) = K(1,j); % save K(1,1)
    X(:,i) = X_pred(:,i) + K(:,j:j+1)*(z(:,i)-H*X_pred(:,i));
    P = (eye(4)-K(:,j:j+1)*H)*P_pred;

    % Change of coordinates
    polar_f(1,i) = sqrt(X(1,i)^2+X(3,i)^2); % filtered estimate of D
    polar_f(2,i) = atan(X(1,i)/X(3,i));     % filtered estimate of beta

    j = j + 2;
end

figure
hold on
plot(x(1,:),x(2,:),'linewidth',1.2)
plot(x_meas,y_meas,'linewidth',1.2)
plot(X(1,:),X(3,:),'linewidth',1.2)
xlabel('x')
ylabel('y')
legend('True trajectory','Pseudo-measurements','Filtered estimates')
title('Fig.4: Trajectory in Cartesian coordinate system')
grid on
xlim([-inf 1e4])
```




```
figure
polarplot(beta,D,'linewidth',1.2)
hold on
polarplot(beta_meas,D_meas,'linewidth',1.2)
polarplot(polar_f(2,:),polar_f(1,:),'linewidth',1.2)
title('Fig.5: Trajectory an object that moves uniformly : Trajectory in Polar coordinate system
rlim([10e3 14e3])
legend('True trajectory','Measurements','Filtered estimates')
```



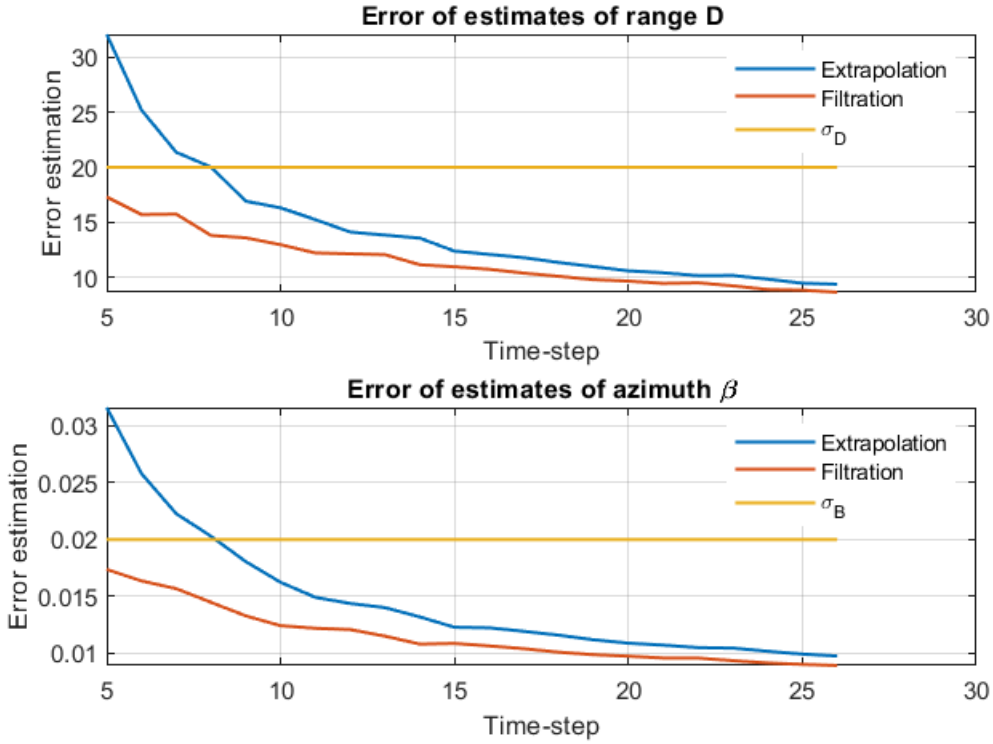
Task 10: Run Kalman filter algorithm over = 500 runs. Calculate true errors of estimation.

```
M = 500;

Final_Error_estr = zeros(2,n);
error_estr = zeros(2,n);
Final_Error_filt = zeros(2,n);
error_filt = zeros(2,n);
for run = 1:M
    eta = randn(2,n);
    eta(1,:) = eta(1,:)*sqrt(var_D);    % eta_D
    eta(2,:) = eta(2,:)*sqrt(var_beta); % eta_beta

    D_meas = D + eta(1,:);
    beta_meas = beta + eta(2,:);

    j = 1;
    for i = 1:n
        R(1,j:j+1) = [var_D*(sin(beta_meas(i)))^2+(D_meas(i))^2*(cos(beta_meas(i)))^2*var_beta,
            sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta)];
        R(2,j:j+1) = [sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta),...
            var_D*(cos(beta_meas(i)))^2+(D_meas(i))^2*(sin(beta_meas(i)))^2*var_beta ];
        j = j + 2;
    end
    z = [D_meas.*sin(beta_meas);D_meas.*cos(beta_meas)]; % Pseuso-measurements
```

```
    X = zeros(4,n);
    X(:,1) = [40000 -20 40000 -20]'; % Initial filtered estimate of state vector
    P = 1e10*eye(4);                 % Initial filtration error covariance matrix
    polar_f(1,:) = sqrt(X(1,1)^2+X(3,1)^2)*ones(1,n); % Initial filtered estimate of D
    polar_f(2,:) = atan(X(1,1)/X(3,1))*ones(1,n);     % Initial filtered estimate of beta

    X_pred = zeros(4,n);
    polar_e = zeros(2,n);
    K = zeros(4,2*n);
    j = 1;
    for i = 2:n
        % Prediction
        X_pred(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi';
        polar_e(1,i) = sqrt(X_pred(1,i)^2+X_pred(3,i)^2); % extrapolated estimate of D
        polar_e(2,i) = atan(X_pred(1,i)/X_pred(3,i));     % extrapolated estimate of beta

        % Filtration
        K(:,j:j+1) = P_pred*H'*inv(H*P_pred*H'+R(:,j:j+1));
        X(:,i) = X_pred(:,i) + K(:,j:j+1)*(z(:,i)-H*X_pred(:,i));
        P = (eye(4)-K(:,j:j+1)*H)*P_pred;
        polar_f(1,i) = sqrt(X(1,i)^2+X(3,i)^2); % filtered estimate of D
        polar_f(2,i) = atan(X(1,i)/X(3,i));     % filtered estimate of beta

        j = j + 2;
    end

    for i = 3:n
        %Errors of extrapolation estimates of range D
        error_estr(1,i) = (D(i)-polar_e(1,i))^2;
        Final_Error_estr(1,i)= Final_Error_estr(1,i)+error_estr(1,i);
        %Errors of filtration estimates of range D
        error_filt(1,i) = (D(i)-polar_f(1,i))^2;
        Final_Error_filt(1,i)= Final_Error_filt(1,i)+error_filt(1,i);

        %Errors of extrapolation estimates of azimuth beta
        error_estr(2,i) = (beta(i)-polar_e(2,i))^2;
        Final_Error_estr(2,i)= Final_Error_estr(2,i)+error_estr(2,i);
        %Errors of filtration estimates of azimuth beta
        error_filt(2,i) = (beta(i)-polar_f(2,i))^2;
        Final_Error_filt(2,i)= Final_Error_filt(2,i)+error_filt(2,i);
    end
end
Final_Error_estr = sqrt(Final_Error_estr/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
subplot(2,1,1)
sgtitle('Fig.6: Errors of estimation')
plot(Final_Error_estr(1,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(1,:),'linewidth',1.2)
plot(sqrt(var_D)*ones(1,n),'linewidth',1.2)
grid on
```

```
title('Error of estimates of range D')
legend('Extrapolation','Filtration','\sigma_D')
ylabel('Error estimation')
xlabel('Time-step')
xlim([5 30])

subplot(2,1,2)
plot(Final_Error_estr(2,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(2,:),'linewidth',1.2)
plot(sqrt(var_beta)*ones(1,n),'linewidth',1.2)
title('Error of estimates of azimuth \beta')
legend('Extrapolation','Filtration','\sigma_B')
ylabel('Error estimation')
xlabel('Time-step')
grid on
xlim([5 30])
```

Fig.6: Errors of estimation

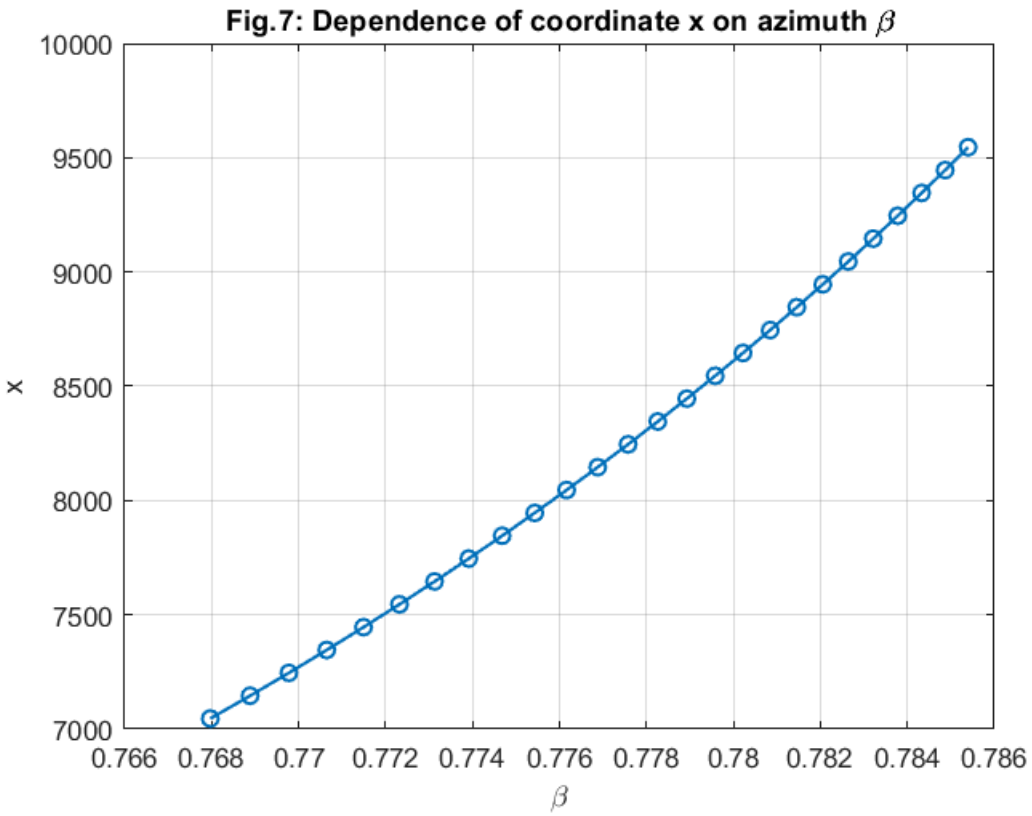


The errors decreses as the time step increases.

Task 11: Analyze dependence of coordinate x on azimuth beta

```
figure
plot(beta,x(1,:),'-o','linewidth',1.2)
ylabel('x')
xlabel('\beta')
grid on
```

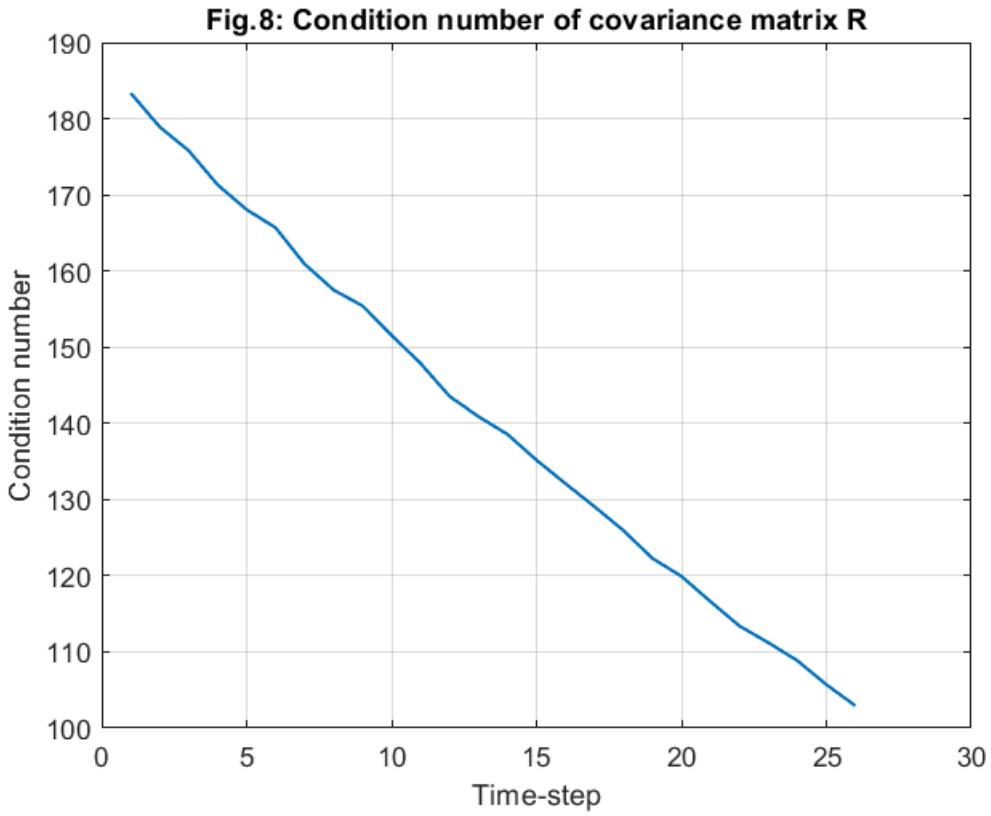
```
title('Fig.7: Dependence of coordinate x on azimuth \beta')
```



It is close to a linear trend and hence the linearization errors are insignificant.

Task 12: Calculate condition number of covariance matrix R

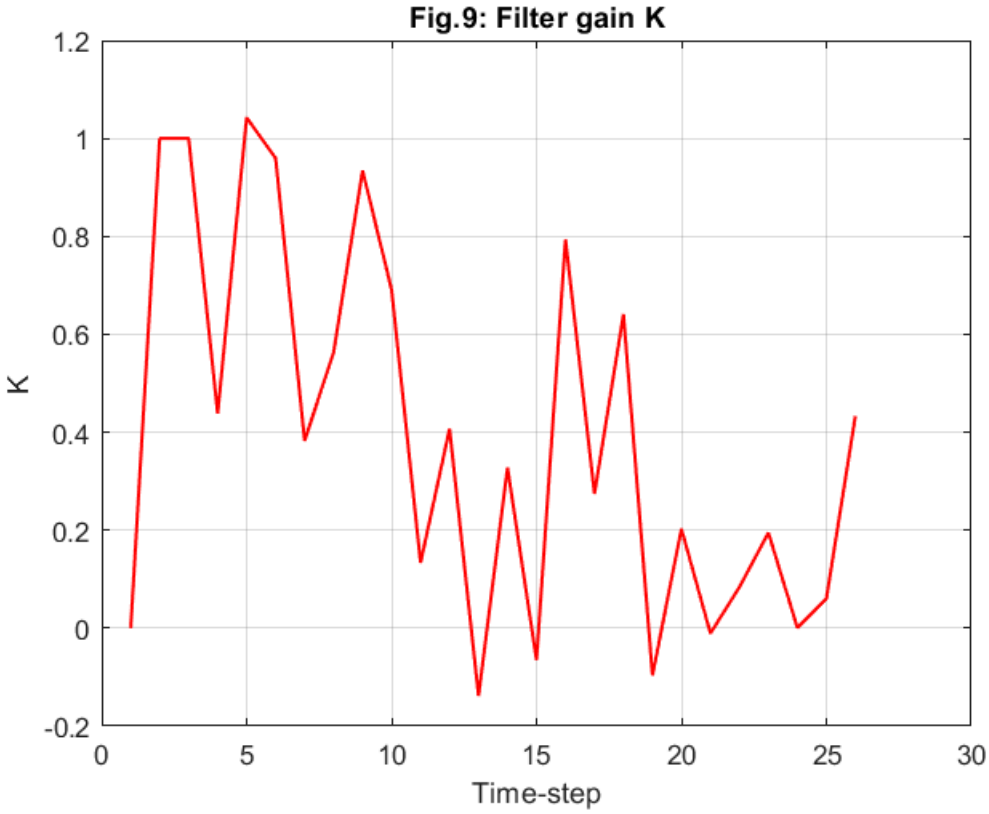
```
lambda_1 = var_D*ones(1,n);
lambda_2 = (D_meas.^2)*var_beta;
condition_number = zeros(1,n);

for i = 1:n
    if lambda_1(i)>lambda_2(i)
        condition_number(i) = lambda_1(i)/lambda_2(i);
    else
        condition_number(i) = lambda_2(i)/lambda_1(i);
    end
end
figure
plot(condition_number,'linewidth',1.2)
title('Fig.8: Condition number of covariance matrix R')
ylabel('Condition number')
xlabel('Time-step')
grid on
```



It can be observed that the condition number decreses over time from 181 to 102 and is not closer to 1, so it is a ill-conditioned matrix.

Task 13: Analyze filter gain K

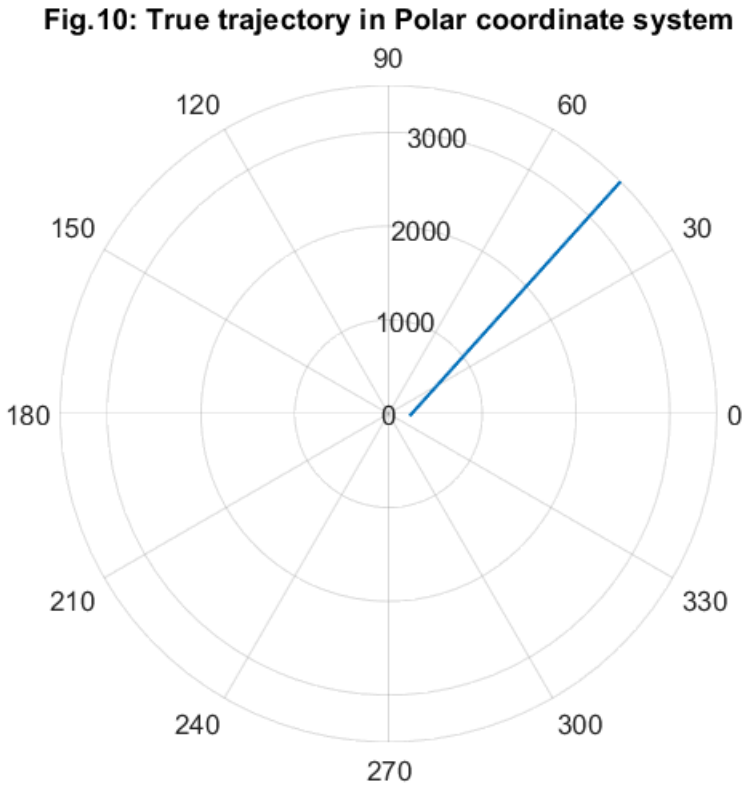
```
figure
plot(k,'r','linewidth',1.2)
title('Fig.9: Filter gain K')
ylabel('K')
xlabel('Time-step')
grid on
```



Task 14: The object starts it motion at a quite close distance from an observer. Run filter again over = 500 runs.

```
clc
clear

n = 26; % Size of trajectory
T = 2;  % Interval between measurements
var_D = 20^2;
var_beta = 0.02^2;

%Initial components of velocity
V = zeros(2,n);
V(1,1)= -50; % V_x
V(2,1)= -45; % V_y

% Initial coordinates
x = zeros(2,n);
x(1,1)= 3500/sqrt(2); % x
x(2,1)= 3500/sqrt(2); % y

for i = 2:n
    x(1,i) = x(1,i-1)+V(1,i-1)*T;
    x(2,i) = x(2,i-1)+V(2,i-1)*T;
    V(1,i) = V(1,i-1);
    V(2,i) = V(2,i-1);
end
```

```
D = sqrt(x(1,:).^2 + x(2,:).^2);
beta = atan(x(1,:)./x(2,:));

figure
polarplot(beta,D,'linewidth',1.2)
title('Fig.10: True trajectory in Polar coordinate system')
```




```
% Transition matrix
phi = eye(4)+ T*diag(ones(3,1),1);
phi(2,3)=0;

% Observation matrix
H = eye(4);
H(2:2:4,:) = [];

% Run the filter M times
M = 500;

Final_Error_estr = zeros(2,n);
error_estr = zeros(2,n);
Final_Error_filt = zeros(2,n);
error_filt = zeros(2,n);
for run = 1:M
    eta = randn(2,n);
    eta(1,:) = eta(1,:)*sqrt(var_D);    % eta_D
    eta(2,:) = eta(2,:)*sqrt(var_beta); % eta_beta
```

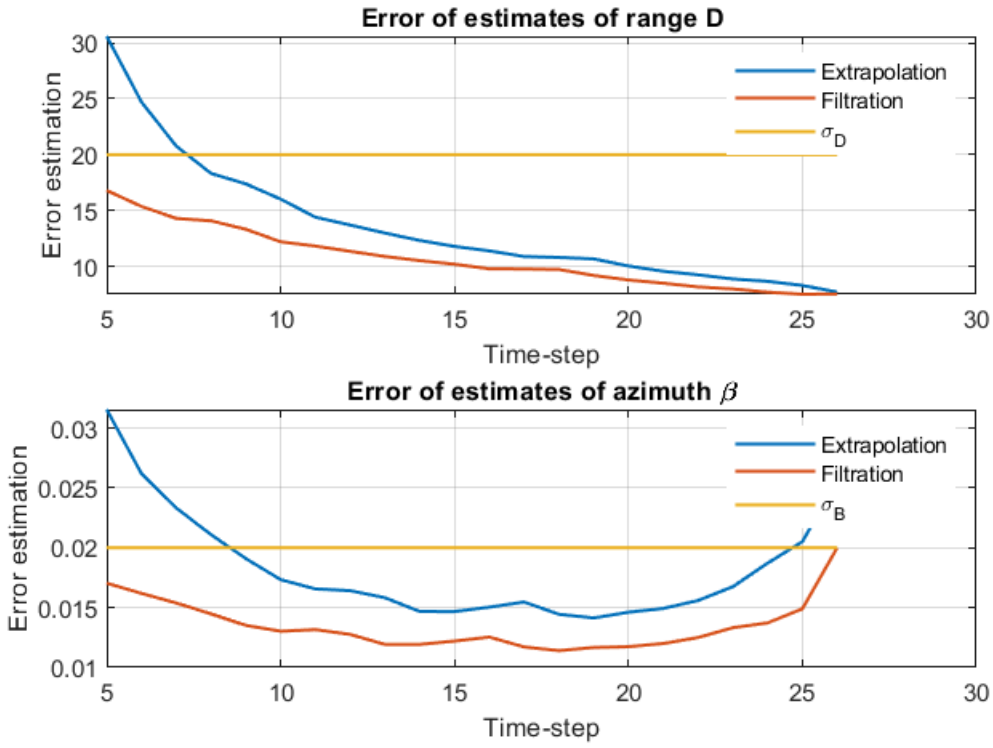
```
D_meas = D + eta(1,:);
beta_meas = beta + eta(2,:);

j = 1;
for i = 1:n
    R(1,j:j+1) = [var_D*(sin(beta_meas(i)))^2+(D_meas(i))^2*(cos(beta_meas(i)))^2*var_beta,
        sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta)];
    R(2,j:j+1) = [sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta),...
        var_D*(cos(beta_meas(i)))^2+(D_meas(i))^2*(sin(beta_meas(i)))^2*var_beta ];
    j = j + 2;
end
z = [D_meas.*sin(beta_meas);D_meas.*cos(beta_meas)]; % Pseuso-measurements

X = zeros(4,n);
X(:,1) = [40000 -20 40000 -20]'; % Initial filtered estimate of state vector
P = 1e10*eye(4);                 % Initial filtration error covariance matrix
polar_f(1,:) = sqrt(X(1,1)^2+X(3,1)^2)*ones(1,n); % Initial filtered estimate of D
polar_f(2,:) = atan(X(1,1)/X(3,1))*ones(1,n);     % Initial filtered estimate of beta

X_pred = zeros(4,n);
polar_e = zeros(2,n);
K = zeros(4,2*n);
j = 1;
for i = 2:n
    % Prediction
    X_pred(:,i) = phi*X(:,i-1);
    P_pred = phi*P*phi';
    polar_e(1,i) = sqrt(X_pred(1,i)^2+X_pred(3,i)^2); % extrapolated estimate of D
    polar_e(2,i) = atan(X_pred(1,i)/X_pred(3,i));     % extrapolated estimate of beta

    % Filtration
    K(:,j:j+1) = P_pred*H'*inv(H*P_pred*H'+R(:,j:j+1));
    X(:,i) = X_pred(:,i) + K(:,j:j+1)*(z(:,i)-H*X_pred(:,i));
    P = (eye(4)-K(:,j:j+1)*H)*P_pred;
    polar_f(1,i) = sqrt(X(1,i)^2+X(3,i)^2); % filtered estimate of D
    polar_f(2,i) = atan(X(1,i)/X(3,i));     % filtered estimate of beta

    j = j + 2;
end

for i = 3:n
    %Errors of extrapolation estimates of range D
    error_estr(1,i) = (D(i)-polar_e(1,i))^2;
    Final_Error_estr(1,i)= Final_Error_estr(1,i)+error_estr(1,i);
    %Errors of filtration estimates of range D
    error_filt(1,i) = (D(i)-polar_f(1,i))^2;
    Final_Error_filt(1,i)= Final_Error_filt(1,i)+error_filt(1,i);

    %Errors of extrapolation estimates of azimuth beta
    error_estr(2,i) = (beta(i)-polar_e(2,i))^2;
    Final_Error_estr(2,i)= Final_Error_estr(2,i)+error_estr(2,i);
    %Errors of filtration estimates of azimuth beta
    error_filt(2,i) = (beta(i)-polar_f(2,i))^2;
    Final_Error_filt(2,i)= Final_Error_filt(2,i)+error_filt(2,i);
```

```
    end
end
Final_Error_estr = sqrt(Final_Error_estr/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));
```

Task 15: Calculate true errors of estimation

```
figure
subplot(2,1,1)
sgtitle('Fig.11: Errors of estimation')
plot(Final_Error_estr(1,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(1,:),'linewidth',1.2)
plot(sqrt(var_D)*ones(1,n),'linewidth',1.2)
grid on
title('Error of estimates of range D')
legend('Extrapolation','Filtration','\sigma_D')
ylabel('Error estimation')
xlabel('Time-step')
xlim([5 30])

subplot(2,1,2)
plot(Final_Error_estr(2,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(2,:),'linewidth',1.2)
plot(sqrt(var_beta)*ones(1,n),'linewidth',1.2)
title('Error of estimates of azimuth \beta')
legend('Extrapolation','Filtration','\sigma_B')
ylabel('Error estimation')
xlabel('Time-step')
grid on
xlim([5 30])
```

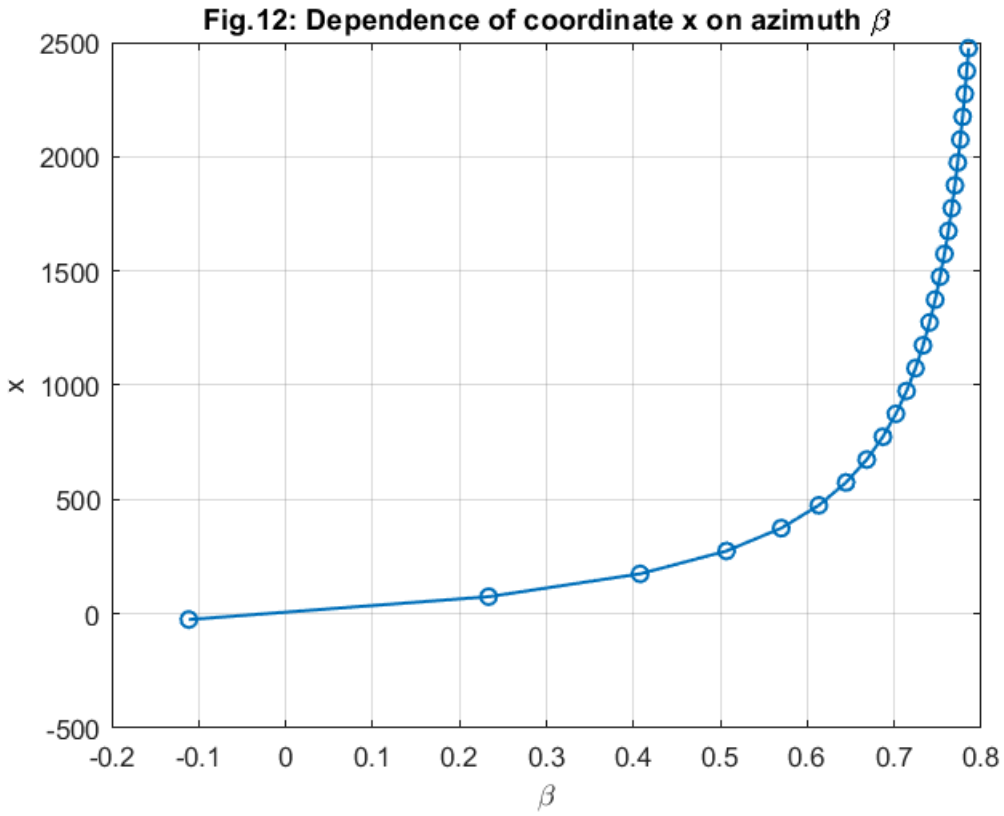
Fig.11: Errors of estimation



The estimation errors for range (D) decreases with the increase in the time-steps, but the azimuth (beta) first decreses and then increases as the time-steps are incresed.

Task 16: Analyze dependence of coordinate on azimuth beta

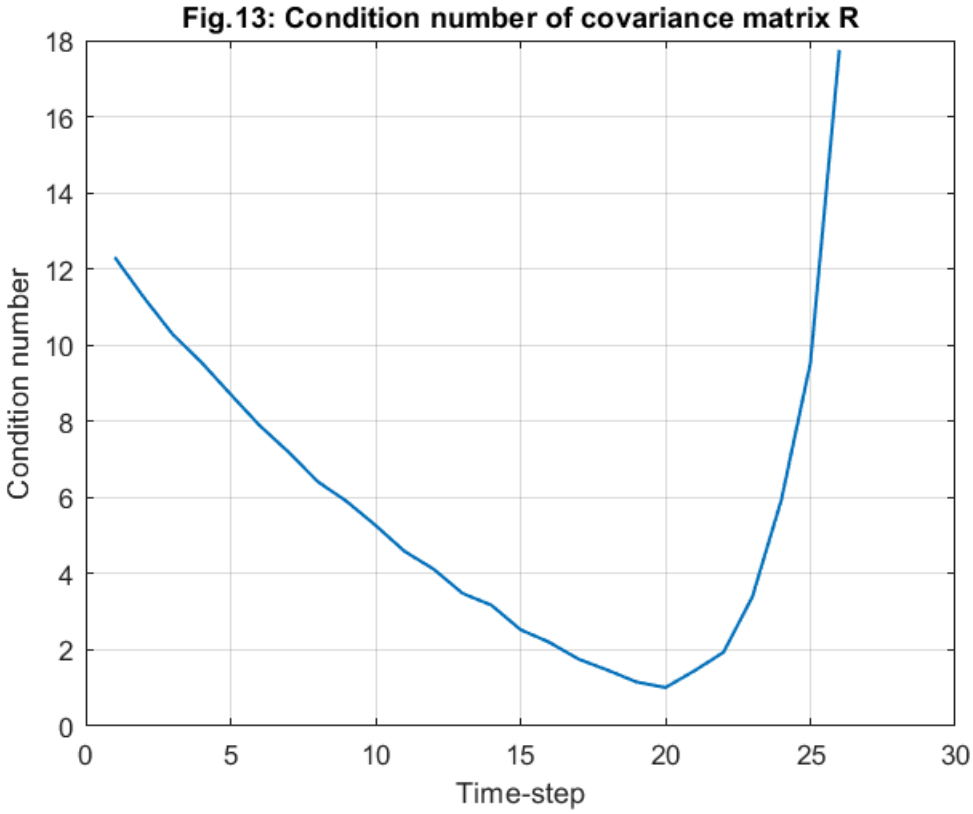
```
figure
plot(beta,x(1,:),'-o','linewidth',1.2)
ylabel('x')
xlabel('\beta')
grid on
title('Fig.12: Dependence of coordinate x on azimuth \beta')
```



The dependence of coodinate x on azimuth beta is non-linear. This means that the linearization errors are significan.

Task 17: Calculate condition number of covariance matrix R

```
lambda_1 = var_D*ones(1,n);
lambda_2 = (D_meas.^2)*var_beta;
condition_number = zeros(1,n);

for i = 1:n
    if lambda_1(i)>lambda_2(i)
        condition_number(i) = lambda_1(i)/lambda_2(i);
    else
        condition_number(i) = lambda_2(i)/lambda_1(i);
    end
end

figure
plot(condition_number,'linewidth',1.2)
title('Fig.13: Condition number of covariance matrix R')
ylabel('Condition number')
xlabel('Time-step')
grid on
```



It can be observed that from the given set of conditions the condition number decreses over time from 12 to ~1 and then increases to 23 with increase in time step. Since the condition number reaches ~1 it can be a well-conditioned matrix.

Task 18: Make conclusions how linearization errors affect tracking accuracy and how important for tracking accuracy is starting position of a moving object.

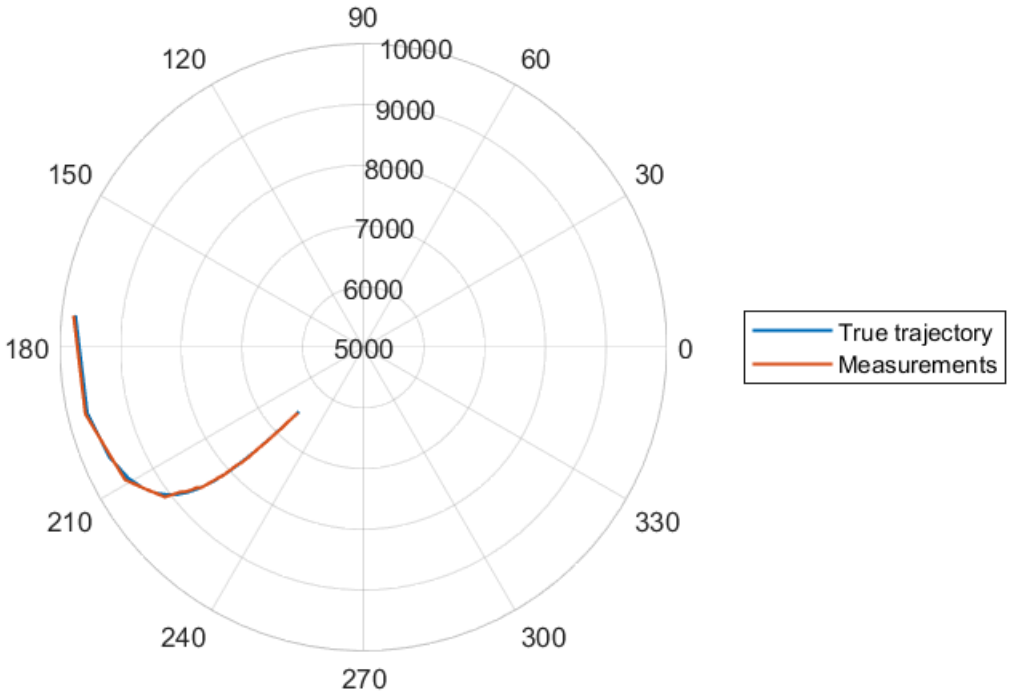
Comparing the accuracy of tracking results for an object that starts its motion at a great distance from an observer and another one which starts it at a quite close distance, it can be observed that the best result is achieved in the former case. This means that the starting position of a moving object has an important role in the relation of coordinate x on azimuth, which is linear in the first scenario and nonlinear in the second. Thus, linearity is a crucial aspect for the well-working of the filter.

Task 19: The object starts it motion at a quite close distance from an observer. Run filter again over = 500 runs.

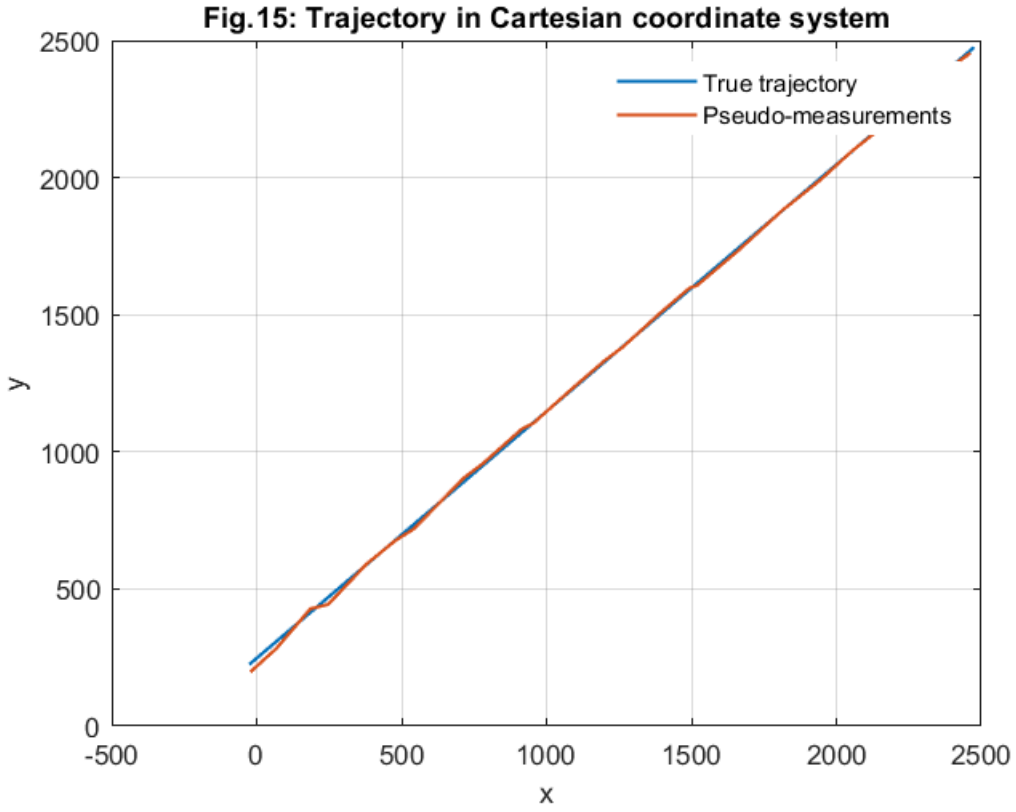
```
clc
clear

n = 26; % Size of trajectory
T = 2;  % Interval between measurements
var_D = 50^2;
var_beta = 0.0015^2;
```

```
%Initial components of velocity
V = zeros(2,n);
V(1,1)= -50; % V_x
V(2,1)= -45; % V_y

% Initial coordinates
x = zeros(2,n);
x(1,1)= 3500/sqrt(2); % x
x(2,1)= 3500/sqrt(2); % y

for i = 2:n
    x(1,i) = x(1,i-1)+V(1,i-1)*T;
    x(2,i) = x(2,i-1)+V(2,i-1)*T;
    V(1,i) = V(1,i-1);
    V(2,i) = V(2,i-1);
end

D = sqrt(x(1,:).^2 + x(2,:).^2);
beta = atan(x(1,:)./x(2,:));

% Transition matrix
phi = eye(4)+ T*diag(ones(3,1),1);
phi(2,3)=0;

% Observation matrix
H = eye(4);
H(2:2:4,:) = [];

eta = randn(2,n);
eta(1,:) = eta(1,:)*sqrt(var_D);    % eta_D
eta(2,:) = eta(2,:)*sqrt(var_beta); % eta_beta

D_meas = D + eta(1,:);
beta_meas = beta + eta(2,:);

z = [D_meas.*sin(beta_meas);D_meas.*cos(beta_meas)]; % Pseuso-measurements

figure
polarplot(beta,D,'linewidth',1.2)
hold on
polarplot(beta_meas,D_meas,'linewidth',1.2)
title('Fig.14: Trajectory in Polar coordinate system')
rlim([5e3 10e3])
legend('True trajectory','Measurements')
```

Fig.14: Trajectory in Polar coordinate system




```
figure
plot(x(1,:),x(2,:),'linewidth',1.2)
hold on
plot(z(1,:),z(2,:),'linewidth',1.2)
title('Fig.15: Trajectory in Cartesian coordinate system')
legend('True trajectory','Pseudo-measurements')
xlabel('x')
ylabel('y')
grid on
```

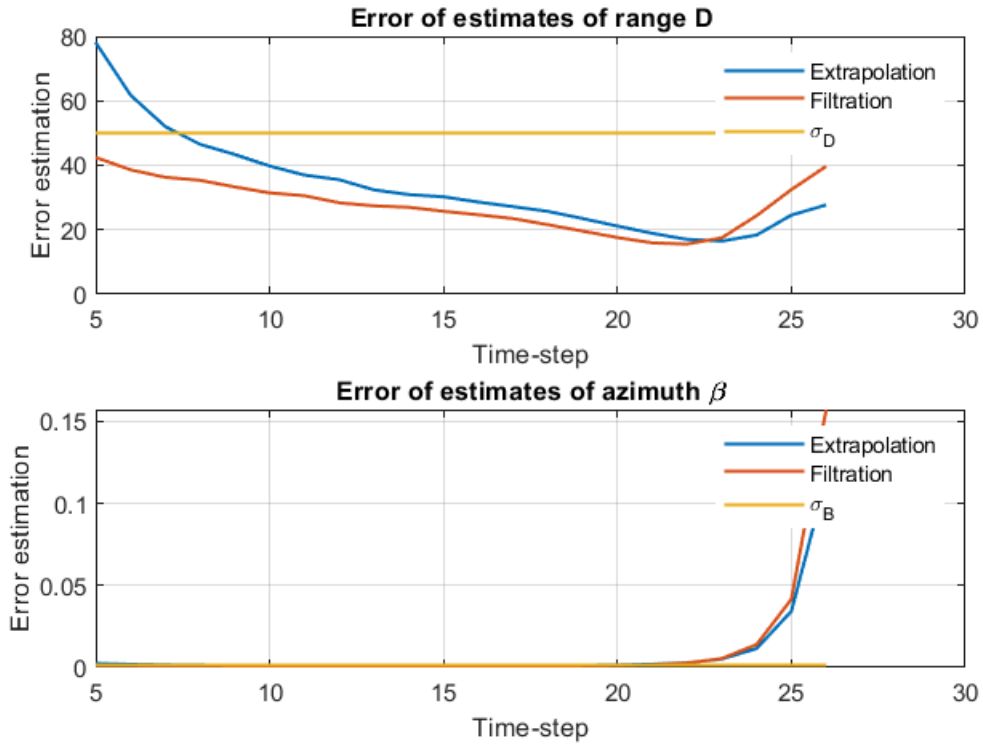



```
% Run the filter M times
M = 500;

Final_Error_estr = zeros(2,n);
error_estr = zeros(2,n);
Final_Error_filt = zeros(2,n);
error_filt = zeros(2,n);
for run = 1:M
    eta = randn(2,n);
    eta(1,:) = eta(1,:)*sqrt(var_D);    % eta_D
    eta(2,:) = eta(2,:)*sqrt(var_beta); % eta_beta

    D_meas = D + eta(1,:);
    beta_meas = beta + eta(2,:);

    j = 1;
    for i = 1:n
        R(1,j:j+1) = [var_D*(sin(beta_meas(i)))^2+(D_meas(i))^2*(cos(beta_meas(i)))^2*var_beta,
            sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta)];
        R(2,j:j+1) = [sin(beta_meas(i))*cos(beta_meas(i))*(var_D-(D_meas(i))^2*var_beta),...
            var_D*(cos(beta_meas(i)))^2+(D_meas(i))^2*(sin(beta_meas(i)))^2*var_beta ];
        j = j + 2;
    end
    z = [D_meas.*sin(beta_meas);D_meas.*cos(beta_meas)]; % Pseuso-measurements

    X = zeros(4,n);
    X(:,1) = [40000 -20 40000 -20]'; % Initial filtered estimate of state vector
    P = 1e10*eye(4);                 % Initial filtration error covariance matrix
```

```
    polar_f(1,:) = sqrt(X(1,1)^2+X(3,1)^2)*ones(1,n); % Initial filtered estimate of D
    polar_f(2,:) = atan(X(1,1)/X(3,1))*ones(1,n);     % Initial filtered estimate of beta

    X_pred = zeros(4,n);
    polar_e = zeros(2,n);
    K = zeros(4,2*n);
    j = 1;
    for i = 2:n
        % Prediction
        X_pred(:,i) = phi*X(:,i-1);
        P_pred = phi*P*phi';
        polar_e(1,i) = sqrt(X_pred(1,i)^2+X_pred(3,i)^2); % extrapolated estimate of D
        polar_e(2,i) = atan(X_pred(1,i)/X_pred(3,i));     % extrapolated estimate of beta

        % Filtration
        K(:,j:j+1) = P_pred*H'*inv(H*P_pred*H'+R(:,j:j+1));
        X(:,i) = X_pred(:,i) + K(:,j:j+1)*(z(:,i)-H*X_pred(:,i));
        P = (eye(4)-K(:,j:j+1)*H)*P_pred;
        polar_f(1,i) = sqrt(X(1,i)^2+X(3,i)^2); % filtered estimate of D
        polar_f(2,i) = atan(X(1,i)/X(3,i));     % filtered estimate of beta

        j = j + 2;
    end

    for i = 3:n
        %Errors of extrapolation estimates of range D
        error_estr(1,i) = (D(i)-polar_e(1,i))^2;
        Final_Error_estr(1,i)= Final_Error_estr(1,i)+error_estr(1,i);
        %Errors of filtration estimates of range D
        error_filt(1,i) = (D(i)-polar_f(1,i))^2;
        Final_Error_filt(1,i)= Final_Error_filt(1,i)+error_filt(1,i);

        %Errors of extrapolation estimates of azimuth beta
        error_estr(2,i) = (beta(i)-polar_e(2,i))^2;
        Final_Error_estr(2,i)= Final_Error_estr(2,i)+error_estr(2,i);
        %Errors of filtration estimates of azimuth beta
        error_filt(2,i) = (beta(i)-polar_f(2,i))^2;
        Final_Error_filt(2,i)= Final_Error_filt(2,i)+error_filt(2,i);
    end
end
Final_Error_estr = sqrt(Final_Error_estr/(M-1));
Final_Error_filt = sqrt(Final_Error_filt/(M-1));

figure
subplot(2,1,1)
sgtitle('Fig.16: Errors of estimation')
plot(Final_Error_estr(1,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(1,:),'linewidth',1.2)
plot(sqrt(var_D)*ones(1,n),'linewidth',1.2)
grid on
title('Error of estimates of range D')
legend('Extrapolation','Filtration','\sigma_D')
ylabel('Error estimation')
```

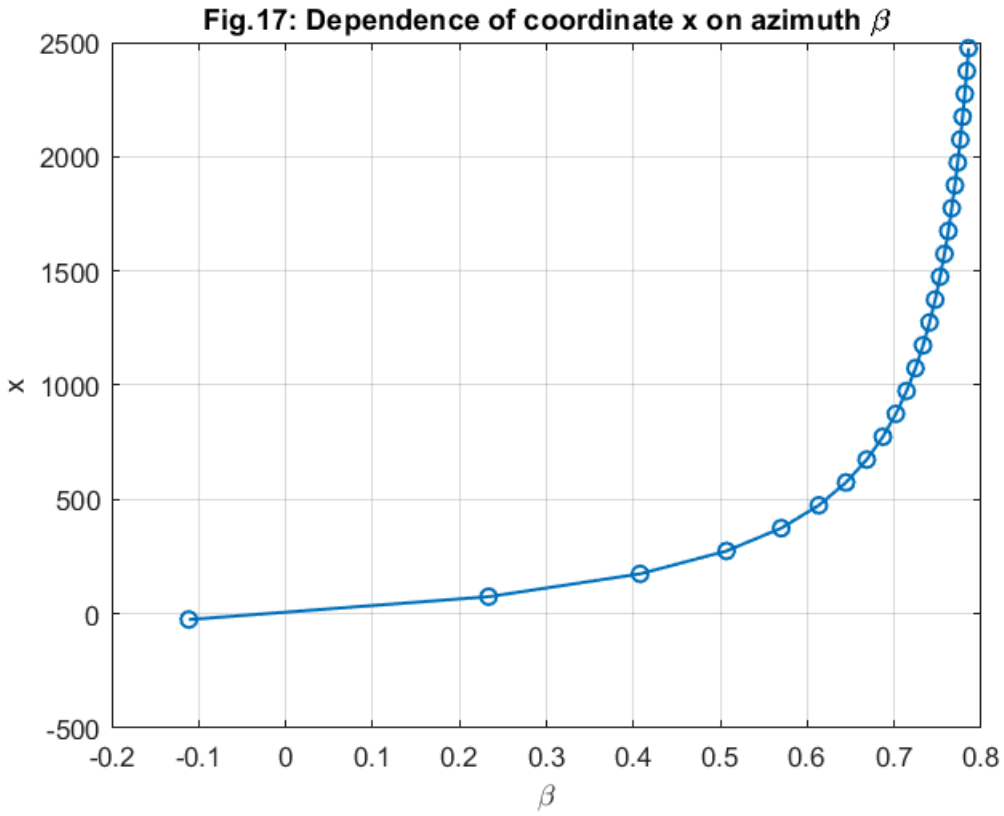
```
xlabel('Time-step')
xlim([5 30])

subplot(2,1,2)
plot(Final_Error_estr(2,:), 'linewidth',1.2)
hold on
plot(Final_Error_filt(2,:),'linewidth',1.2)
plot(sqrt(var_beta)*ones(1,n),'linewidth',1.2)
title('Error of estimates of azimuth \beta')
legend('Extrapolation','Filtration','\sigma_B')
ylabel('Error estimation')
xlabel('Time-step')
grid on
xlim([5 30])
```

Fig.16: Errors of estimation



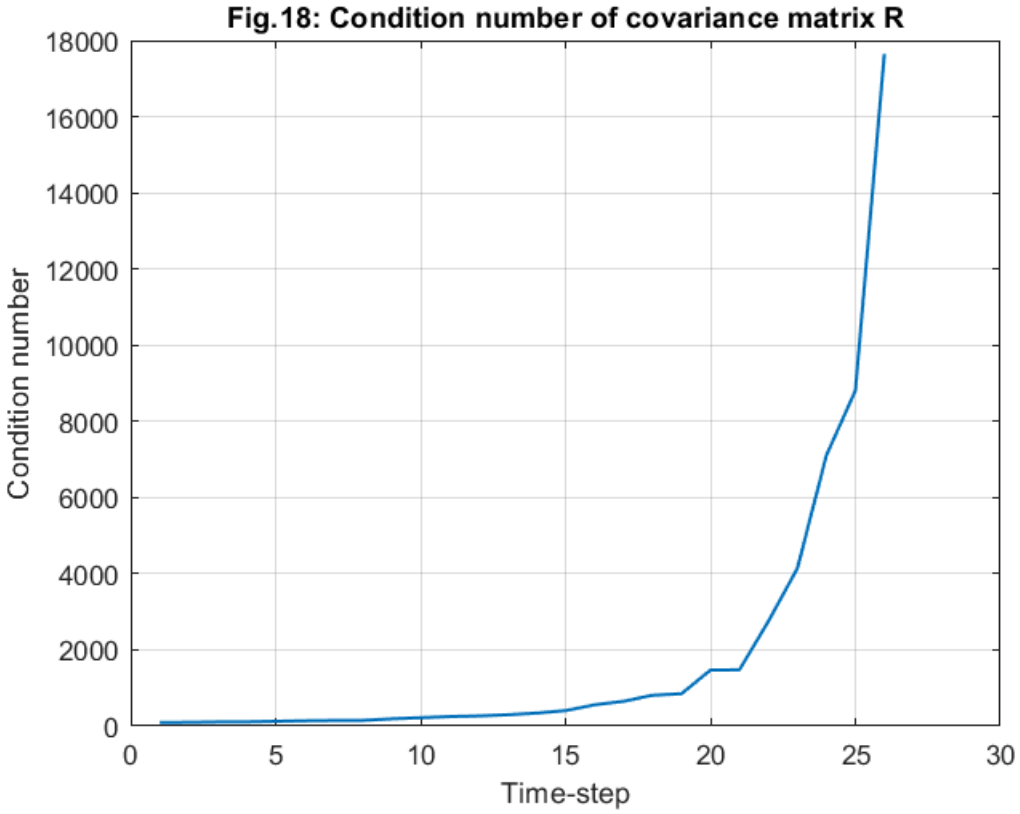
The estimation errors for range (D) decrease and then increase with increased time-steps, while the azimuth (beta) firstly remains constant with the azimuth standard deviation and then increases sharply as the final time-steps come.

```
%Analyze dependence of coordinate  on azimuth beta
figure
plot(beta,x(1,:),'-o','linewidth',1.2)
ylabel('x')
xlabel('\beta')
grid on
title('Fig.17: Dependence of coordinate x on azimuth \beta')
```



The dependence of coodinate x on azimuth beta is non-linear. This means that the linearization errors are significant.

```
%Calculate condition number of covariance matrix R
lambda_1 = var_D*ones(1,n);
lambda_2 = (D_meas.^2)*var_beta;
condition_number = zeros(1,n);

for i = 1:n
    if lambda_1(i)>lambda_2(i)
        condition_number(i) = lambda_1(i)/lambda_2(i);
    else
        condition_number(i) = lambda_2(i)/lambda_1(i);
    end
end

figure
plot(condition_number,'linewidth',1.2)
title('Fig.18: Condition number of covariance matrix R')
ylabel('Condition number')
xlabel('Time-step')
grid on
```



It can be observed that from the given set of conditions the condition number increases from ~93 to ~231 at first 10 time steps and then increases sharply to ~15.

Task 21: Make final conclusions under which conditions navigation system may become blind and filter may diverge.

In case when the object is close to observer, linearization errors are of great influence, hence, Kalman filter may diverge. Measurement error covariance matrix R is dependent on measurements and their variances only, so the only thing one can do to make it less ill-defined is to change the way of measuring. For instance, move the measuring station further from the object or add more stations to compare the results.

Conclusions:

When the object starts its motion at quite great distance from the observer:

- The true trajectory is generated for an object that moves uniformly. The trajectory is deterministic and has no random disturbances that affects the motion. The cartesian coordinates x, y, V_x and V_y are calculated using the given set of initial conditions.
- The true values of range (D) and azimuth (beta) are also generated with the given initial conditions. These are in polar coordinate system. The True trajectory plot is thus generated (Fig.1), and we can observe that the object is at great distance from the observer.
- The measurements for the range (D) and azimuth (beta) are calculated in the polar form with the given variances of measurement noise.

- Since the measurements are calculated and plotted in polar form (Fig.2), they are converted to Cartesian coordinate system in order to obtain the pseudo-measurements for the coordinates x and y, i.e. the two components of the pseudo-measurements vector (z). These results are plotted in Fig.3.
- The Kalman filter algorithm is developed for the given initial conditions. The Transition matrix (phi), observation matrix (H) and error covariance matrix (R) are created for the given state space model of the Kalman filter. Also, at each extrapolation and filtration step, the range (D) and azimuth (beta) are calculated. This is then run 500 times to compare the errors of extrapolation and filtration estimates of range (D) and azimuth (beta) relative to their standard deviations.
- Fig.6 shows the erros of estimation for estrapolation and filtered estimates. As it can be observed the trends for range (D) and azimuth (beta) is almost similar to each other for both extrapolated and filtered estimates but the overall error is quite low when compared with their standard deviations. The errors decreses as the time step increases.
- As the coordinate x is related with the sine of the azimuth (beta), the dependence of the coordinate x on azimuth (beta) is plotted (Fig.7). It is close to a linear trend and hence the linearization errors are insignificant.
- The condition number of a measurement error covariance matrix (R) gives us an idea whether the measurement error covariance matrix (R) is well conditioned or ill-conditioned. If condition number closer to 1, R is well-conditioned while if condition number is relatively great, then matrix R is ill-conditioned. An ill-conditioned measurement error covariance matrix (R) will lead to decrese in estimation accuracy and filter may get diverge in these conditions. From Fig.8 it can be observed that from the given set of conditions the condition number decreses over time from 181 to 102 and is not closer to 1, so it is a ill-conditioned matrix.

When the object starts its motion at quite close distance from the observer (changes in the initial conditions of coordinates):

- The true trajectory for new initial conditions of coordinates is generated and plotted (Fig.10). This takes into consideration when the object starts its motion at quite close distance from the observer.
- Upon keeping all the other paramenters same and running the Kalman filter for 500 runs, the errors of estimation for extrapolated and filtration estimates of range (D) and azimuth (beta), along with their corresponding variances, are plotted (Fig.11). It can be observed that the estimation errors for range (D) decreases with the increase in the time-steps, but the azimuth (beta) first decreses and then increases as the time-steps are incresed.
- In case of object quite close to the observer, the dependence of coodinate x on azimuth beta (Fig.12) is non-linear. This means that the linearization errors are significant when observer is relatively close to the object.
- From Fig.13 it can be observed that from the given set of conditions the condition number decreses over time from 12 to ~1 and then increases to 23 with increase in time step. Since the condition number reaches ~1 it can be a well-conditioned matrix.

When the object starts its motion at quite close distance from the observer (changes in the initial conditions of coordinates and variances of measurement noise):

- The true trajectory for new initial conditions of coordinates and variances of measurement noise is generated and plotted (Fig.14). This takes into consideration when the object starts its motion at quite

close distance from the observer. The polar coordinates is converted into Cartesian coordinates and plotted in Fig.15, as the true trajectory and the pseudo-measurements.

- Upon keeping all the other paramenters same and running the Kalman filter for 500 runs the errors of estimation for extrapolated and filtration estimates of range (D) and azimuth (beta), along with their corresponding variances are plotted (Fig.16). It can be observed that the estimation errors for range (D) decrease and then increase with increased time-steps, while the azimuth (beta) firstly remains constant with the azimuth standard deviation, then increases sharply as the final time-steps come.
- In case of object quite close to the observer, the dependence of coodinate x on azimuth beta, displayed in Fig.17, is non-linear. This means that the linearization errors are significant when observer is relatively close to the object.
- From Fig.18 it can be observed that from the given set of conditions the condition number increases from ~93 to ~231 at first 10 time steps and then increases sharply to ~15.