%Prashant V. Patil ARMS Lab

clear all;

close all;

clc;

global x\_1 y\_1 x\_2 y\_2 x\_3 y\_3;

state\_noise\_on=1;

measurement\_noise\_on=1;

input\_noise\_on=0;

samp\_T = 0.1;

n\_st = 3;

n\_op = 8;

n\_ip = 2;

n\_params = 2;

n\_op\_tri = n\_op;

n\_landmarks = 3;

N\_samples = 100;

%may be used for figures

start\_pt=2;

end\_pt=N\_samples;

set(groot,'defaultLineLineWidth',2)

set(0,'DefaultaxesLineWidth', 2)

set(0,'DefaultaxesFontSize', 16)

set(0,'DefaultTextFontSize', 16)

% set(0,'DefaultaxesFontName')

set(0,'defaultAxesXGrid','on')

set(0,'defaultAxesYGrid','on')

set(groot, 'defaultAxesTickLabelInterpreter','latex');

set(groot, 'defaultLegendInterpreter','latex');

set(groot,'DefaultAxesTitle', 'latex');

set(0,'DefaultFigureVisible','on')

disp('Figure Visibility is on')

y = zeros(n\_op,N\_samples);

X\_continuous = zeros(n\_st,N\_samples);

U = zeros(n\_ip,N\_samples);

param\_vect = zeros(n\_params,N\_samples);

landmark\_vect = ones(2\*n\_landmarks, N\_samples);

%triangulation

x\_1=8;y\_1=-5;

x\_2=0;y\_2=17;

x\_3=20;y\_3=5;

landmark\_vect(:,1) = [x\_1 y\_1 x\_2 y\_2 x\_3 y\_3]';

param\_vect(:,1) = [0 0]' ;

meas\_sigma = 0.01\*eye(n\_op) ;

state\_sigma = [ 0.001 0 0;0 0.01 0 ;0 0 0.0001] ; % Standard devistions of Unmeasured white noise, not used

input\_sigma = [ 0.00025 0; 0 0.00025] ;

vk = measurement\_noise\_on\*meas\_sigma\*randn(n\_op,N\_samples) ;

wk = state\_noise\_on\*state\_sigma\* randn(n\_st,N\_samples) ;

ip\_noise= input\_noise\_on\*input\_sigma\* randn(n\_ip,N\_samples) ;

%Filter Initializations

x\_hat\_LCSE\_tri = zeros(n\_st,N\_samples);

x\_pred\_LCSE\_tri = zeros(n\_st,N\_samples);

y\_pred\_LCSE\_tri = zeros(n\_op,N\_samples);

P\_pred\_trace\_LCSE\_tri=zeros(1,N\_samples);

P\_updt\_trace\_LCSE\_tri=zeros(1,N\_samples);

P\_pred\_spr\_LCSE\_tri=zeros(1,N\_samples); %Spectral Radius

P\_updt\_spr\_LCSE\_tri=zeros(1,N\_samples);

Q=0.001\*eye(n\_st);

gama\_d=eye(n\_st);

R=0.0015\*eye(n\_op);

k=1;

X\_continuous(:,1) = [5, 0 , 0]';

temp\_ip = [U(1,1)+param\_vect(1,k) U(2,1)+param\_vect(2,k)];

y(:,1) = h\_odom(temp\_ip, X\_continuous(:,1), landmark\_vect(:,1))+ vk(:,1);

clear temp\_ip

x\_initial\_LCSE\_tri=X\_continuous(:,1);

P\_initial\_LCSE\_tri=0.0025\*eye(n\_st);

x\_hat\_LCSE\_tri(:,k)=x\_initial\_LCSE\_tri;

x\_pred\_LCSE\_tri(:,k)=x\_initial\_LCSE\_tri;

P\_updt\_LCSE=P\_initial\_LCSE\_tri;

e\_LCSE\_tri=zeros(n\_op,N\_samples);

Z=[x\_hat\_LCSE\_tri(:,1);U(:,1);param\_vect(:,1)]; % Vector for linearization point, last two values are v and omega values

Jacobb=Numerical\_Jacobian\_Calc(@syst\_jacob\_diff\_odom,Z); %linearizing the model for every time instant

A\_LCSE\_tri=Jacobb(:,1:n\_st);

B\_LCSE\_tri=Jacobb(:,n\_st+1:n\_st+n\_ip);

G1\_c = Jacobb(:,n\_st+n\_ip+1:n\_st+n\_ip+n\_params);

stack1 = [x\_pred\_LCSE\_tri(:,1); landmark\_vect(:,1);U(:,1); param\_vect(:,1)]; % those three affected r given through param\_vect(:,k) so index k for parameters is fine..

%...useful case when all parameters are changing and only knowledge of three is not available

jach = Numerical\_Jacobian\_Calc(@h\_odom\_jacob, stack1);

C\_LCSE\_tri\_c = jach(:,1:n\_st);

%no matrix needed for d\_f/d\_landmarks

D\_c = jach(:,n\_st+(2\*n\_landmarks)+1:n\_st+(2\*n\_landmarks)+n\_ip);

G2\_c = jach(:,n\_st+(2\*n\_landmarks)+n\_ip+1:n\_st+(2\*n\_landmarks)+n\_ip+n\_params);

c\_mod=ss(A\_LCSE\_tri,B\_LCSE\_tri,C\_LCSE\_tri\_c,D\_c);

sys\_D=c2d(c\_mod,samp\_T);

SYSC2 = ss(A\_LCSE\_tri, G1\_c, C\_LCSE\_tri\_c, G2\_c);

SYSD2 = c2d(SYSC2,samp\_T,'zoh');

phy\_LCSE=sys\_D.a;

gama\_LCSE\_tri=sys\_D.b;

H\_LCSE=sys\_D.c;

D\_disc=sys\_D.d;

D\_u = D\_disc;

G1\_LCSE = SYSD2.B;

G2\_LCSE = SYSD2.D;

%Measures of Performance

eps\_x\_LCSE\_tri=zeros(n\_st,N\_samples);

beta\_k\_LCSE\_tri=zeros(1,N\_samples);

s\_u\_LCSE\_tri=zeros(n\_st,N\_samples);

s\_l\_LCSE\_tri=zeros(n\_st,N\_samples);

RMSE\_LCSE\_tri=[0;0;0];

%Threshold calculation

alpha\_c=0.05;

xi\_1 = chi2inv(alpha\_c,n\_st)\*ones(N\_samples,1);

xi\_2 = chi2inv((1-alpha\_c), n\_st)\*ones(N\_samples,1);

beta\_hit\_LCSE\_tri=0;

%theta\_hat\_madapusi = 0.01\*ones(n\_params,N\_samples);

theta\_hat\_pp = zeros(n\_params,N\_samples);

theta\_hat\_madapusi\_filtered = zeros(n\_params,N\_samples);

theta\_hat\_madapusi\_filtered(:,1)= param\_vect(:,1);

theta\_hat\_madapusi\_filtered\_lp(:,1)= param\_vect(:,1);

lp\_filter\_param = [0.69 0.69 0.69 0.69]';

theta\_hat = zeros(n\_params,N\_samples);

theta\_hat\_madapusi = zeros(n\_params,N\_samples);

P\_theta\_pred = 10 \*eye(n\_params);

P\_theta\_updt = 10 \*eye(n\_params);

Q\_theta =0.0000\*ones(n\_params);%

R\_theta =0.00000\*ones(n\_params);%

forg\_fact\_theta =eye(n\_op);

path = zeros(2,N\_samples);

i = 1;

x\_fol(i) = 0;

y\_fol(i) = 0.6;

head\_fol = pi/2;

t\_2 = 0;

t\_dot = 0.5;

a = 5;

b = 5;

theta = 0;

theta\_sum = 0;

x\_lead = zeros(1,N\_samples);

y\_lead = zeros(1,N\_samples);

% path = [2.00 1.00;

% 1.25 1.75;

% 7.25 8.25;

% 9.25 8.75;

% 11.75 10.75;

% 12.00 10.00];

t\_2=0;

path= zeros(50,2);

for l=1:50

path(l,1) = a \* cos(2\*t\_2);

path(l,2) = a \* sin(2\*t\_2);

t\_2=t\_2 + 0.015;

end

robotInitialLocation = [X\_continuous(1,1); X\_continuous(2,1)];

robotGoal = path(end,:);

initialOrientation = X\_continuous(3,1);

robotCurrentPose = [robotInitialLocation ;initialOrientation];

controller = controllerPurePursuit;

controller.Waypoints = path;

controller.DesiredLinearVelocity = 0.7;

controller.MaxAngularVelocity = 2;

controller.LookaheadDistance = 0.3;

goalRadius = 0.1;

distanceToGoal = norm(robotInitialLocation - robotGoal);

vizRate = rateControl(1/samp\_T);

figure

% Determine vehicle frame size to most closely represent vehicle with plotTransforms

robot = differentialDriveKinematics("TrackWidth", 1, "VehicleInputs", "VehicleSpeedHeadingRate");

frameSize = robot.TrackWidth/0.8;

while( distanceToGoal > goalRadius && k<N\_samples )

k=k+1

if k<100

param\_vect(:,k) =[0.05 -0.05]' ;

else

param\_vect(:,k) =[-0.05 0.05]';

end

landmark\_vect(:,k) = [x\_1 y\_1 x\_2 y\_2 x\_3 y\_3]';

robotCurrentPose\_e = [x\_hat\_LCSE\_tri(1,k-1); x\_hat\_LCSE\_tri(2,k-1); x\_hat\_LCSE\_tri(3,k-1)];

[v, omega] = controller(robotCurrentPose\_e);

% Re-compute the distance to the goal

distanceToGoal = norm(robotCurrentPose\_e(1:2) - robotGoal(:));

% True State Simulation (Ground Truth)

% Get the robot's velocity using controller inputs

vel = derivative(robot, robotCurrentPose, [v + param\_vect(1,k) omega + param\_vect(2,k)]);

% Update the current pose

robotCurrentPose = robotCurrentPose + vel\*samp\_T;

% path(1,k) = a \* cos(t\_2);

%path(2,k) = b \* sin(2\*t\_2);

% % Control Law Section

% x\_lead(1,k) = a \* cos(t\_2);

% y\_lead(1,k) = b \* sin(2 \* t\_2);

%

% ang\_lead = atan2((y\_lead(1,k) - X\_continuous(2,k)), (x\_lead(1,k) - X\_continuous(1,k)));

% if ang\_lead < 0

% ang\_lead = ang\_lead + 2 \* pi;

% end

%

% theta\_old = theta;

% theta\_sum = theta\_sum + theta;

%

% % Errors

% dist = sqrt((x\_lead(1,k) - X\_continuous(1,k-1)) ^ 2 + (y\_lead(1,k) - X\_continuous(2,k-1)) ^ 2);

% theta = ang\_lead - X\_continuous(3,k-1);

% if theta < -pi

% theta = theta + 2 \* pi;

% else if theta > pi

% theta = theta - 2 \* pi;

% end

% end

%

% % Control law

% kp = 0.1;

% kd = 0.000;

% ki = 0.0010;

% v = 0.5;

% w = kp \* theta + kd \* (theta - theta\_old) + ki \* theta\_sum;

% kh = 0.4;

% %w = kh \* 2 \* v \* sin(theta)/dist;

% t\_2 = t\_2 + t\_dot;

%

%

%

%

U\_unicycle(:,k) = [v; omega];

v\_l = U\_unicycle(1,k) + 0.5\*(0.15)\*U\_unicycle(2,k);

v\_r = U\_unicycle(1,k) - 0.5\*(0.15)\*U\_unicycle(2,k);

U(:,k)=[v\_l v\_r]';

temp\_ip\_true = [U(1,k)+param\_vect(1,k) U(2,k)+param\_vect(2,k)];

[t, Xt] = ode45(@(t,Xt)syst\_diff\_ip(t,Xt,temp\_ip\_true), [0, samp\_T],X\_continuous(:,k-1));

X\_continuous(:,k) = robotCurrentPose;

%Triangulation True y generation

temp\_ip\_op = [U(1,k)+param\_vect(1,k) U(2,k)+param\_vect(2,k)];

y(:,k) = h\_odom(temp\_ip\_op, X\_continuous(:,k), landmark\_vect(:,k))+ vk(:,k); % Here it should be k

%% LCSE Code here. Refer to the pseudo code and type the relevant code

% here.

display('Paused.Refer to the pseudo code and type the relevant code here.');

pause;

Z=[x\_hat\_LCSE\_tri(:,k-1);U(:,k-1);param\_vect(:,1)]; % Vector for linearization point

Jacob=Numerical\_Jacobian\_Calc(@syst\_jacob\_diff\_odom,Z); %linearizing the model for every time instant

A\_LCSE\_tri=Jacob(:,1:n\_st);

B\_LCSE\_tri=Jacob(:,n\_st+1:n\_st+n\_ip);

G1\_c = Jacob(:,n\_st+n\_ip+1:n\_st+n\_ip+n\_params);

%% RLS algorithm

display('Paused. Refer to the algorithm and type the code to the RLS algorithm here')

pause;

display('You may contact the authors for this section of the code individually')

hold off

% Plot path each instance so that it stays persistent while robot mesh

% moves

plot(path(:,1), path(:,2),"k--d")

hold all

% Plot the path of the robot as a set of transforms

plotTrVec = [robotCurrentPose(1:2); 0];

plotRot = axang2quat([0 0 1 robotCurrentPose(3)]);

plotTransforms(plotTrVec', plotRot, "MeshFilePath", "groundvehicle.stl", "Parent", gca, "View","2D", "FrameSize", frameSize);

light;

xlim([-a-5, 13])

ylim([-b-5, 13])

xlabel('x(m)');

ylabel('y(m)');

waitfor(vizRate);

for l = 1:3

scatter(landmark\_vect(2\*l-1),landmark\_vect(2\*l), 'black', 'filled');

hold on;

end

end

hit\_LCSE\_tri=beta\_hit\_LCSE\_tri/N\_samples;

LCSE\_mean\_tri = mean(e\_LCSE\_tri');

LCSE\_cov\_tri = cov(e\_LCSE\_tri');

%%

start\_idx=1;

end\_idx=min(k,N\_samples);

prefix\_image\_index = 'IM';

%%

path\_i = '.\';

path\_for\_images\_jpg = path\_i;

path\_for\_images\_mat = path\_i;

im\_index = 1;

%%

set(0,'DefaultFigureVisible','on')

disp('Figure Visibility is on')

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot(x\_hat\_LCSE\_tri(1,start\_idx:end\_idx),'black--');

hold on;

plot(X\_continuous(1,start\_idx:end\_idx),'red');

hold on;

plot(s\_u\_LCSE\_tri(1,start\_idx:end\_idx),'green--')

hold on;

plot(s\_l\_LCSE\_tri(1,start\_idx:end\_idx),'green--')

xlabel('Sampling Instants');

ylabel('y^{r}(m)');

legend('Estimate(LCSE)', 'True State', 'Upper Bound', 'Lower Bound');

plot\_name = 'X';

title('LCSE: X');

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(figure(FigH), fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot(x\_hat\_LCSE\_tri(2,start\_idx:end\_idx),'black--');

hold on;

plot(X\_continuous(2,start\_idx:end\_idx),'red');

hold on;

plot(s\_u\_LCSE\_tri(2,start\_idx:end\_idx),'green--')

hold on;

plot(s\_l\_LCSE\_tri(2,start\_idx:end\_idx),'green--')

xlabel('Sampling Instants');

ylabel('y^{r}(m)');

legend('Estimate(LCSE)', 'True State', 'Upper Bound', 'Lower Bound');

plot\_name = 'Y';

title('LCSE: Y');

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(figure(FigH), fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot(x\_hat\_LCSE\_tri(3,start\_idx:end\_idx),'black--');

hold on;

plot(X\_continuous(3,start\_idx:end\_idx),'red');

hold on;

plot( s\_u\_LCSE\_tri(3,start\_idx:end\_idx),'green--')

hold on;

plot( s\_l\_LCSE\_tri(3,start\_idx:end\_idx),'green--')

xlabel('Sampling Instants');

ylabel('Heading Angle(rad)');

legend('Estimate(LCSE)', 'True State', 'Upper Bound', 'Lower Bound');

plot\_name = 'Heading';

title('LCSE: Heading');

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(figure(FigH), fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot(P\_pred\_spr\_LCSE\_tri(start\_idx:end\_idx),'black');

hold on;

plot(P\_updt\_spr\_LCSE\_tri(start\_idx:end\_idx),'red');

xlabel('Sampling Instants');

ylabel('Spectral Radius)');

legend('Predicted', 'Updated');

plot\_name = 'Spectral Radius';

title('LCSE: Spectral Radius');

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(figure(FigH), fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot(P\_pred\_trace\_LCSE\_tri(start\_idx:end\_idx),'black');

hold on;

plot(P\_updt\_trace\_LCSE\_tri(start\_idx:end\_idx),'red');

xlabel('Sampling Instants');

ylabel('Trace of Covariance Matrix)');

legend('Predicted', 'Updated');

plot\_name = 'Trace of Covariance Matrix';

title('LCSE: Trace of Covariance Matrix');

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(figure(FigH), fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot((param\_vect(1,start\_idx:end\_idx)-param\_vect(1,1)\*ones(1,end\_idx-start\_idx+1)),'black');hold on; % Delta D

plot(theta\_hat(1,start\_idx:end\_idx),'yellow-.');hold on;

plot(theta\_hat\_madapusi(1,start\_idx:end\_idx),'cyan:');

xlabel('Sampling Instants');

ylabel('\delta\_{v\_l}(m/s)');title('Unknown Input 1');

legend('True Value','RLS','Proposition 1');

plot\_name = 'vl';

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(FigH, fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

plot((param\_vect(2,start\_idx:end\_idx)-param\_vect(2,1)\*ones(1,-(start\_idx-end\_idx-1))),'black');hold on;

plot(theta\_hat(2,start\_idx:end\_idx),'yellow-.');hold on;

plot(theta\_hat\_madapusi(2,start\_idx:end\_idx),'cyan:');hold on;

xlabel('Sampling Instants');

ylabel('\delta\_{v\_r}(m/s)');title('Unknown Input 2');

legend('True Value','RLS','Proposition 1');

plot\_name = 'vr';

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(FigH, fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

subplot(2,1,1);

plot(U\_unicycle(1,start\_idx:end\_idx)); ylabel('v(m/s)');xlabel('Sampling Instants');

subplot(2,1,2);

plot(U\_unicycle(2,start\_idx:end\_idx)); ylabel('\omega(rad/s)');xlabel('Sampling Instants');

plot\_name = 'U';

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(FigH, fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

FigH = figure('Position', get(0, 'Screensize'));

F = getframe(FigH);

subplot(2,1,1);

plot(U(1,start\_idx:end\_idx)); ylabel('v\_l(m/s)');xlabel('Sampling Instants');

subplot(2,1,2);

plot(U(2,start\_idx:end\_idx)); ylabel('v\_r(m/s)');xlabel('Sampling Instants');

plot\_name = 'VLVR';

baseFileName\_jpg=strcat(prefix\_image\_index,plot\_name,'.jpg');

fullFileName\_jpg = fullfile(path\_for\_images\_jpg, baseFileName\_jpg);

saveas(FigH, fullFileName\_jpg);

baseFileName\_fig = strcat(prefix\_image\_index,plot\_name,'.fig');

fullFileName\_fig = fullfile(path\_for\_images\_mat, baseFileName\_fig);

saveas(FigH, fullFileName\_fig);

save([path\_i,'/image\_index.mat'],'im\_index','path\_i'); %Enclose the variable names in inverted commas