**Appendix E – Codes accompanying Question 5**

|  |
| --- |
| % Question 5 - Tune EKF  close all, clearvars, clc  load("cooplocalization\_finalproj\_KFdata.mat");    x0 = [10 0 pi/2 -60 0 -pi/2]';  u0 = [2 -pi/18 12 pi/25]';  Dt = 0.1;  n = size(x0,1);  steps = 1000;  seed = 100;  rng(seed);    Q = diag([.0015, .0015, 0.01, 0.001, 0.005, 0.01]);  P0 = diag([1 1 0.025 1 1 0.025]);    runs = 100;  EX = zeros(n, steps+1, runs);  p = 5;  EY = zeros(p, steps+1, runs);  PS = zeros(n, n, steps+1, runs);  SS = zeros(p, p, steps+1, runs);  fig1 = figure(1);  fig2 = figure(2);  fig3 = figure(3);  fig4 = figure(4);  enablePlotDuring = true;  for run = 1:runs  disp(['run #', num2str(run)]);    % generate truth for run  [x, y] = GenerateTruth(x0, u0, P0, Qtrue, Rtrue, Dt, steps, true);  t = (0:(length(x)-1))\*Dt;    % assume we can get exact measurement noise from  % specifications of sensors  R = Rtrue;    % Run filter for all time-steps of run #k  [x\_est, y\_est, P, S] = EKF(x0, P0, u0, y, Q, R, Dt);    % wrap angle diff too!!  ex = x - x\_est;  ex(3,:) = angdiff(x\_est(3,:),x(3,:));  ex(6,:) = angdiff(x\_est(6,:),x(6,:));  ey = y - y\_est;  ey(1,:) = angdiff(y\_est(1,:),y(1,:));  ey(3,:) = angdiff(y\_est(3,:),y(3,:));    % Plot error during monte carlo runs  if enablePlotDuring == true  PlotStates(fig1,t,ex, ['State Errors, Run ',num2str(run)], P);  PlotMeasurements(fig2,t,y,'Ground Truth Measurements');  PlotStates(fig3,t,x,'Ground Truth States');  PlotMeasurements(fig4,t,ey,['Ground Truth Measurement Errors, Run ',num2str(run)]);  end    % save run data from NEES/NIS tests  EX(:, :, run) = ex;  EY(:, :, run) = ey;  PS(:, :, :, run) = P;  SS(:, :, :, run) = S;  end    %% Calculate NEES and NIS statistics  [NEES\_bar, NIS\_bar] = CalcStats(EX, EY, PS, SS);    %--------------------------------------------------------------------------  % Plots for (a)  PlotStates(fig1,t,x - x\_est, ['State Errors, Run ',num2str(run)], P);  PlotMeasurements(fig2,t,y,'Ground Truth Measurements');  PlotStates(fig3,t,x,'Ground Truth States');  PlotMeasurements(fig4,t,y - y\_est,['Ground Truth Measurement Errors, Run ',num2str(run)]);    %--------------------------------------------------------------------------  % Plots for (b)  fig5 = figure(5);  alpha = 0.05;  PlotNees(fig5, NEES\_bar, runs, n, alpha);  %--------------------------------------------------------------------------  % Plots for (c)  fig6 = figure(6);  PlotNis(fig6, NIS\_bar, runs, p, alpha); |
| function [x\_est, y\_est, P, S] = EKF(x0, P0, u, y, Q, R, Dt)    % set simulation tolerances for ode45  opt = odeset('RelTol',1e-6,'AbsTol',1e-6);    n = size(x0, 1); % number of states  p = size(R, 1); % number of measurements  steps = size(y,2); % number of time steps    x\_est = zeros(n, steps); % state estimate vector  y\_est = zeros(p, steps); % measurement estimate vector  P = zeros(n, n, steps); % coveriance  S = zeros(p, p, steps);    % start with initial estimate of total state  % and covariance  x\_p = x0;  P\_p = P0;    for i=1:steps  %----------------------------------------------------------  % Prediction Step  % use full NL equations to estimate state at next time step  % using state at previous time step; since Wk is AWGN,  % its expected value is zero, set input to zero  wk = zeros(1,n);  [~, x\_m] = ode45(@NL\_DynModel, [0.0 Dt], x\_p', opt, u', wk);  x\_m = x\_m(end,:)';    x\_m(3) = wrapToPi(x\_m(3));  x\_m(6) = wrapToPi(x\_m(6));    % to calculate covariance, linearize "online"  % about current state estimate  [A\_t,B\_t,C\_t] = Linearize(x\_m, u);  [F, ~, H] = Discretize(A\_t, B\_t ,C\_t, Dt);  P\_m = F\*P\_p\*F' + Q;    x\_m(3) = wrapToPi(x\_m(3));  x\_m(6) = wrapToPi(x\_m(6));    % uese estimated state from NL ODEs; since Wk is AWGN,  % its expected value is zero, set to zero  vk = zeros(p,1);  y\_est(:,i) = NL\_MeasModel(x\_m, vk);  y\_est(1,i) = wrapToPi(y\_est(1,i));  y\_est(3,i) = wrapToPi(y\_est(3,i));    % calculate innovation vector  e\_y = y(:,i) - y\_est(:,i);  e\_y(1) = wrapToPi(e\_y(1));  e\_y(3) = wrapToPi(e\_y(3));    %----------------------------------------------------------  % Correction Step  % calculate gain using linearized measurement  % matrix H and covariance from Prediction Step  S\_p = H\*P\_m\*H' + R;  K = P\_m\*H'/S\_p;    % calculate posterior state estimate and covariance  x\_p = x\_m + K\*e\_y;  P\_p = (eye(n) - K\*H)\*P\_m;    x\_p(3) = wrapToPi(x\_p(3));  x\_p(6) = wrapToPi(x\_p(6));    % for each time-step, save estimate and covariance  x\_est(:,i) = x\_p;  P(:,:,i) = P\_p;  S(:,:,i) = 0.5\*(S\_p + S\_p');  end    end |
| function [x, y] = GenerateTruth(x0, u, P0, Q, R, Dt, steps, wrapOn)  opt = odeset('RelTol',1e-6,'AbsTol',1e-6);  useChol = true;  n = size(x0,1);  p = size(R,1);    x = zeros(n,steps+1);  y = zeros(p,steps+1);    % set initial conditions  dx = mvnrnd(zeros(1,n),P0);  x(:,1) = x0 + dx';  x(3,1) = wrapToPi(x(3,1));  x(6,1) = wrapToPi(x(6,1));    for i = 2:steps+1    % generate noisy state  if useChol==true  wk = chol(Q)\*randn(n,1);  else  wk = mvnrnd(zeros(1,n),Q)';  end  [~,next\_x] = ode45(@NL\_DynModel, [0 Dt], x(:,i-1)', opt, u', wk);    if wrapOn == true  % wrap angle to [-pi pi]  next\_x(3) = wrapToPi(next\_x(3));  next\_x(6) = wrapToPi(next\_x(6));  end  x(:,i) = next\_x(end,:)';  end    for i = 1:steps+1  % generate noisy measurement  if useChol==true  vk = chol(R)\*randn(p,1);  else  vk = mvnrnd(zeros(1,p),R)';  end  y(:,i) = NL\_MeasModel(x(:,i), vk);    if wrapOn == true  % wrap angle to [-pi pi]  y(1,i) = wrapToPi(y(1,i));  y(3,i) = wrapToPi(y(3,i));  end  end    end |
| function [NEES, NIS] = CalcStats(EX, EY, P, S)    steps = size(EX, 2);  runs = size(EX, 3);    NEES\_all = zeros(runs,steps);  NIS\_all = zeros(runs,steps);  NEES = zeros(1,steps);  NIS = zeros(1,steps);    for run=1:runs  for step=1:steps  NEES(step) = EX(:,step, run)' / P(:,:,step, run) \* EX(:,step, run);  NIS(step) = EY(:,step, run)' / S(:,:,step, run) \* EY(:,step, run);  end    NEES\_all(run,:) = NEES;  NIS\_all(run,:) = NIS;  end    % calculate mean at each time step  for i=1:steps  NEES(i) = mean(NEES\_all(:,i));  NIS(i) = mean(NIS\_all(:,i));  end  end |
| function PlotMeasurements(hdl,t,y, title)  figure(hdl)  ftSize = 10;  sgtitle(title,'FontSize',ftSize+2,'Interpreter','latex')  subplot(3,2,1)  plot(t,y(1,:))  ylabel('$\gamma\_{ag}$ (rad)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(y(1,:)) max(y(1,:))])  grid on    subplot(3,2,2)  plot(t,y(2,:))  ylabel('$\rho\_{ga}$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(y(2,:)) max(y(2,:))])  grid on    subplot(3,2,3)  plot(t,y(3,:))  ylabel('$\gamma\_{ga}$ (rad)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(y(3,:)) max(y(3,:))])  grid on    subplot(3,2,4)  plot(t,y(4,:))  ylabel('$\xi\_a$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(y(4,:)) max(y(4,:))])  grid on    subplot(3,2,[5,6])  plot(t,y(5,:))  ylabel('$\eta\_a$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(y(5,:)) max(y(5,:))])  grid on  end |
| function PlotNees(hdl, epsNEESbar, Nsimruns, n, alpha)    figure(hdl);  Nnx = Nsimruns\*n;    %%compute intervals:  r1x = chi2inv(alpha/2, Nnx )./ Nsimruns;  r2x = chi2inv(1-alpha/2, Nnx )./ Nsimruns;    figure(hdl)  plot(epsNEESbar,'ro','MarkerSize',6,'LineWidth',2),hold on  plot(r1x\*ones(size(epsNEESbar)),'r--','LineWidth',2)  plot(r2x\*ones(size(epsNEESbar)),'r--','LineWidth',2)  ylabel('NEES statistic, $\bar{\epsilon}\_x$','FontSize',14,'Interpreter','latex')  xlabel('time step, k','FontSize',14)  title('NEES Estimation Results','FontSize',14)  legend('NEES @ time k', 'r\_1 bound', 'r\_2 bound')  end |
| function PlotNis(hdl, epsNISbar, Nsimruns, p, alpha)    figure(hdl);    Nny = Nsimruns\*p;    %%compute intervals:  r1y = chi2inv(alpha/2, Nny )./ Nsimruns;  r2y = chi2inv(1-alpha/2, Nny )./ Nsimruns;    plot(epsNISbar,'bo','MarkerSize',6,'LineWidth',2),hold on  plot(r1y\*ones(size(epsNISbar)),'b--','LineWidth',2)  plot(r2y\*ones(size(epsNISbar)),'b--','LineWidth',2)  ylabel('NIS statistic, $\bar{\epsilon}\_y$','FontSize',14,'Interpreter','latex')  xlabel('time step, k','FontSize',14)  title('NIS Estimation Results','FontSize',14)  legend('NIS @ time k', 'r\_1 bound', 'r\_2 bound')  end |
| function PlotStates(hdl,t,x, title, P)    if nargin > 4  p = zeros(size(x));  for ind = 1:size(x,2)  p(:,ind) = 2\*sqrt(diag(P(:,:,ind)));  end  displayError = true;  else  displayError = false;  end    figure(hdl)  ftSize = 10;  sgtitle(title,'FontSize',ftSize+2,'Interpreter','latex')  subplot(3,2,1)  state = 1;  plot(t,x(state,:))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\xi\_g$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(x(state,:)) ...  max(x(state,:))])  grid on    subplot(3,2,3)  state = state + 1;  plot(t,x(state,:))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\eta\_g$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(x(state,:)) ...  max(x(state,:))])  grid on    subplot(3,2,5)  state = state + 1;  plot(t,wrapToPi(x(state,:)))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\theta\_g$ (rad)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(wrapToPi(x(state,:))) ...  max(wrapToPi(x(state,:)))])  grid on    subplot(3,2,2)  state = state + 1;  plot(t,x(state,:))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\xi\_a$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(x(state,:)) ...  max(x(state,:))])  grid on    subplot(3,2,4)  state = state + 1;  plot(t,x(state,:))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\eta\_a$ (m)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(x(state,:)) ...  max(x(state,:))])  grid on    subplot(3,2,6)  state = state + 1;  plot(t,wrapToPi(x(state,:)))  if displayError == true  hold all, plot(p(state,:),'b--'), plot(-p(state,:),'b--'), hold off  end  ylabel('$\theta\_a$ (rad)','FontSize',ftSize,'Interpreter','latex')  xlabel('Time(s)','FontSize',ftSize,'Interpreter','latex')  axis([min(t) max(t) min(wrapToPi(x(state,:))) ...  max(wrapToPi(x(state,:)))])  grid on  end |