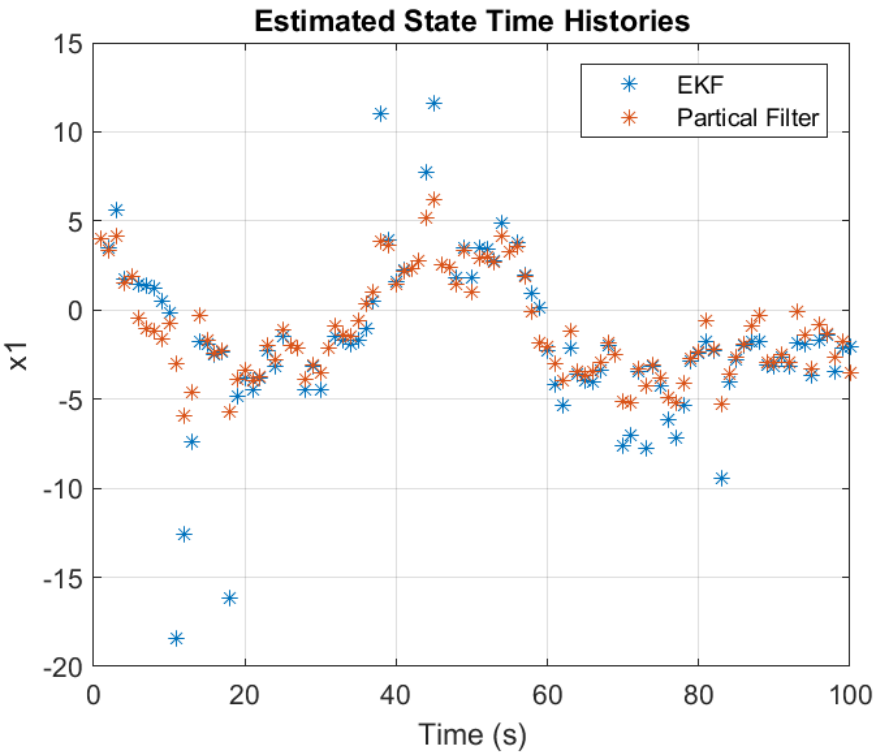
HW8 Problem 4 Final

Spencer Freeman

AOE 5784

12/17/2024

Results (based in part on random draw):



Script hw8_prob34_final.m (includes local functions defined at the bottom):

```
%% Solve the particle filtering example problem that was presented in lecture. Re-do Problem 3 using an extended Kalman filter
% Spencer Freeman, 12/17/2024
% AOE 5784, Estimation and Filtering
%
% This script solves number 4 of problem set 8
% ------------------------------------------------------------------------
clear;clc;close all

disp('HW8-P4_final')

%% load data
load('measdata_pfexample.mat')

n = length(zkhist); % samples
nx = length(xhat0);
nv = size(Q, 1);
thist = 1:n;

%% filter the data
[xhathist_pft,Phist,sigmahist,enuhist] = ...
   particle_filter(zkhist, xhat0, P0, Q, R);
```

```
[xhathist_ekf,Phist,sigmahist,enuhist] = ...
   efk(zkhist, xhat0, P0, Q, R);

%% plotting
close all

% time histories
names = "x1";

fig = figure;
fig.WindowStyle = 'Docked';
for i = 1:nx
   subplot(nx, 1, i)
   plot(thist, xhathist_ekf(:, i), '*'); hold on; grid on
   plot(thist, xhathist_pft(:, i), '*'); hold on; grid on
   ylabel(names(i))
   if i == 1
      title('Estimated State Time Histories')
      legend('EKF', 'Partical Filter')
   end % if
end % for
xlabel('Time (s)')
grid on


%% functions

% Particle Filter ------------------------------------------------------------
function [xhathist,Phist,sigmahist,enuhist] = ...
   particle_filter(zkhist, xhat0, P0, Q, R)

n = length(zkhist); % samples
nx = length(xhat0);
nv = size(Q, 1);
thist = 1:n;

t = 0; % s
xhat = xhat0; % initial state estimate
phat = P0; % initial state covariance
ev = 0;

ts = nan(1, n);
xhats_pft = nan(nx, n);
phats_pft = nan(nx * nx, n);
evs = nan(1, n);

Rinv = inv(R);

Ns = 400; % # of particles

w0 = 1/Ns * ones(1, Ns); % initial weights
chi0 = chol(P0)'*randn(nx, Ns) + xhat0; % initial particles

w = w0;
chi = chi0;
for i = 1:n

   ts(i) = t;
   xhats_pft(:, i) = xhat;
   phats_pft(:, i) = phat(:); % unwrap to column vector
   % evs(i) = ev;

   vss = chol(P0)'*randn(nv, Ns);

   z = zkhist(i);

   log_wtil = nan(1, Ns);
   for j = 1:Ns
      chi(:, j) = f_class_example(i, chi(:, j), vss(:, j)); % propagate to k+1
```

```
    log_wtil(j) = log(w(j)) - .5 * (z - h_class_example(chi(:, j)))' * Rinv * (z - h_class_example(chi(:, j)));
    % wtil(j) = w(j) * exp( -.5 * (z - h(chi(:, j)))' * Rinv * (z - h(chi(:, j))) );
  end % for
  log_wtil_max = max(log_wtil);
  wtiltil = exp(log_wtil - log_wtil_max);

  w = wtiltil / sum(wtiltil); % normalized weights

  xhat = sum(w .* chi, 2); % compute a posteriori state estimate
  phat = zeros(nx);
  for j = 1:Ns
    phat = phat + w(j) * (chi(:, j) - xhat)*(chi(:, j) - xhat)'; % compute a posteriori error covariance matrix
  end % for

  % resampling
  c = nan(1, Ns + 1);
  c(1) = 0;
  c(end) = 1 + 10^-10;
  for j = 2:Ns
    c(j) = sum(w(1:j - 1));
  end % for

  chi_new = nan(nx, Ns);
  for l = 1:Ns
    nl = rand;
    ind = find(nl >= c, 1, 'last');
    chi_new(:, l) = chi(:, ind);
  end % for

  chi = chi_new;
  w = w0;

end % for

% record the final filter outputs
ts(n) = t;
xhats_pft(:, n) = xhat;
phats_pft(:, n) = phat(:); % unwrap to column vector

xhathist = xhats_pft';
Phist = [];
sigmahist = [];
enuhist = [];


end % function



% Extended Kalman Filter ------------------------------------------------
function [xhathist,Phist,sigmahist,enuhist] = ...
  efk(zkhist, xhat0, P0, Q, R)

n = length(zkhist); % samples
nx = length(xhat0);
nv = size(Q, 1);
thist = 1:n;

% EKF
t = 0; % s
xhat = xhat0; % initial state estimate
phat = P0; % initial state covariance
ev = 0;

ts = nan(1, n);
vs = nan(1, n);
xhats = nan(nx, n);
phats = nan(nx * nx, n);
evs = nan(1, n);
```

```
for i = 1:(n - 1)

  ts(i) = t;
  xhats(:, i) = xhat;
  phats(:, i) = phat(:); % unwrap to column vector
  evs(i) = ev;

  % propagate
  % tkp1 = thist(i); % s

  % [fprinted, dfprinted_dxk, dfprinted_dvk] = ...
  %    c2dnonlinear(xhat, [], [0; 0], t, tkp1, nRK, fscriptname, true);

  xbar = f_class_example(i, xhat, 0);
  F = 2*sec(xhat)^2; % df / dxk
  GAMMA = 1;

  pbar = F * phat * F' + GAMMA * Q * GAMMA';
  % t = tkp1;

  % measurement update
  zbar = h_class_example(xbar);
  H = 1 + 2*xbar + 3*xbar^2; % dh /dx

  z = zkhist(i);
  v = z - zbar; % innovation
  S = H * pbar * H' + R; Sinv = inv(S);
  W = pbar * H' * Sinv;

  xhat = xbar + W * v;
  phat = pbar - W * S * W';

  ev = v' * Sinv * v; % estimation error statistic

end % for

% record the final filter outputs
ts(n) = t;
xhats(:, n) = xhat;
phats(:, n) = phat(:); % unwrap to column vector
evs(n) = ev;

xhathist = xhats';
Phist = reshape(phats, nx, nx, n);
sigmahist = [];
enuhist = [];



end % function


% nonlinear dynamics function class example ------------------------------
function xkp1 = f_class_example(k, x, v)
xkp1 = 2*atan(x) + .5*cos(pi*k/3) + v;
end % function

% nonlinear measurement function class example ----------------------------
function z = h_class_example(x)
z = x + x.^2 + x.^3;
end % function
```