HW8 Problem 4 Final

Spencer Freeman

AOE 5784

12/17/2024

**Results (based in part on random draw):**

**A graph with red and blue dots

Description automatically generated**

**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