Kalman Filter Homework 3

Table of Contents

Robotics Motion Planning, Spring 2016, Prof.Yap/Dr.Ciconnet

INTRODUCTION

This setup file must be executed BEFORE running the kalman tests.

Consider the linear system:

```
             x = Ax + w
             z = Hx + v
```

where

```
             x = state (dimension n),
             z = observation (dimension m),
```

```
% A, H are given matrices,
%  w and v are Gaussian noise with mean zero with
%  covariance matrices
%  Q = cov(w) and R = cov(v).
%
%  In the discrete setting, the states should be subscripted
%  x_1, x_2, etc.   But we omit it here for simplicity.
%
% For our physical example, imagine a particle moving along
% the real line (x-axis).  At time t, it is at distance s(t)
% from the origin, and its velocity is v(t).
% Thus, our state
%
%  x = [s; v]
%
% where s and v are are above.
%
% We model the situation where the velocity is decaying at
% a constant rate delta (0<delta<1). So the update equation for v is
%
% v = v.delta
%
% Also the update equation for s is
%
% s = s + v.rho
%
```

```
% For some rho (0<rho).  Think of rho as the discrete sampling rate.
% Therefore the matrix 2x2 matrix A in Matlab notation is simply
%
% A =    [ [1 rho]; [0 delta]]
%
% As for observation z, we assume we are directly observing x.
% Thus
% z = H.x + v
% where
%  H is identity matrix (H = eye(2) in Matlab).
%
% All these variables (and more) is collected in a
% Matlab "structure" which we call ppp (for "parameters").
%
% How to treat noise:
% Assume that the covariance matrices Q (nxn), R (mxm)
%  are proportional to identity matrices:
%  Q = eye(n)*q
%  R = eye(n)*r
% where q and r are some positive values
% But q and r are unknown!
% So we estimate by some value qh (qhat) and rh (rhat).
% The initial state covariance matrix P (nxn) is also
% of this form:
%  P = eye(n)*p.
%
% The only actual noise we need to actually generate are to
% perturb the observations.
% To ensure repeatability of our experiments, we need to
%  (1) fix a seed for the random number generator
%  (2) store the sequences of process noise (PN)
%   and observation noise (ON).
%   They are generated using the parameters q,r.
% We store the sequence of actual (noisy) states in an array XX:
%   x(k+1) = A*x(k) + PN(k)
% Then the sequence of observations is simply
%   ZZ = XX + ON
% The Kalman filter will produce two sequences:
%  XH = estimated states (x-hat)
%  PH = covariance of x-hat's
%   These are generated using estimates for
%   q and r (called qh and rh)
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% But since we want to run our experiments with different choices
% of the parameters, we need to make ppp into an array.
% We will specify ppp(1) for you.  For your experiments, you
% can define ppp(2), ppp(3), etc.
```

Default set of parameters, ppp(1):

Please DO NOT modify any of the following initialization.

```
clear

i=1; % i refers to the first set of parameters
n=2; % state dimension
m=2; % observation dimension

ppp(i).seed = 111;     % random seed to ensure repeatable tests
ppp(i).N = 15;         % number of steps
% Physical Problem
ppp(i).n = n;          % number of variables in state x
ppp(i).A = eye(n);     % matrix A (temporarily the identity)
ppp(i).m = m;          % number of variables in observation z
ppp(i).H = eye(m,n); % observation matrix H (temporarily identity)
ppp(i).delta = 0.5;   % decay rate
ppp(i).rho = 0.2;   % sampling rate
ppp(i).s0 = 0;   % initial distance
ppp(i).v0 = 10;   % initial speed
% Noise parameters:
ppp(i).ph = 0.2; % initial uncertainty (p-hat)
ppp(i).q = 0.2;  % process noise
ppp(i).r = 0.3;  % observation noise
ppp(i).qh = 0.2; % estimated process noise (q-hat)
ppp(i).rh = 0.3; % estimated observation noise (r-hat)
% Time Sequences: XI, XX, ZZ, PN, ON, XH, PH
% such that for i=1,...,N,
%  XI(i,:) = the ideal noise-free state,
%  XX(i,:) = x-hat, the actual (unknown) state, "XI+PN"
%  ZZ(i,:) = x-hat, the observed state,  "XX+ON"
%  PN(i,:) = v, the process noise in XX
%  ON(i,:) = w, the observation noise in ZZ
%  XH(i,:) = x-hat, the KF estimated state
%  PH{i}   = P, the covariance of x-hat
%
ppp(i).XI = zeros(2, ppp(i).N); % XI(:,k) is the ideal noise-free
 state at time k
ppp(i).XX = zeros(2, ppp(i).N); % XX(:,k) is the actual state at time
 k
ppp(i).ZZ = zeros(2, ppp(i).N); % ZZ(:,k) is the observation at time k

ppp(i).PN = zeros(2, ppp(i).N); % PN(:,k) is the process noise at time
 k
ppp(i).ON = zeros(2, ppp(i).N); % ON(:,k) is the observation noise at
 time k

ppp(i).XH = zeros(2, ppp(i).N); % XX(:,k) is the estimated state at
 time k
ppp(i).PH = cell(1, ppp(i).N); % PH{k} is estimated covariance matrix
 P at time k
    % NOTE: unlike the other arrays, PH is a cell array.

% The first entry in XH and PH is somewhat arbitrary, and
%  randomly determined by ppp(i).ph:
%  Please do not accidentally delete this first entry.
ppp(i).XH(:,1) = [ppp(i).s0; ppp(i).v0] + ppp(i).ph *randn(2,1);
ppp(i).PH{1}   = ppp(i).ph *randn(2,1);
```

SOME initializiation

```
% Setup the matrix A, H, P, Q, R
ppp(i).A = [[1  ppp(i).rho]; [0 ppp(i).delta]];
ppp(i).H = eye(m,n);
ppp(i).P = ppp(i).ph * eye(ppp(i).n);
ppp(i).Q = ppp(i).qh * eye(ppp(i).n);
ppp(i).R = ppp(i).rh * eye(ppp(i).m);

% Initialize the random parameters:
rng(ppp(i).seed);    % random number generator
ppp(i).PN = ppp(i).q * randn(size(ppp(i).PN)); % process noise array
ppp(i).ON = ppp(i).r * randn(size(ppp(i).ON)); % observation noise
 array

ppp(i).PH{1} = ppp(i).P;   % covariance P at time t=1

% NOTE: Q,R are estimates based on qh, rh;
%  but the noise PN,ON are based on q, r.
```

Additional set of parameters

BELOW, you can add your own set of parameters

```
% for ppp(2), ppp(3), etc.
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
i=2; % second set of parameters (illustrative)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=2; % state dimension
m=4; % observation dimension (we can repeat an observations, etc)

ppp(i).seed = 111;     % random seed to ensure repeatable tests
ppp(i).N = 40;         % number of steps (perhaps we now need more
 steps...)
% Physical Problem
ppp(i).n = n;          % number of variables in state x
ppp(i).A = eye(n);     % matrix A (temporarily the identity)
ppp(i).m = m;          % number of variables in observation z
ppp(i).H = eye(m,n); % observation matrix H (temporarily identity)

% ...etc
%
```

Published with MATLAB® R2015b