AOE 5784 Final Exam Solution
Fall 2022

1) Problems 3.7.1 & 3.7.2 in Bar-Shalom [15 pts]

3.7.1: $\bar{z} = E[x+w] = \bar{x}$

$$P_{xx} = \sigma_0^2$$
$$P_{xz} = E[(x-\bar{x})(z-\bar{z})]$$
$$= E[(x-\bar{x})(\{x-\bar{x}\}+w)]$$
$$= E[(x-\bar{x})(x-\bar{x})] + E[(x-\bar{x})w]$$
$$= \sigma_0^2 + \rho\sigma_0\sigma_w$$
$$P_{zz} = E[(z-\bar{z})(z-\bar{z})]$$
$$= E[(\{x-\bar{x}\}+w)(\{x-\bar{x}\}+w)]$$
$$= E[(x-\bar{x})(x-\bar{x})] + 2E[(x-\bar{x})w] + E[ww]$$
$$= \sigma_0^2 + 2\rho\sigma_0\sigma_w + \sigma_w^2$$

$$\hat{x} = \bar{x} + P_{xz}P_{zz}^{-1}(z-\bar{z})$$

$$\therefore \boxed{\hat{x} = \bar{x} + \left[\frac{(\sigma_0^2+\rho\sigma_0\sigma_w)}{\sigma_0^2+2\rho\sigma_0\sigma_w+\sigma_w^2}\right][z-\bar{x}]}$$

3.7.2: $\text{MMSE} = P_{xx} - P_{xz}P_{zz}^{-1}P_{xz}^T$

$$= \sigma_0^2 - \frac{(\sigma_0^2+\rho\sigma_0\sigma_w)^2}{\sigma_0^2+2\rho\sigma_0\sigma_w+\sigma_w^2}$$

$$= \frac{[\sigma_0^4+2\rho\sigma_0^3\sigma_w+\sigma_0^2\sigma_w^2]-[\sigma_0^4+2\rho\sigma_0^3\sigma_w+\rho^2\sigma_0^2\sigma_w^2]}{\sigma_0^2+2\rho\sigma_0\sigma_w+\sigma_w^2}$$

or

$$\boxed{MMSE = \frac{(1-\rho^2)\sigma_v^2\sigma_w^2}{\sigma_v^2 + 2\rho\sigma_v\sigma_w + \sigma_w^2} = \frac{(1-\rho^2)\sigma_v^2}{1+2\rho\left(\frac{\sigma_v}{\sigma_w}\right)+\left(\frac{\sigma_v}{\sigma_w}\right)^2}}$$

2) Problem 5-5.1 in Bar-Shalom [15 PTS]

$$\hat{\underline{x}}(k+1|k+1) = \hat{\underline{x}}(k+1|k) + W(k+1)\left[\underline{z}(k+1) - H(k+1)\hat{\underline{x}}(k+1|k)\right]$$
$$= \left[I - W(k+1)H(k+1)\right]\hat{\underline{x}}(k+1/k) + W(k+1)\underline{z}(k+1)$$

for arbitrary $W(k+1)$

$$\tilde{\underline{x}}(k+1|k+1) = \underline{x}(k+1) - \hat{\underline{x}}(k+1|k+1)$$
$$\tilde{\underline{x}}(k+1|k) = \underline{x}(k+1) - \hat{\underline{x}}(k+1|k)$$

$$\tilde{\underline{x}}(k+1|k+1) = \underline{x}(k+1) - \left[I - W(k+1)H(k+1)\right]\hat{\underline{x}}(k+1|k)$$
$$- W(k+1)\underline{z}(k+1)$$
$$= \underline{x}(k+1) - \left[I - W(k+1)H(k+1)\right]\hat{\underline{x}}(k+1/k)$$
$$- W(k+1)\left\{H(k+1)\underline{x}(k+1) + \underline{w}(k+1)\right\}$$
$$= \left[I - W(k+1)H(k+1)\right]\underline{x}(k+1) - \left[I - W(k+1)H(k+1)\right]\hat{\underline{x}}(k+1|k)$$
$$- W(k+1)\underline{w}(k+1)$$
$$= \left[I - W(k+1)H(k+1)\right]\tilde{\underline{x}}(k+1|k)$$
$$- W(k+1)\underline{w}(k+1)$$

Therefore

$$P(k+1/k+1) = [I - W(k+1)H(k+1)]E\{\tilde{x}(k+1/k)\tilde{x}^T(k+1/k)\} \cdot \ldots$$
$$[I - W(k+1)H(k+1)]^T$$
$$- [I - W(k+1)H(k+1)]E\{\tilde{x}(k+1/k)\underline{w}^T(k+1)\}W^T(k+1)$$
$$- W(k+1)E\{\underline{w}(k+1)\tilde{x}^T(k+1/k)\}[I - W(k+1)H(k+1)]^T$$
$$+ W(k+1)E\{\underline{w}(k+1)\underline{w}^T(k+1)\}W^T(k+1)$$

but
$$E\{\tilde{x}(k+1/k)\tilde{x}^T(k+1/k)\} = P(k+1/k)$$
$$E\{\tilde{x}(k+1/k)\underline{w}^T(k+1)\} = 0$$
$$E\{\underline{w}(k+1)\underline{w}^T(k+1)\} = R(k+1)$$

There fore

$$P(k+1/k+1) = [I - W(k+1)H(k+1)]P(k+1/k)[I - W(k+1)H(k+1)]^T$$
$$+ W(k+1)R(k+1)W^T(k+1)$$

Regardless of whether or not $W(k+1)$ is the optimal Kalman Filter gain matrix

3) Problems 10-3.1 & 10-3.2 in Bar-Shalom [15 pts]

10-3.1:

$$x_1 = \rho \cos\theta$$
$$x_2 = \rho \sin\theta$$

$$y = \begin{bmatrix} z_1 \cos z_2 \\ z_1 \sin z_2 \end{bmatrix}$$

10-3.2:

$$\underline{w}_c = \underline{y} - \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} \{z_1 \cos z_2 - \rho\cos\theta\} \\ \{z_1 \sin z_2 - \rho\sin\theta\} \end{bmatrix}$$

$$= \begin{bmatrix} \{(\rho + w_1)\cos(\theta + w_2) - \rho\cos\theta\} \\ \{(\rho + w_1)\sin(\theta + w_2) - \rho\sin\theta\} \end{bmatrix}$$

The linearization of the dependence of $\underline{w}_c$ on w_1 & w_2 expands this formula for $\underline{w}_c$ in a 1st-order Taylor Series in the small quantities w_1 & w_2:

$$(\rho + w_1)\cos(\theta + w_2) \cong \rho\cos\theta + w_1\cos\theta - \rho\sin\theta\, w_2$$

$$(\rho + w_1)\sin(\theta + w_2) \cong \rho\sin\theta + w_1\sin\theta + \rho\cos\theta\, w_2$$

Substitution back into the formula for $\underline{w}_c$ yields

$$\underline{w}_c \cong \begin{bmatrix} \{\rho \cos\theta + w_1 \cos\theta - \rho \sin\theta\, w_2 - \rho\cos\theta\} \\ \{\rho \sin\theta + w_1 \sin\theta + \rho\cos\theta\, w_2 - \rho \sin\theta\} \end{bmatrix}$$

or

$$\underline{w}_c \cong \begin{bmatrix} \cos\theta & -\rho\sin\theta \\ \sin\theta & \rho\cos\theta \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Linearizing about $\rho_0 = 10^5\,m$, $\theta_0 = 45°$ this relationship becomes

$$\underline{w}_c \cong \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-10^5}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{10^5}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} =$$

and

$$R_c = E\{\underline{w}_c \underline{w}_c^T\} \cong \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -10^5 \\ 1 & 10^5 \end{bmatrix} \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -10^5 & 10^5 \end{bmatrix} \frac{1}{\sqrt{2}}$$

$$\cong \frac{1}{2} \begin{bmatrix} \sigma_\rho^2 & -10^5\sigma_\theta^2 \\ \sigma_\rho^2 & 10^5\sigma_\theta^2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -10^5 & 10^5 \end{bmatrix}$$

or

$$R_c \cong \begin{bmatrix} \left\{\dfrac{\sigma_\rho^2 + 10^{10}\sigma_\theta^2}{2}\right\} & \left\{\dfrac{\sigma_\rho^2 - 10^{10}\sigma_\theta^2}{2}\right\} \\ \left\{\dfrac{\sigma_\rho^2 - 10^{10}\sigma_\theta^2}{2}\right\} & \left\{\dfrac{\sigma_\rho^2 + 10^{10}\sigma_\theta^2}{2}\right\} \end{bmatrix}$$

where σ_θ must be expressed in radians in order for this formula to be correct. If $\sigma_{\theta D}$ is expressed in degrees, then the correct value to use in the above formula is $\sigma_\theta = \sigma_{\theta D}\left(\frac{\pi}{180}\right)$.

4) Problem #5 of Problem Set 7 [20 pts]

Assume the use of the same linearized dynamics and measurement models as were used in the EKF forward filtering pass wherever that would be appropriate

$$\underline{x}(k+1) \cong \underline{f}\left[k, \underline{\hat{x}}(k), \underline{u}(k), 0\right] + F(k)\left[\underline{x}(k) - \underline{\hat{x}}(k)\right] + \Gamma(k)\underline{v}(k)$$

or

$$\underline{x}(k+1) - \underline{\bar{x}}(k+1) = F(k)\left[\underline{x}(k) - \underline{\hat{x}}(k)\right] + \Gamma(k)\underline{v}(k)$$

where $\underline{\bar{x}}(k+1) = \underline{f}\left[k, \underline{\hat{x}}(k), \underline{u}(k), 0\right]$

$$F(k) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{k, \bar{\underline{x}}(k), \underline{u}(k), 0}$$

$$\Gamma(k) = \left.\frac{\partial \underline{f}}{\partial \underline{v}}\right|_{k, \bar{\underline{x}}(k), \underline{u}(k), 0}$$

The first linearized form of the dynamic model is exactly the same as the model in the original linear KF/Smoother derivation except that the terms

$$\underline{f}\left[k, \bar{\underline{x}}(k), \underline{u}(k), 0\right] - F(k)\bar{\underline{x}}(k)$$

replace the term $G(k)\underline{u}(k)$ in the original linear KF/Smoother model.

$$\underline{z}(k+1) \cong \underline{h}\left[k+1, \bar{\underline{x}}(k+1)\right] + H(k+1)\left[\underline{x}(k+1) - \bar{\underline{x}}(k+1)\right]$$

$$+ \underline{w}(k+1)$$

where $H(k+1) = \left.\frac{\partial \underline{h}}{\partial \underline{x}}\right|_{k+1, \bar{\underline{x}}(k+1)}$

Rearranging terms

$$\underline{z}(k+1) - \underline{h}\left[k+1, \bar{\underline{x}}(k+1)\right] + H(k+1)\bar{\underline{x}}(k+1)$$

$$\cong H(k+1)\underline{x}(k+1) + \underline{w}(k+1)$$

This exactly the same formula as in the derivation of the linear KF/Smoother if one defines

$$\underline{z}_{EKF}(k+1) = \underline{z}(k+1) - \underline{h}[k+1, \bar{\underline{x}}(k+1)] + H(k+1)\bar{\underline{x}}(k+1)$$

and if one uses this quantity in place of $\underline{z}(k+1)$ in all of the K.F. & smoother calculations.

Therefore, the forward EKF pass with the replacements of $\underline{f}[k, \hat{\underline{x}}(k), \underline{u}(k), 0] - F(k)\hat{\underline{x}}(k)$ for $G(k)\underline{u}(k)$ and $\underline{z}(k+1) - \underline{h}[k+1, \bar{\underline{x}}(k+1)] + H(k+1)\bar{\underline{x}}(k+1)$ for $\underline{z}(k+1)$ becomes

① Set $k=0$, $\hat{\underline{x}}(k) = \hat{\underline{x}}(0)$, & $P(k) = P(0)$

② Compute

$$\bar{\underline{x}}(k+1) = F(k)\hat{\underline{x}}(k) + \{\underline{f}[k, \hat{\underline{x}}(k), \underline{u}(k), 0] - F(k)\hat{\underline{x}}(k)\}$$
$$= \underline{f}[k, \hat{\underline{x}}(k), \underline{u}(k), 0]$$

and $\bar{P}(k+1) = F(k)P(k)F^T(k) + \Gamma(k)Q(k)\Gamma^T(k)$

where $F(k) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{k, \hat{\underline{x}}(k), \underline{u}(k), 0}$

$$\Gamma(k) = \left.\frac{\partial \underline{f}}{\partial \underline{v}}\right|_{k, \hat{\underline{x}}(k), \underline{u}(k), 0}$$

③ Compute

$$\underline{\nu}(k+1) = \underline{z}_{EKF}(k+1) - H(k+1)\bar{\underline{x}}(k+1)$$
$$= \underline{z}(k+1) - \underline{h}[k+1, \bar{\underline{x}}(k+1)]$$
$$S(k+1) = H(k+1)\bar{P}(k+1)H^T(k+1) + R(k+1)$$

$$W(k+1) = \bar{P}(k+1) H^T(k+1) S^{-1}(k+1)$$
$$\hat{\underline{x}}(k+1) = \bar{\underline{x}}(k+1) + W(k+1)\underline{\nu}(k+1)$$
$$\& \quad P(k+1) = \bar{P}(k+1) - W(k+1) S(k+1) W^T(k+1)$$

where $H(k+1) = \frac{\partial h}{\partial \underline{x}}\Big|_{k+1,\, \bar{\underline{x}}(k+1)}$

(4) If $k+1 = N$, then stop. Otherwise, replace k by $k+1$ and go to Step (2)

Thus, the first part of the extended smoother is the standard EKF

In all of the covariance smoother calculations from lecture $\underline{z}(k+1)$ does not appear explicitly, and $G(k)\underline{u}(k)$ only appears in the inverse dynamics propagation that determines $\underline{x}(k)$ as a function of $\underline{x}(k+1)$, $\underline{u}(k)$, and $\underline{v}(k)$. This backwards propagation can be replaced by using the function $\underline{F}^{-1}[k, \underline{x}(k+1), \underline{u}(k), \underline{v}(k)]$.

Therefore, the first extended smoother calculations take the form:

(A.) Set $\underline{x}^*(N) = \hat{\underline{x}}(N)$ & $P^*(N) = P(N)$ from the EKF. Also, set $k = N-1$.

(B.) Compute

$$\underline{v}^*(k) = Q(k)\,\Gamma^T(k)\,\bar{P}^{-1}(k+1)\left[\underline{x}^*(k+1) - \bar{\underline{x}}(k+1)\right]$$

& $\underline{x}^*(k) = \underline{f}^{-1}\left[k, \underline{x}^*(k+1), \underline{u}(k), \underline{v}^*(k)\right]$

(C.) Compute

$$P^*(k) = P(k) - P(k)F^T(k)\bar{P}^{-1}(k+1)\left[\bar{P}(k+1) - \ldots\right.$$
$$\left. P^*(k+1)\right]\bar{P}^{-1}(k+1)F(k)P(k)$$

(D.) If $k=0$ Stop. Otherwise replace k by $k-1$ and go to Step (B).

Note that the values of $\bar{\underline{x}}(k+1)$, $\bar{P}(k+1)$, $F(k)$, $\Gamma(k)$, & $P(k)$ must have been retained from the EKF forward pass.

The alternate extended smoother that does not use $f^{-1}[k, \underline{x}(k+1), \underline{u}(k), \underline{v}(k)]$ replaces Step B above with the calculation

Alt. (B) Compute

$$\underline{x}^*(k) = \hat{\underline{x}}(k) + P(k)F^T(k)\bar{P}^{-1}(k+1)\left[\underline{x}^*(k+1) - \bar{\underline{x}}(k+1)\right]$$

as in a standard linear smoother, where $\hat{\underline{x}}(k)$ is retained from the EKF forward pass.

<u>In Summary</u>: The only changes to go from a linear smoother to an extended smoother for a nonlinear problem are

I) Use an EKF for the forward filtering pass rather than a standard linear filter

II) If the backwards propagation to determine $\underline{x}^*(k)$ is to be the one based on calculation of $\underline{v}^*(k)$ and inverse dynamics, then replace the linear inverse dynamics

$$\underline{x}^*(k) = F^{-1}(k)\left[\underline{x}^*(k+1) - G(k)\underline{u}(k) - \Gamma(k)\underline{v}^*(k)\right]$$

with

$$\underline{x}^*(k) = \underline{f}^{-1}\left[k, \underline{x}^*(k+1), \underline{u}(k), \underline{v}^*(k)\right]$$

Note, one might be tempted simply to replace $G(k)\underline{u}(k)$ by $\underline{f}[k, \hat{\underline{x}}(k), \underline{u}(k), 0] - F(k)\hat{\underline{x}}(k)$ in the linearized inverse dynamics recursion. This would yield the backwards smoothed state propagation :

$$\underline{x}^*(k) = F^{-1}(k)\Big\{\underline{x}^*(k+1) - \underline{f}\big[k, \underline{\bar{x}}(k), \underline{u}(k), \underline{0}\big]$$
$$+ F(k)\,\underline{\bar{x}}(k) - \Gamma(k)\,\underline{v}^*(k)\Big\}$$

or

$$\underline{x}^*(k) = \underline{\hat{x}}(k) + F^{-1}(k)\Big\{\underline{x}^*(k+1) - \underline{\bar{x}}(k+1) - \Gamma(k)\,\underline{v}^*(k)\Big\}$$

This would also be a reasonable extended smoother, but it would give up any possible advantage from using the truly nonlinear backwards dynamic propagation in $\underline{f}^{-1}\big[k, \underline{x}(k+1), \underline{u}(k), \underline{v}(k)\big]$.

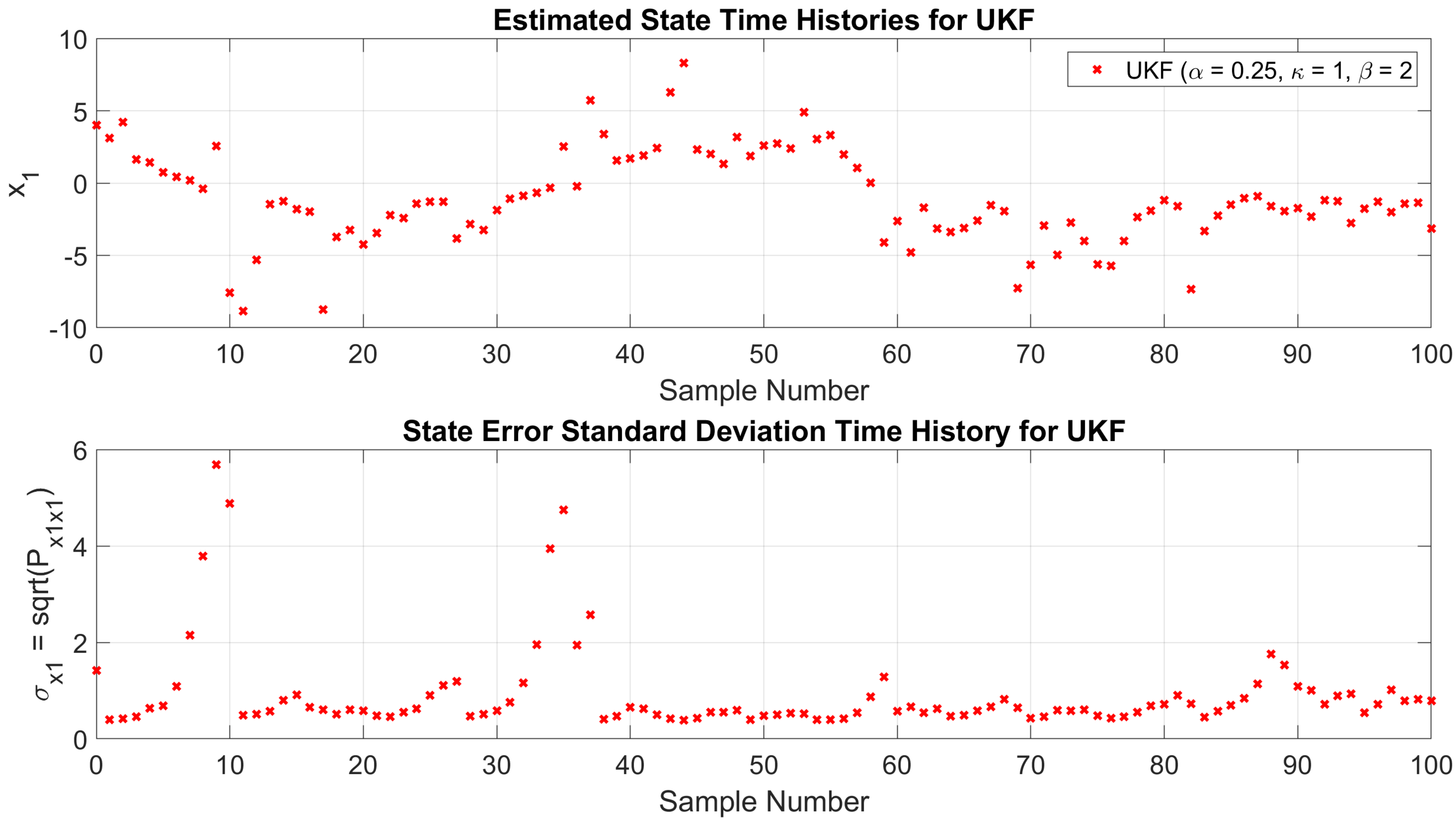
5) Problem #5 of Problem Set 8 [15 pts] Sheet 13 of 29


```
% batch_ukf_ps8prob5.m
%
%  Copyright (c) 2015 Mark L. Psiaki.  All rights reserved.
%
%
%  This Matlab script uses the function ukf_1step_ps8prob5.m to do
%  Unscented Kalman Filtering for the example problem
%  defined by the dynamics model function ffunct_ps8prob5.m, the
%  measurement model function hfunct_ps8prob5.m, and the data
%  in measdata_pfexample.mat to solve Problem 5 of
%  Problem Set 8, except with a modified alpha
%  value as prescribed for use in the MAE 6760
%  Fall 2015 final exam.
%


%
%  Clear the Matlab workspace.
%
   clear
%
%  Set up the problem function handles
%
%  Define the dynamics function.
%
   fmodel_fnct = @(xk_argdum,vk_argdum,k_argdum) ...
             ffunct_ps8prob5(xk_argdum,vk_argdum,k_argdum);
%
%  Define the measurement model function.
%
   hmodel_fnct = @(xkp1_argdum,kp1_argdum) ...
             hfunct_ps8prob5(xkp1_argdum,kp1_argdum);
%
%  Load the data to be filtered.
%
   load measdata_pfexample
%
%  Set the UKF tuning parameters.
%
   kappafltr = 1;
%  alphafltr = 1;
   alphafltr = 0.25; % Change specified for Fall 2015 final exam
   betafltr = 2;
%
%  Determine the number of samples and the number of states
%  and set up arrays to store results.
%
   nx = size(xhat0,1);
   K = size(zkhist,1);
   Kp1 = K + 1;
   xhatkhist = zeros(Kp1,nx);
   Pkhist = zeros(nx,nx,Kp1);
```

```
%
% Store the initial estimate and its error covariance.
%
  xhatkhist(1,:) = xhat0';
  Pkhist(:,:,1) = P0;
%
% Initialize the state estimate and its covariance for
% use in the filter iterations.
%
  xhatkp1 = xhat0;
  Pkp1 = P0;
%
% This is the main loop that executes the UKF filtering
% calculations.
%
  for k = 0:(K-1)
     xhatk = xhatkp1;
     Pk = Pkp1;
     kp1 = k + 1;
     zkp1 = zkhist(kp1,:)';
     Qk = Q;
     Rkp1 = R;
     [xhatkp1,Pkp1] = ...
               ukf_1step_ps8prob5(xhatk,Pk,k,Qk,zkp1,Rkp1,fmodel_fnct,...
                                  hmodel_fnct,kappafltr,alphafltr,...
                                  betafltr);
     kp2 = kp1 + 1;
     xhatkhist(kp2,:) = xhatkp1';
     Pkhist(:,:,kp2) = Pkp1;
  end
%
% Compare results with Particle Filter results for Problem 3
% from file pf_ps8prob3.mat and with Extended Kalman Filter
% results for Problem 4 from file ekf_ps8prob4.mat.
%
  pfresults_structure = load('pf_ps8prob3.mat');
  ekfresults_structure = load('ekf_ps8prob4.mat');
%
% Plot all three results.
%
  subplot(211)
  plot((0:K)',pfresults_structure.xhatkhist,'b*')
  hold on
  set(get(gcf,'CurrentAxes'),'FontSize',16)
  plot((0:K)',ekfresults_structure.xhatkhist,'gx')
  plot((0:K)',xhatkhist,'r.')
  xlabel('Sample Number')
  ylabel('x_1')
  title('Estimated State Time Histories for 3 Nonlinear Filters')
  legend('PF (N_s = 400)',...
         'EKF',...
```

```
        ['UKF (\alpha = ',num2str(alphafltr),...
              ', \kappa = ',num2str(kappafltr),...
              ', \beta = ',num2str(betafltr)]);
   grid
   hold off
   subplot(212)
   sigmadumvec = sqrt(pfresults_structure.Pkhist(:));
   semilogy((0:K)',sigmadumvec,'b*')
   hold on
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   sigmadumvec = sqrt(ekfresults_structure.Pkhist(:));
   semilogy((0:K)',sigmadumvec,'gx')
   sigmadumvec = sqrt(Pkhist(:));
   semilogy((0:K)',sigmadumvec,'r.')
   xlabel('Sample Number')
   ylabel('\sigma_x_1 = sqrt(P_x_1_x_1)')
   title(['State Error Standard Deviation Time Histories',...
        ' for 3 Nonlinear Filters'])
   legend('PF (N_s = 400)',...
        'EKF',...
        ['UKF (\alpha = ',num2str(alphafltr),...
              ', \kappa = ',num2str(kappafltr),...
              ', \beta = ',num2str(betafltr)]);
   ylim([0.0003 10])
   grid
   hold off
%
%  Give the final numerical values for the state and covariance
%  for each of the three filters.
%
   format long
   xhatf = xhatkhist((K-1):Kp1,1)

   Pf = Pkhist(1,1,(K-1):Kp1)

%
%  These final results displayed as follows:
%
%  xhatf(98)  = -1.415210043838208
%  xhatf(99)  = -1.374137131281926
%  xhatf(100) = -3.161050050730308
%
%  Pf(98)  = 0.624012825121864
%  Pf(99)  = 0.664564512091797
%  Pf(100) = 0.629135189744099
%
%  The 3 filter's final state estimates are fairly close,
%  but their final covariances vary considerably.
%
```

```
%
% Clear unneeded temporary quantities.
%
  clear Kp1 xhatkp1 Pkp1 k xhatk Pk kp1 zkp1 Qk Rkp1 kp2 ...
        pfresults_structure ekfresults_structure sigmadumvec
%
% Save the results.
%
  text_commands = ['These data have been created by the',...
                   ' code in batch_ukf_ps8prob5.m.'];
  save ukf_ps8prob5
```



Estimated State Time Histories for UKF
UKF (α = 0.25, κ = 1, β = 2
x_1
10
5
0
-5
-10
0
10
20
30
40
50
60
70
80
90
100
Sample Number
State Error Standard Deviation Time History for UKF
σ_{x1} = sqrt(P_{x1x1})
6
4
2
0
0
10
20
30
40
50
60
70
80
90
100
Sample Number

Sheet 18 of 29


```
function [xhatkp1,Pkp1] = ...
                 ukf_1step_ps8prob5(xhatk,Pk,k,Qk,zkp1,Rkp1,fmodel_fnct,...
                                   hmodel_fnct,kappafltr,alphafltr,...
                                   betafltr)
%
%  Copyright (c) 2015 Mark L. Psiaki.  All rights reserved.
%
%
%  This function executes 1 iteration of an Unscented
%  Kalman Filter (UKF).
%
%  This function uses the UKF sigma-points step size and
%  weighting tuning parameters
%  kappafltr, alphafltr, and betafltr from the paper:
%
%    Wan, E.A., and van der Merwe, R., "The Unscented Kalman Filter,"
%    in Kalman Filtering and Neural Networks, S. Haykin, ed.,
%    Wiley, (New York, 2001), Chapter 7, pp. 221-280.
%
%  Inputs:
%
%    xhatk               The nx-by-1 vector that constitutes the UKF's
%                        a posteriori filtered state estimate at
%                        sample k.
%
%    Pk                  The nx-by-nx matrix that constitutes the UKF's
%                        a posteriori filtered state estimation error
%                        covariance at sample k.
%
%    k                   The scalar sample index.
%
%    Qk                  The nv-by-nv matrix that constitutes the UKF's
%                        a priori process noise covariance at sample k.
%                        The process noise mean is assumed to be zero.
%
%    zkp1                The nz-by-1 vector of measurements at
%                        sample kp1 = k + 1.  These will be used
%                        to do the state update.
%
%    Rkp1                The nz-by-nz measurement error covariance
%                        matrix at sample kp1 = k + 1.
%
%    fmodel_fnct         The function handle of the Matlab function that
%                        evaluates the discrete-time dynamics model
%                        function in the nonlinear difference equation
%                        xkp1 = f(xk,vk,k).  The Matlab command
%
%                           fk = fmodel_fnct(xk,vk,k)
%
%                        implements the nonlinear dynamics model for
%                        the state transition from sample k to sample
```

```
%                      kp1 = k + 1.
%
%    hmodel_fnct       The function handle of the Matlab function that
%                      evaluates the discrete-time measurement model
%                      function in the nonlinear measurement equation
%                      zkp1 = h(xkp1,kp1) + wkp1.  The Matlab command
%
%                         hkp1 = hmodel_fnct(xkp1,kp1)
%
%                      implements the nonlinear measurement at sample
%                      kp1 = k + 1.
%
%    kappafltr         The scalar kappa value that gets used to design
%                      sigma point distributions and sigma-point
%                      weightings in the UKF calculations.
%
%    alphafltr         The scalar alpha value that gets used to design
%                      sigma point distributions and sigma-point
%                      weightings in the UKF calculations.  This is
%                      normally a small positive number.
%
%    betafltr          The scalar beta value that gets used to design
%                      sigma point distributions and sigma-point
%                      weightings in the UKF calculations.  This
%                      is normally a positive number.  The paper
%                      by Wan and van der Merwe says that the value 2
%                      is optimal for Gaussian distributions.
%
%  Outputs:
%
%    xhatkp1           The nx-by-1 vector that constitutes the UKF's
%                      a posteriori filtered state estimate at sample
%                      kp1 = k + 1.
%
%    Pkp1              The nx-by-nx matrix that constitutes the UKF's
%                      a posteriori filtered state estimation error
%                      covariance at sample kp1 = k + 1.
%


%
%  Determine various dimensions and set up
%  inputs for calls to the square-root unscented filter calculations.
%
   nx = size(xhatk,1);
   nv = size(Qk,1);
   nz = size(zkp1,1);
   kp1 = k + 1;
   Sxxhatk = chol(Pk)';
   Svvk = chol(Qk)';
%
%  Compute the lambdafltr scalar that is part of the sigma
```

```
%  points length and weighting calculations.
%
   lambdafltr = (alphafltr^2)*(nx + nv + kappafltr) - (nx + nv);
%
%  Calculate the sigma points and their weightings, and do the
%  the propagation and mean calculations.
%
   Nsigma = 1 + 2*(nx + nv);
   vhatk = zeros(nv,1);
   xkp1sigmamat = zeros(nx,Nsigma);
   zkp1sigmamat = zeros(nz,Nsigma);
   Wfac = 1/(nx + nv + lambdafltr);
   Wvec = [lambdafltr;(ones((Nsigma-1),1)*0.5)]*Wfac;
   sigmafac = sqrt(nx + nv + lambdafltr);
   xbarkp1 = zeros(nx,1);
   zbarkp1 = zeros(nz,1);
   for jj = 1:Nsigma
      xdum = xhatk;
      vdum = vhatk;
      if jj > 1
         if jj <= (1+nx)
            xdum = xdum + sigmafac*Sxxhatk(:,(jj-1));
         elseif jj <= (1+2*nx)
            xdum = xdum - sigmafac*Sxxhatk(:,(jj-(nx+1)));
         elseif jj <= (1+2*nx+nv)
            vdum = vdum + sigmafac*Svvk(:,(jj-(2*nx+1)));
         else
            vdum = vdum - sigmafac*Svvk(:,(jj-(2*nx+nv+1)));
         end
      end
      xkp1_jj = fmodel_fnct(xdum,vdum,k);
      zkp1_jj = hmodel_fnct(xkp1_jj,kp1);
      W_jj = Wvec(jj,1);
      xbarkp1 = xbarkp1 + xkp1_jj*W_jj;
      zbarkp1 = zbarkp1 + zkp1_jj*W_jj;
      xkp1sigmamat(:,jj) = xkp1_jj;
      zkp1sigmamat(:,jj) = zkp1_jj;
   end
%
%  Modify Wvec(1,1) to be the proper weighting for the covariance
%  calculations.
%
   Wvec(1,1) = Wvec(1,1) + 1 + betafltr - alphafltr^2;
%
%  Compute the components of the covariance matrix of
%  [xbarkp1;zbarkp1]
%
   Pbarkp1 = zeros(nx,nx);
   Pxzkp1 = zeros(nx,nz);
   Pzzkp1 = zeros(nz,nz);
   for jj = 1:Nsigma
```

```
      dxjj = xkp1sigmamat(:,jj) - xbarkp1;
      dzjj = zkp1sigmamat(:,jj) - zbarkp1;
      W_jj = Wvec(jj,1);
      Pbarkp1 = Pbarkp1 + W_jj*(dxjj*(dxjj'));
      Pxzkp1 = Pxzkp1 + W_jj*(dxjj*(dzjj'));
      Pzzkp1 = Pzzkp1 + W_jj*(dzjj*(dzjj'));
   end
   Pzzkp1 = Pzzkp1 + Rkp1;
%
%  Complete the Kalman Filter Update.
%
   Wkp1 = Pxzkp1/Pzzkp1;
   nukp1 = zkp1 - zbarkp1;
   xhatkp1 = xbarkp1 + Wkp1*nukp1;
   Pkp1 = Pbarkp1 - Wkp1*(Pxzkp1');
```

Sheet 22 of 29


```
function [fk,dfkdxk,dfkdvk] = ffunct_ps8prob5(xk,vk,k)
%
%  Copyright (c) 2015 Mark L. Psiaki.  All rights reserved.
%
%
%  This function models the dynamics of the first filtering example
%  for MAE 6760.  The measurement model is
%
%      xkp1 = f(xk,vk,k) = 2*atan(xk) + 0.5*cos(pi*k/3) + vk.
%
%  This function is for use in testing the particle filter and related
%  Matlab functions.
%
%  Inputs:
%
%    xk          The 1-by-1 state vector of this system at time tk =
%                k samples.
%
%    vk          The 1-by-1 process noise vector of this system that
%                operates from time tk = to time tkp1.
%
%    k           The index of the current sample time.
%
%  Outputs:
%
%    fk          The 1-by-1 dynamics function that gives xkp1 in the
%                difference equation model.  Its elements have the
%                same units as the corresponding elements of xk.
%
%    dfkdxk      The 1-by-1 partial derivative of fk with respect to
%                xk.  This is an empty array on output if iflagfonly = 1.
%
%    dfkdvk      The 1-by-1 partial derivative of fk with respect to
%                vk.  This is an empty array on output if iflagfonly = 1.
%


%
%  Compute fk.
%
   atanxk = atan(xk);
   fk = 2*atanxk + 0.5*cos(pi*k/3) + vk;
%
%  Compute the two required Jacobian matrices.
%
   dfkdxk = 2/(1 + xk^2);
   dfkdvk = 1;
```

```
function [hk,dhkdxk] = hfunct_ps8prob5(xk,k)
%
%  Copyright (c) 2015 Mark L. Psiaki.  All rights reserved.
%
%
%  This function models the measurements of the first filtering example
%  for MAE 6760.  The measurement model is
%
%      zk = h(xk,k) + wk = xk + xk^2 + xk^3 + wk
%
%  This function is for use in testing the particle filter and related
%  Matlab functions.
%
%  Inputs:
%
%    xk          The 1-by-1 state vector of this system at time tk =
%                k samples.
%
%    k           The index of the current sample time.
%
%  Outputs:
%
%    hk          The 1-by-1 measurement function that gives zk.
%
%    dhkdxk      The 1-by-1 partial derivative of hk with respect to
%                xk.  This is an empty array on output if iflaghonly = 1.
%

%
%  Compute hk.  Note: these measurements are completely arbitrary.
%
   hk = xk + xk^2 + xk^3;
%
%  Compute the required Jacobian matrix.
%
   dhkdxk = 1 + 2*xk + 3*(xk^2);
```

6) Problem #8 of Problem Set 8 [20pts] Sheet 24 of 29

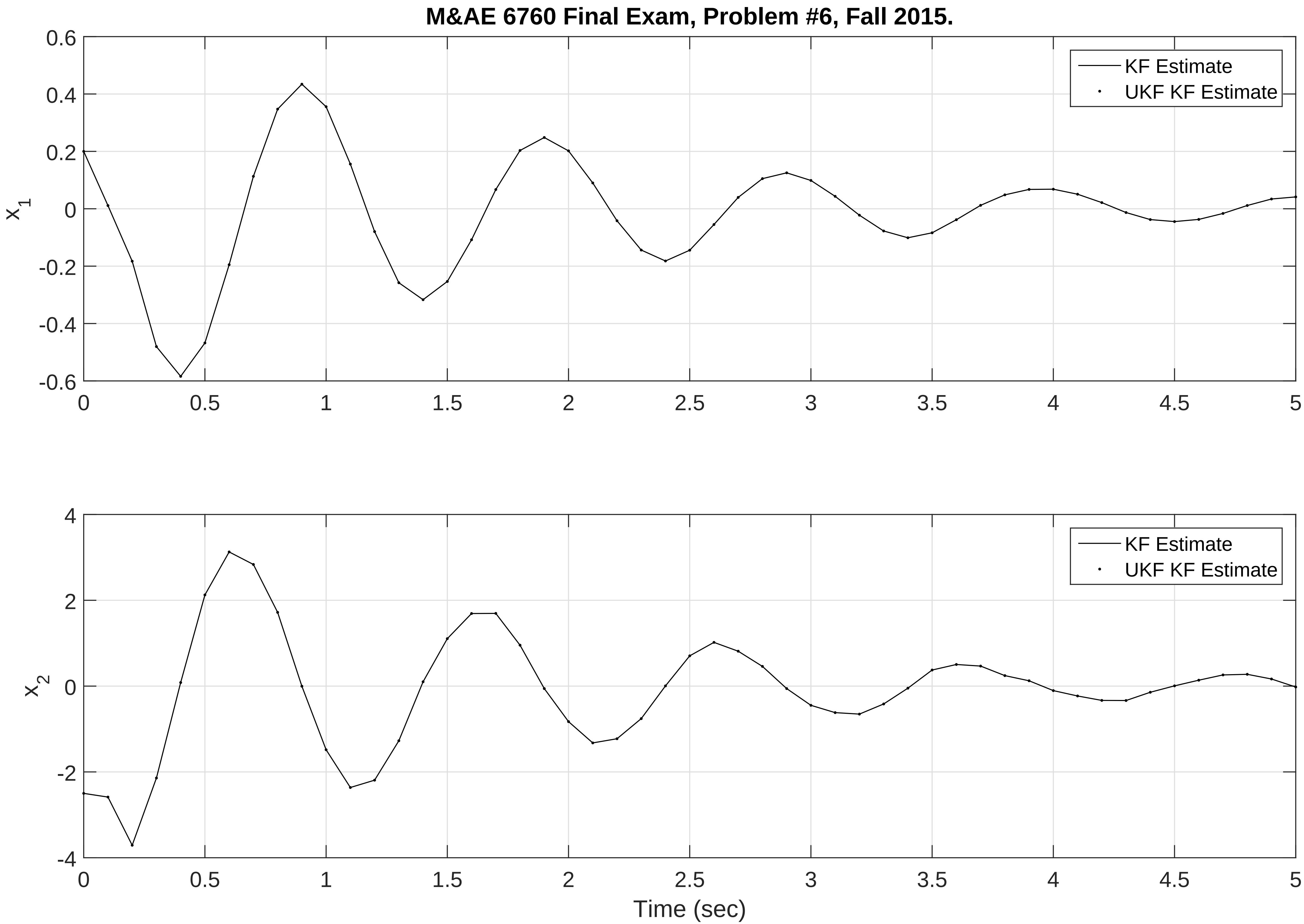

```
% final_f15_pr08ukf.m
%
% This Matlab script solves the unscented Kalman filter case for
% the 7th problem of the final exam for M&AE 6760 during
% the Fall semester of 2015.  This is a modified
% version of Problem 3 of Assignment #5 that uses the
% data in kf_example02b.m in order define the problem.
%


%
% Clear the Matlab workspace and load the problem matrices and vectors.
%
  clear
  kf_example02b;
%
% This change makes for cleaner notation, even though it
% doesn't matter due to the time-invariant nature of
% the problem.
%
  Hkp1 = Hk;
  Rkp1 = Rk;
  clear Hk Rk
%
% Set up output arrays.  Note that row k+1 of xhathist
% corresponds to sample k, and Phist(:,:,k+1) corresponds
% to sample k.
%
  nx = size(xhat0,1);
  xhathist = zeros(51,nx);
  Phist = zeros(nx,nx,51);
%
% Initialize the filter state and covariance and store the
% results for k = 0.
%
  k = 0;
  xhatk = xhat0;
  Pk = P0;
  kp1 = k + 1;
  xhathist(kp1,:) = xhatk';
  Phist(:,:,kp1) = Pk;
%
% Initialize some parameters that are needed by the UKF
% calculations to size its sigma points spread.  Also compute
% its mean and covariance weights.
%
  nv = size(Qk,1);
  nz = size(Rkp1,1);
  kappa = 3 - (nx + nv);
  alpha = 1;  % No need to use a small value for this linear problem.
  lambda = (alpha^2)*(nx + nv + kappa) - (nx + nv);
  beta = 2;
```

```
   twonx = 2*nx;
   twonxp1 = twonx + 1;
   twonxpnv = twonx + nv;
   twonxptwonv = 2*(nx + nv);
   twonxptwonvp1 = twonxptwonv + 1;
   Wmvec = [(lambda/(nx + nv + lambda));...
            (ones(twonxptwonv,1)*(1/(2*(nx + nv + lambda))))];
   Wcvec = Wmvec;
   Wcvec(1,1) = Wmvec(1,1) + 1 - (alpha^2) + beta;
%
%  Compute the sigma points spread factor and the
%  actual sigma perturbation matrix for the process noise.
%
   facss = sqrt(nx + nv + lambda);
   Svk = chol(Qk)';
   facss_Svk = facss*Svk;
%
%  Iterate the filter for the 50 sample times.
%
   for k = 0:49
%
%  Generate the sigma points.  First put in their
%  nominal values.  Then perturb those that need
%  perturbing.
%
      xksigmapts = xhatk*ones(1,twonxptwonvp1);
      vksigmapts = zeros(nv,twonxptwonvp1);
      Sxk = chol(Pk)';
      facss_Sxk = facss*Sxk;
      for ii = 1:nx
         iip1 = ii + 1;
         iipnxp1 = iip1 + nx;
         xksigmapts(:,iip1) = xhatk + facss_Sxk(:,ii);
         xksigmapts(:,iipnxp1) = xhatk - facss_Sxk(:,ii);
      end
      for ii = twonxp1:twonxpnv
         iip1 = ii + 1;
         iipnvp1 = iip1 + nv;
         iim2nx = ii - twonx;
         vksigmapts(:,iip1) = facss_Svk(:,iim2nx);
         vksigmapts(:,iipnvp1) = - facss_Svk(:,iim2nx);
      end
      clear ii iip1 iipnxp1 iipnvp1 iim2nx Sxk facss_Sxk
%
%  Propagate the sigma points through the dynamics and
%  through tme measurement models.
%
      xbarkp1sigmapts = zeros(nx,twonxptwonvp1);
      zkp1sigmapts = zeros(nz,twonxptwonvp1);
      for iip1 = 1:twonxptwonvp1
         xbarkp1sigmaptii = Fk*xksigmapts(:,iip1) + ...
```

```
                            Gammak*vksigmapts(:,iip1);
         xbarkp1sigmapts(:,iip1) = xbarkp1sigmaptii;
         zkp1sigmapts(:,iip1) = Hkp1*xbarkp1sigmaptii;
      end
      clear iip1 xbarkp1sigmaptii xksigmapts vksigmapts
%
%  Compute the a priori state and measurment at sample k+1.
%  This matrix-vector multiplication is equivalent to
%  the usual weighted sum of the UKF, but is probably
%  executes more rapidly than would a for loop.
%
      xbarkp1 = xbarkp1sigmapts*Wmvec;
      zbarkp1 = zkp1sigmapts*Wmvec;
%
%  Compute the a priori covariances of xbarkp1 and zbarkp1
%  along with their cross correlation.
%
      dxbarkp1sigmapts = xbarkp1sigmapts - xbarkp1*ones(1,twonxptwonvp1);
      dzkp1sigmapts = zkp1sigmapts - zbarkp1*ones(1,twonxptwonvp1);
      Pbarkp1 = zeros(nx,nx);
      Pbarxzkp1 = zeros(nx,nz);
      Pbarzzkp1 = Rkp1;
      for iip1 = 1:twonxptwonvp1
         dxbarkp1sigmaptii = dxbarkp1sigmapts(:,iip1);
         dzkp1sigmaptii = dzkp1sigmapts(:,iip1);
         Wcii = Wcvec(iip1,1);
         Pbarkp1 = Pbarkp1 + ...
                      Wcii*(dxbarkp1sigmaptii*(dxbarkp1sigmaptii'));
         Pbarxzkp1 = Pbarxzkp1 + ...
                      Wcii*(dxbarkp1sigmaptii*(dzkp1sigmaptii'));
         Pbarzzkp1 = Pbarzzkp1 + ...
                      Wcii*(dzkp1sigmaptii*(dzkp1sigmaptii'));
      end
      clear dxbarkp1sigmapts dzkp1sigmapts iip1 dxbarkp1sigmaptii ...
          dzkp1sigmaptii Wcii xbarkp1sigmapts zkp1sigmapts
%
%  Measurement update.
%
      Pbarzzkp1inv = inv(Pbarzzkp1);
      kp1 = k + 1;
      nukp1 = zhist(kp1,:)' - zbarkp1;
      Wkp1 = Pbarxzkp1*Pbarzzkp1inv;
      xhatkp1 = xbarkp1 + Wkp1*nukp1;
      Pkp1 = Pbarkp1 - Pbarxzkp1*Pbarzzkp1inv*(Pbarxzkp1');
%
%  Storage of results.
%
      kp2 = kp1 + 1;
      xhathist(kp2,:) = xhatkp1';
      Phist(:,:,kp2) = Pkp1;
%
```

```
%  Prepare for the next iteration.
%
      xhatk = xhatkp1;
      Pk = Pkp1;
   end
   clear xhatk Pk k kp1 kp2 twonx twonxp1 twonxpnv twonxptwonv ...
         twonxptwonvp1 facss Svk facss_Svk xbarkp1 zbarkp1 ...
         Pbarkp1 Pbarxzkp1 Pbarzzkp1 Pbarzzkp1inv nukp1 Wkp1 ...
         xhatkp1 Pkp1
%
%  Save the results.
%
   save final_f15_pr08_ukfsoln
%
%  Plot results along with results for the Kalman filter,
%  which are stored in the file final_f15_pr08_kfsoln.mat.
%  The data in that latter file will be loaded as a
%  Matlab structure.
%
   kfresults_struct = load('final_f15_pr08_kfsoln.mat');
   tplothist = [0;thist];
   subplot(211);
   plot(tplothist,kfresults_struct.xhathist(:,1),'k-',...
        tplothist,xhathist(:,1),'k.')
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   legend('KF Estimate','UKF KF Estimate')
   grid
   ylabel('x_1')
   title('M&AE 6760 Final Exam, Problem #6, Fall 2015.')
   subplot(212);
   plot(tplothist,kfresults_struct.xhathist(:,2),'k-',...
        tplothist,xhathist(:,2),'k.')
   set(get(gcf,'CurrentAxes'),'FontSize',16)
   legend('KF Estimate','UKF KF Estimate')
   grid
   ylabel('x_2')
   xlabel('Time (sec)')
%
%  Display final state estimate and covariance.
%
   format long
   xhat50_ukf = xhathist(51,:)'
   xhat50_kf = kfresults_struct.xhathist(51,:)'
   P50_ukf = Phist(:,:,51)
   P50_kf = kfresults_struct.Phist(:,:,51)
%
%  Results are:
%
% xhat50_ukf =  [  0.041293309029757;...
%                 -0.020772665133846]
%
```

```
% xhat50_kf =    [  0.041293309029757;...
%                  -0.020772665133846]
%
% P50_ukf =  1.0e-003*[ 0.006371942787185,  0.000930691200954;...
%                       0.000930691200954,  0.813329461749668]
%
% P50_kf =   1.0e-003*[ 0.006371942787187,  0.000930691200940;...
%                       0.000930691200968,  0.813329461749557]
%
%  These results agree to more than 10 significant digits, as
%  expected.  Any differences are probably the results of
%  round-off error.  If a smaller alpha had been used, then
%  the relative effects of round-off error probably would have
%  been larger.
```


M&AE 6760 Final Exam, Problem #6, Fall 2015.
KF Estimate
UKF KF Estimate
x_1
x_2
Time (sec)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
4
2
-2
-4
0.5
1
1.5
2.5
3
3.5
4.5
5