ASE 381P Problem Set #7

Posting Date: November 21, 2019

You need not hand in anything. Instead, be prepared to answer any of these problems on an upcoming take-home exam. You may discuss your solutions with classmates up until the time that the exam becomes available, but do not swap work.

1. The orbital dynamics of a spacecraft in orbit around a spherical Earth of uniform density can be modeled as

$$\begin{bmatrix} \dot{\boldsymbol{r}}_s \\ \dot{\boldsymbol{v}}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_s \\ \frac{-\mu \boldsymbol{r}_s}{||\boldsymbol{r}_s||^3} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ I \end{bmatrix} \tilde{\boldsymbol{u}}(t) + \begin{bmatrix} \boldsymbol{0} \\ I \end{bmatrix} \tilde{\boldsymbol{v}}(t)$$

where $\boldsymbol{r}_s$ and $\boldsymbol{v}_s$, are the spacecraft's Earth-Centered-Inertial (ECI) position and velocity vectors (both 3×1), $\mu = G * M_\oplus$ is the Earth's gravitational constant, $\boldsymbol{u}(t)$ is a vector of control accelerations, and $\tilde{\boldsymbol{v}}(t)$ is a vector of disturbance accelerations. The dynamics equations can be rewritten more compactly as

$$\dot{\tilde{\boldsymbol{x}}}(t) = \tilde{\boldsymbol{f}}[t, \tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{u}}(t)] + D(t)\tilde{\boldsymbol{v}}(t)$$

Assume that $\tilde{\boldsymbol{u}}(t) = \boldsymbol{u}(k)$ and $\tilde{\boldsymbol{v}}(t) = \boldsymbol{v}(k)$ on the interval $t_k \leq t < t_{k+1}$ (this is the zero-order-hold assumption introduced in lecture). Write a numerical integration function in Matlab that computes the right-hand side of the corresponding discrete-time model

$$\boldsymbol{x}(k+1) = \boldsymbol{f}[k, \boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{v}(k)]$$

and also computes the 6×6 linearized state transition matrix

$$F(k) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}(k)}\right]$$

and the 6×3 linearized disturbance matrix

$$\Gamma(k) = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}(k)}\right]$$

Use at least 4th-order Runge-Kutta integration. You may write the Runge-Kutta algorithm yourself or you may use a Matlab solver (`ode45` and `ode113` are good choices). You may either write your code from scratch or fill in the missing code wherever the symbol `????` appears in `propagateOrbit_temp.m` and `Afun_temp.m`. Remove the `_temp` in the filenames before using these functions.

Test your code in the following way. Assume a circular orbit of altitude 780 km and a spherical Earth radius of 6378 km. Let $\mu = 3.986005 \times 10^{14}$. Pick an arbitrarily oriented initial position vector and an initial velocity vector that is perpendicular to the initial position vector. Starting at this initial state, integrate over one complete orbit assuming

that control and process accelerations are zero. Look at the magnitude of the difference between the initial and final positions. The analytic solution would predict zero difference. Repeat your experiment with a different integration step size (or a different `RelTol` value if using a Matlab ODE solver).

To validate the $F(k)$ matrix, again assume zero control and disturbance accelerations. Use your integration function to propagate from an arbitrary initial $\boldsymbol{x}(k)$ to $\boldsymbol{x}(k+1)$. Store the $F(k)$ matrix generated from this integration. Now pick some $\bar{\boldsymbol{x}}(k)$ that is close to $\boldsymbol{x}(k)$ and use you integration routine to propagate from $\bar{\boldsymbol{x}}(k)$ to $\bar{\boldsymbol{x}}(k+1)$. If your $F(k)$ matrix is correct, then $[\boldsymbol{x}(k+1) - \bar{\boldsymbol{x}}(k+1)] \approx F(k)[\boldsymbol{x}(k) - \bar{\boldsymbol{x}}(k)]$ for short update intervals $t_{k+1} - t_k$.

Be prepared to apply your integration function to an arbitrary $\boldsymbol{x}(0)$ and arbitrary time histories for $\boldsymbol{u}(k)$ and $\boldsymbol{v}(k)$.

2. Suppose that you are given a nonlinear discrete-time dynamics model of the form:

$$\boldsymbol{x}(k+1) = \boldsymbol{f}[k, \boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{v}(k)]$$

Suppose, also, that this dynamics function has a functional inverse:

$$g[k, \boldsymbol{x}(k+1), \boldsymbol{u}(k), \boldsymbol{v}(k)] = \boldsymbol{f}^{-1}[k, \boldsymbol{x}(k+1), \boldsymbol{u}(k), \boldsymbol{v}(k)]$$

where the inverse is defined such that

$$\boldsymbol{g}\{k, \boldsymbol{f}[k, \boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{v}(k)], \boldsymbol{u}(k), \boldsymbol{v}(k)\} = \boldsymbol{x}(k)$$

Prove that

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}(k+1)} = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}(k)}\right]^{-1}$$

and

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{v}(k)} = -\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}(k)}\right]^{-1} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}(k)}$$

where each partial derivative is evaluated at appropriate values of the original functions' input arguments.

Hints: Differentiate the equation that defines the inverse. Differentiate it with respect to $\boldsymbol{x}(k)$ or with respect to $\boldsymbol{v}(k)$, depending on which relationship you are proving. Make sure that you apply the chain rule correctly.

3. Derive an SRIF version of the extended Kalman filter. It should be equivalent to the normal covariance-based extended Kalman filter, but it should keep track of its state and covariance by using components of an information equation, as with the SRIF for linear problems.

Hints: The SRIF derivation involves use of inverse dynamics to eliminate $\boldsymbol{x}(k)$ from its information equation. Afterwards, the equations get manipulated using QR factorizations.

Use the inverse of the linearized dynamics equation to eliminate $\boldsymbol{x}(k)$ in the extended SRIF. Also, linearize the measurement equation about an appropriate value of $\boldsymbol{x}(k+1)$ before you incorporate it into the usual SRIF calculations. Don't forget to re-normalize the measurement equation on each iteration so that its re-normalized measurement error covariance matrix equals the identity.

4. Derive an SRIF version of the iterated extended Kalman filter. It should be equivalent to the normal covariance-based iterated extended Kalman filter, but it should keep track of its state and covariance by using components of an information equation, as with the SRIF for linear problems.

   Hints: The iterated filter re-linearizes its measurement equation about new estimates of $\boldsymbol{x}(k+1)$. This can be done in the context of the SRIF's normalized measurement error equation. Certain of the QR-factorization-based calculations will have to be repeated after each re-linearization.

5. Derive a square-root extended nonlinear smoother based on the same principles as the extended Kalman filter. What smoother equations must change relative to the SRIS that was introduced in lecture?

6. Derive a Kalman filter for the following system, which effectively has correlated process noise and measurement noise:

$$\boldsymbol{x}(k+1) = F(k)\boldsymbol{x}(k) + G(k)\boldsymbol{u}(k) + \Gamma(k)\boldsymbol{v}(k)$$

$$\boldsymbol{z}(k) = H(k)\boldsymbol{x}(k) + \Lambda(k)\boldsymbol{v}(k) + \boldsymbol{w}(k)$$

   where $\boldsymbol{v}(k) \sim N[0, Q(k)]$, $\boldsymbol{w}(k) \sim N[0, R(k)]$, $E[\boldsymbol{v}(k)\boldsymbol{w}^T(k)] = 0$, $E[\boldsymbol{v}(k)\boldsymbol{v}^T(j)] = 0$ if $k \neq j$, and $E[\boldsymbol{w}(k)\boldsymbol{w}^T(j)] = 0$ if $k \neq j$. This problem can be solved by using the techniques of Bar-Shalom Section 8.3.1 if one makes appropriate re-definitions of the process noise and the measurement noise, but do not solve the problem in this way. Instead, create augmented vectors

$$\boldsymbol{x}_a(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{v}(k) \end{bmatrix}$$

   and

$$\boldsymbol{x}_a(k+1) = \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{v}(k+1) \end{bmatrix}$$

   and work with expectation values, covariance matrices, and updates for these vectors. Express the expectation values and covariance matrices of these vectors in block form so that it is obvious how to determine each component of the augmented vectors and matrices.

Hints: You will have to derive the measurement update based on the principles that are outlined in Bar-Shalom Section 3.2.1 and that are used in Bar-Shalom Section 5.2.2 in order to derive the measurement update for the normal Kalman filter. In order to do this, you will have to figure out the expectation values of $\boldsymbol{x}_a(k+1)$ and $\boldsymbol{z}(k+1)$ given the data up through time k. You will also have to determine these vectors' error covariances and their cross-correlation. In order to do this, you will need to use the dynamics model, the measurement model, the joint a posteriori state and process noise statistics at time k, the a priori process noise statistics at time $k+1$, and the a priori measurement error statistics at time $k+1$.

Also do the following problems from Bar Shalom: 10-3, 10-10