Optimal Estimation Methods

(Lecture 15 – Unscented Kalman Filtering)

Dr. John L. Crassidis

University at Buffalo – State University of New York
Department of Mechanical & Aerospace Engineering
Amherst, NY 14260-4400
johnc@buffalo.edu
http://www.buffalo.edu/~johnc

University at Buffalo The State University of New York

Estimate Derivation (i)

- Start with the following problem formulation
  - Given a random variable $\mathbf{x}$ and a set of measurements, denoted by $\tilde{\mathbf{Y}}_k = \{\tilde{\mathbf{y}}_0, \tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k\}$, evaluate an estimate of $\mathbf{x}$, i.e.,
$$\hat{\mathbf{x}}_k = \mathbf{f}[\tilde{\mathbf{y}}_i, \quad 0 = 1, 2 \ldots, k] \qquad (1)$$
  - Say we wish to minimize the conditional mean-square error
$$J = E\left\{(\hat{\mathbf{x}}_k - \mathbf{x}_k)^T(\hat{\mathbf{x}}_k - \mathbf{x}_k)|\tilde{\mathbf{Y}}_k\right\}$$
$$= E\left\{\hat{\mathbf{x}}_k^T\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^T\mathbf{x}_k - \mathbf{x}_k^T\hat{\mathbf{x}}_k + \mathbf{x}_k^T\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}$$
  - Need to determine $E\left\{\hat{\mathbf{x}}_k|\tilde{\mathbf{Y}}_k\right\}$ and $E\left\{\hat{\mathbf{x}}_k^T\hat{\mathbf{x}}_k|\tilde{\mathbf{Y}}_k\right\}$
  - Because the estimate is a function of the measurements, from Eq. (1) we have

all measurements

$$E\left\{\hat{\mathbf{x}}_k|\tilde{\mathbf{Y}}_k\right\} = \hat{\mathbf{x}}_k, \quad \text{and} \quad E\left\{\hat{\mathbf{x}}_k^T\hat{\mathbf{x}}_k|\tilde{\mathbf{Y}}_k\right\} = \hat{\mathbf{x}}_k^T\hat{\mathbf{x}}_k$$
  - Then J can be explicitly written out as

$$J = \hat{\mathbf{x}}_k^T\hat{\mathbf{x}}_k - \hat{\mathbf{x}}_k^T E\left\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\} - E\left\{\mathbf{x}_k^T|\tilde{\mathbf{Y}}_k\right\}\hat{\mathbf{x}}_k + E\left\{\mathbf{x}_k^T\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}$$

Estimate Derivation (ii)

- Add and subtract $E\{\mathbf{x}_k^T|\tilde{\mathbf{Y}}_k\}E\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\}$ to give

$$J = \left[\hat{\mathbf{x}}_k - E\left\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}\right]^T \left[\hat{\mathbf{x}}_k - E\left\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}\right]$$
$$\underbrace{- E\{\mathbf{x}_k^T|\tilde{\mathbf{Y}}_k\}E\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\} + E\left\{\mathbf{x}_k^T\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}}$$

- truth dependent only

- Note that the last two terms do not depend on the estimate
- This is clearly minimized when

all measurements up to k.

$$\boxed{\hat{\mathbf{x}}_k = E\left\{\mathbf{x}_k|\tilde{\mathbf{Y}}_k\right\}}$$

- Provides a rigorous approach to understand what the estimate is from a conditional expectation point of view
  - Note that the estimate at the k^{th} time point is conditioned on all the measurements up to that point
  - This is where the "memory" comes into the estimate
  - Obviously seen in the sequential least squares estimator as an example

MAP KF Derivation (i)

- Begin with the following density functions

$$p(\hat{\mathbf{x}}_k^-) = \frac{1}{\left[\det(2\pi P_k^-)\right]^{1/2}} \exp\left[-\frac{1}{2}(\hat{\mathbf{x}}_k^- - \mathbf{x}_k)^T (P_k^-)^{-1}(\hat{\mathbf{x}}_k^- - \mathbf{x}_k)\right]$$

$$p(\tilde{\mathbf{y}}_k|\mathbf{x}_k) = \frac{1}{\left[\det(2\pi R_k)\right]^{1/2}} \exp\left[-\frac{1}{2}(\tilde{\mathbf{y}}_k - H_k\mathbf{x}_k)^T R_k^{-1}(\tilde{\mathbf{y}}_k - H_k\mathbf{x}_k)\right] \quad (1)$$

- Use Bayes' rule on $p(\mathbf{x}_k|\tilde{\mathbf{Y}}_k)$, with $\tilde{\mathbf{Y}}_k = \{\tilde{\mathbf{y}}_0, \tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_k\}$

$$\begin{aligned} p(\mathbf{x}_k|\tilde{\mathbf{Y}}_k) &= \frac{p(\tilde{\mathbf{Y}}_k|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\tilde{\mathbf{Y}}_k)} \\ &= \frac{p(\tilde{\mathbf{y}}_k, \tilde{\mathbf{Y}}_{k-1}|\mathbf{x}_k)p(\mathbf{x}_k)}{p(\tilde{\mathbf{y}}_k, \tilde{\mathbf{Y}}_{k-1})} \end{aligned} \quad (2)$$

- Consider the following conditional relationship

$$p(\mathbf{a}|\mathbf{b}) = \frac{p(\mathbf{a}, \mathbf{b})}{p(\mathbf{b})} \quad \text{or} \quad p(\mathbf{a}, \mathbf{b}) = p(\mathbf{a}|\mathbf{b})p(\mathbf{b}) \quad (3)$$

MAP KF Derivation (ii)

- Define $\mathbf{a} \equiv \tilde{\mathbf{y}}_k, \quad \mathbf{b} \equiv \tilde{\mathbf{Y}}_{k-1}$, then the denominator in Eq. (2) becomes

$$p(\tilde{\mathbf{y}}_k, \tilde{\mathbf{Y}}_{k-1}) = p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}) p(\tilde{\mathbf{Y}}_{k-1}) \tag{4}$$

- Consider the following relationship

$$p(\mathbf{a}, \mathbf{b} \,|\, \mathbf{c}) = \frac{p(\mathbf{a}, \mathbf{b}, \mathbf{c})}{p(\mathbf{c})} \tag{5}$$

- Rewrite Eq. (3) using different notation (replacing $\mathbf{b}$ with $\mathbf{d}$)

$$p(\mathbf{a}, \mathbf{d}) = p(\mathbf{a} | \mathbf{d}) p(\mathbf{d}) \tag{6}$$

- Using Eq. (6) with $\mathbf{d} \equiv [\mathbf{b}^T \quad \mathbf{c}^T]^T$, then Eq. (5) becomes

$$p(\mathbf{a}, \mathbf{b} \,|\, \mathbf{c}) = \frac{p(\mathbf{a}, \mathbf{b}, \mathbf{c})}{p(\mathbf{c})} = \frac{p(\mathbf{a} \,|\, \mathbf{b}, \mathbf{c}) p(\mathbf{b}, \mathbf{c})}{p(\mathbf{c})}$$

- Finally use Eq. (3) on $p(\mathbf{b}, \mathbf{c})$ to give

$$\begin{aligned} p(\mathbf{a}, \mathbf{b} \,|\, \mathbf{c}) &= \frac{p(\mathbf{a} \,|\, \mathbf{b}, \mathbf{c}) p(\mathbf{b}, \mathbf{c})}{p(\mathbf{c})} = \frac{p(\mathbf{a} \,|\, \mathbf{b}, \mathbf{c}) p(\mathbf{b} | \mathbf{c}) p(\mathbf{c})}{p(\mathbf{c})} \\ &= p(\mathbf{a} \,|\, \mathbf{b}, \mathbf{c}) p(\mathbf{b} | \mathbf{c}) \end{aligned}$$

MAP KF Derivation (iii)

- Define $\mathbf{a} \equiv \tilde{\mathbf{y}}_k, \;\; \mathbf{b} \equiv \tilde{\mathbf{Y}}_{k-1}, \;\; \mathbf{c} \equiv \mathbf{x}_k$, then

$$p(\tilde{\mathbf{y}}_k, \tilde{\mathbf{Y}}_{k-1} | \mathbf{x}_k) = p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}, \mathbf{x}_k) p(\tilde{\mathbf{Y}}_{k-1} | \mathbf{x}_k) \qquad (7)$$

- From Eqs. (4) and (7), Eq. (2) becomes

$$p(\mathbf{x}_k | \tilde{\mathbf{Y}}_k) = \frac{p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}, \mathbf{x}_k) p(\tilde{\mathbf{Y}}_{k-1} | \mathbf{x}_k) p(\mathbf{x}_k)}{p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}) p(\tilde{\mathbf{Y}}_{k-1})}$$

- Use Bayes' rule on $p(\tilde{\mathbf{Y}}_{k-1} | \mathbf{x}_k)$ to give

$$p(\mathbf{x}_k | \tilde{\mathbf{Y}}_k) = \frac{p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}, \mathbf{x}_k) p(\mathbf{x}_k | \tilde{\mathbf{Y}}_{k-1}) p(\tilde{\mathbf{Y}}_{k-1}) p(\mathbf{x}_k)}{p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}) p(\tilde{\mathbf{Y}}_{k-1}) p(\mathbf{x}_k)}$$

- Using the fact that the measurement sequence is conditionally independent given the current state gives $p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1}, \mathbf{x}_k) = p(\tilde{\mathbf{y}}_k | \mathbf{x}_k)$
- Then

$$p(\mathbf{x}_k | \tilde{\mathbf{Y}}_k) = \frac{p(\tilde{\mathbf{y}}_k | \mathbf{x}_k) p(\mathbf{x}_k | \tilde{\mathbf{Y}}_{k-1})}{p(\tilde{\mathbf{y}}_k | \tilde{\mathbf{Y}}_{k-1})} \qquad (8)$$

MAP KF Derivation (iv)

- By definition $p(\mathbf{x}_k|\tilde{\mathbf{Y}}_{k-1}) \equiv p(\hat{\mathbf{x}}_k^-)$ so Eq. (8) becomes

$$\boxed{p(\mathbf{x}_k|\tilde{\mathbf{Y}}_k) = \frac{p(\tilde{\mathbf{y}}_k|\mathbf{x}_k)p(\hat{\mathbf{x}}_k^-)}{p(\tilde{\mathbf{y}}_k|\tilde{\mathbf{Y}}_{k-1})}} \tag{9}$$

- The MAP estimate for the update is given by maximizing Eq. (9)
- Note that the denominator of Eq. (9) does not depend on $\mathbf{x}_k$
- Taking the natural log of Eq. (9) and using Eq. (1), leads to the following necessary condition

$$\frac{\partial}{\partial \mathbf{x}_k}\left[(\tilde{\mathbf{y}}_k - H_k\mathbf{x}_k)^T R_k^{-1}(\tilde{\mathbf{y}}_k - H_k\mathbf{x}_k) + (\hat{\mathbf{x}}_k^- - \mathbf{x}_k)^T(P_k^-)^{-1}(\hat{\mathbf{x}}_k^- - \mathbf{x}_k)\right]\Big|_{\hat{\mathbf{x}}_k^+} = \mathbf{0}$$

- This yields

$$-H_k^T R_k^{-1}(\tilde{\mathbf{y}}_k - H_k\hat{\mathbf{x}}_k^+) + (P_k^-)^{-1}(\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-) = \mathbf{0}$$

- Rearranging, and adding and subtracting $H_k\hat{\mathbf{x}}_k^-$ yields

$$(P_k^-)^{-1}(\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-) = H_k^T R_k^{-1}(\tilde{\mathbf{y}}_k - H_k\hat{\mathbf{x}}_k^+ + H_k\hat{\mathbf{x}}_k^- - H_k\hat{\mathbf{x}}_k^-)$$

MAP KF Derivation (v)

- Combining terms gives

$$(P_k^-)^{-1}(\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-) = H_k^T R_k^{-1}[\tilde{\mathbf{y}}_k - H_k(\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-) - H_k\hat{\mathbf{x}}_k^-]$$

or

$$\left[(P_k^-)^{-1} + H_k^T R_k^{-1} H_k\right](\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-) = H_k^T R_k^{-1}(\tilde{\mathbf{y}}_k - H_k\hat{\mathbf{x}}_k^-)$$

- Solving for the update gives

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\tilde{\mathbf{y}}_k - H_k\hat{\mathbf{x}}_k^-]$$

$$K_k = P_k^+ H_k^T R_k^{-1}$$

$$P_k^+ \equiv \left[(P_k^-)^{-1} + H_k^T R_k^{-1} H_k\right]^{-1}$$

- Using the matrix inversion lemma on the updated covariance gives

$$P_k^+ = P_k^- - P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1} H_k P_k^-$$

- This shows how the Kalman update equation can be derived from a MAP perspective
- Propagation equations remain the same as before

Joint-Conditional Gaussian (i)

- Consider two random variables $\mathbf{x}$ and $\mathbf{y}$, with means $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_y$ that are jointly Gaussian
  - Define the stack vector $\mathbf{z} \equiv [\mathbf{x}^T \;\; \mathbf{y}^T]^T$ with covariance

$$R^{e_z e_z} \equiv \begin{bmatrix} R^{e_x e_x} & R^{e_x e_y} \\ R^{e_y e_x} & R^{e_y e_y} \end{bmatrix} = \begin{bmatrix} E\left\{\mathbf{e}_x \mathbf{e}_x^T\right\} & E\left\{\mathbf{e}_x \mathbf{e}_y^T\right\} \\ E\left\{\mathbf{e}_y \mathbf{e}_x^T\right\} & E\left\{\mathbf{e}_y \mathbf{e}_y^T\right\} \end{bmatrix}$$

  where $\mathbf{e}_x \equiv \mathbf{x} - \boldsymbol{\mu}_x$ and $\mathbf{e}_y \equiv \mathbf{y} - \boldsymbol{\mu}_y$. Also define $\mathbf{e}_z \equiv \mathbf{z} - \boldsymbol{\mu}_z$, where $\boldsymbol{\mu}_z \equiv [\boldsymbol{\mu}_x^T \;\; \boldsymbol{\mu}_y^T]^T$
  - The block inverse of the covariance is defined by

$$\begin{bmatrix} R^{e_x e_x} & R^{e_x e_y} \\ R^{e_y e_x} & R^{e_y e_y} \end{bmatrix}^{-1} \equiv \begin{bmatrix} \mathcal{R}^{e_x e_x} & \mathcal{R}^{e_x e_y} \\ \mathcal{R}^{e_y e_x} & \mathcal{R}^{e_y e_y} \end{bmatrix} \qquad (1)$$

  - The identities for the matrix partition inverse are

$$(\mathcal{R}^{e_x e_x})^{-1} = R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1} R^{e_y e_x} \qquad (2a)$$

$$(R^{e_y e_y})^{-1} = \mathcal{R}^{e_y e_y} - \mathcal{R}^{e_y e_x}(\mathcal{R}^{e_x e_x})^{-1} \mathcal{R}^{e_x e_y} \qquad (2b)$$

$$(\mathcal{R}^{e_x e_x})^{-1} \mathcal{R}^{e_x e_y} = -R^{e_x e_y}(R^{e_y e_y})^{-1} \qquad (2c)$$

Joint-Conditional Gaussian (ii)

- Recall that the conditional probability is given by

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})}$$

  where $p(\mathbf{x},\mathbf{y})$ is the joint probability
- But $p(\mathbf{x},\mathbf{y})$ is the just $p(\mathbf{z})$
- Then the conditional probability of $\mathbf{x}$ given $\mathbf{y}$ is

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \frac{\left[\det(2\pi R^{e_z e_z})\right]^{-1/2} \exp\left[-\frac{1}{2}\mathbf{e}_z^T (R^{e_z e_z})^{-1}\mathbf{e}_z\right]}{\left[\det(2\pi R^{e_y e_y})\right]^{-1/2} \exp\left[-\frac{1}{2}\mathbf{e}_y^T (R^{e_y e_y})^{-1}\mathbf{e}_y\right]} \\ &= \frac{\left[\det(2\pi R^{e_z e_z})\right]^{-1/2}}{\left[\det(2\pi R^{e_y e_y})\right]^{-1/2}} \exp\left[-\frac{1}{2}\mathbf{e}_z^T (R^{e_z e_z})^{-1}\mathbf{e}_z + \frac{1}{2}\mathbf{e}_y^T (R^{e_y e_y})^{-1}\mathbf{e}_y\right] \end{aligned}$$

- Define the exponent by

$$q \equiv -\frac{1}{2}\mathbf{e}_z^T (R^{e_z e_z})^{-1}\mathbf{e}_z + \frac{1}{2}\mathbf{e}_y^T (R^{e_y e_y})^{-1}\mathbf{e}_y$$

Joint-Conditional Gaussian (iii)

- Substituting Eq. (1) and using the definition of $\mathbf{e}_z$ gives

$$q = -\frac{1}{2}\begin{bmatrix}\mathbf{e}_x^T & \mathbf{e}_y^T\end{bmatrix}\begin{bmatrix}\mathcal{R}^{e_xe_x} & \mathcal{R}^{e_xe_y}\\ \mathcal{R}^{e_ye_x} & \mathcal{R}^{e_ye_y}\end{bmatrix}\begin{bmatrix}\mathbf{e}_x\\ \mathbf{e}_y\end{bmatrix} + \frac{1}{2}\mathbf{e}_y^T(R^{e_ye_y})^{-1}\mathbf{e}_y$$

$$= -\frac{1}{2}(\mathbf{e}_x^T\mathcal{R}^{e_xe_x}\mathbf{e}_x + \mathbf{e}_x^T\mathcal{R}^{e_xe_y}\mathbf{e}_y + \mathbf{e}_y^T\mathcal{R}^{e_ye_x}\mathbf{e}_x + \mathbf{e}_y^T\mathcal{R}^{e_ye_y}\mathbf{e}_y) + \frac{1}{2}\mathbf{e}_y^T(R^{e_ye_y})^{-1}\mathbf{e}_y$$

- Let's prove that

$$\begin{aligned}a &\equiv \mathbf{e}_x^T\mathcal{R}^{e_xe_x}\mathbf{e}_x + \mathbf{e}_x^T\mathcal{R}^{e_xe_y}\mathbf{e}_y + \mathbf{e}_y^T\mathcal{R}^{e_ye_x}\mathbf{e}_x + \mathbf{e}_y^T\mathcal{R}^{e_ye_y}\mathbf{e}_y\\ &= (\mathbf{e}_x + (\mathcal{R}^{e_xe_x})^{-1}\mathcal{R}^{e_xe_y}\mathbf{e}_y)^T\mathcal{R}^{e_xe_x}(\mathbf{e}_x + (\mathcal{R}^{e_xe_x})^{-1}\mathcal{R}^{e_xe_y}\mathbf{e}_y)\\ &+ \mathbf{e}_y^T(\mathcal{R}^{e_ye_y} - \mathcal{R}^{e_ye_x}(\mathcal{R}^{e_xe_x})^{-1}\mathcal{R}^{e_xe_y})\mathbf{e}_y\end{aligned}$$

- Multiplying all terms and using $(R^{e_xe_y})^T = R^{e_ye_x}$ gives

$$\begin{aligned}a &= \mathbf{e}_x^T\mathcal{R}^{e_xe_x}\mathbf{e}_x + \mathbf{e}_x^T\mathcal{R}^{e_xe_y}\mathbf{e}_y + \mathbf{e}_y^T\mathcal{R}^{e_ye_x}\mathbf{e}_x\\ &+ \mathbf{e}_y^T(\mathcal{R}^{e_ye_x}(\mathcal{R}^{e_xe_x})^{-1}\mathcal{R}^{e_xe_y})\mathbf{e}_y + \mathbf{e}_y^T\mathcal{R}^{e_ye_y}\mathbf{e}_y - \mathbf{e}_y^T(\mathcal{R}^{e_ye_x}(\mathcal{R}^{e_xe_x})^{-1}\mathcal{R}^{e_xe_y})\mathbf{e}_y\\ &= \mathbf{e}_x^T\mathcal{R}^{e_xe_x}\mathbf{e}_x + \mathbf{e}_x^T\mathcal{R}^{e_xe_y}\mathbf{e}_y + \mathbf{e}_y^T\mathcal{R}^{e_ye_x}\mathbf{e}_x + \mathbf{e}_y^T\mathcal{R}^{e_ye_y}\mathbf{e}_y \checkmark\end{aligned}$$

Joint-Conditional Gaussian (iv)

- Then

$$q = -\frac{1}{2}\Big[(\mathbf{e}_x + (\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y}\mathbf{e}_y)^T \mathcal{R}^{e_x e_x}(\mathbf{e}_x + (\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y}\mathbf{e}_y)$$
$$+ \mathbf{e}_y^T(\mathcal{R}^{e_y e_y} - \mathcal{R}^{e_y e_x}(\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y})\mathbf{e}_y\Big] + \frac{1}{2}\mathbf{e}_y^T(R^{e_y e_y})^{-1}\mathbf{e}_y$$

- Using Eq. (2a) simplifies the above expression to simply

$$q = -\frac{1}{2}(\mathbf{e}_x + (\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y}\mathbf{e}_y)^T \mathcal{R}^{e_x e_x}(\mathbf{e}_x + (\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y}\mathbf{e}_y)$$

- Now look at

$$\frac{[\det(2\pi R^{e_z e_z})]^{-1/2}}{[\det(2\pi R^{e_y e_y})]^{-1/2}} = \frac{1}{[\det(2\pi R^{e_y e_y})]^{-1/2}[\det(2\pi R^{e_z e_z})]^{1/2}}$$

- Use the following relation on $R^{e_z e_z}$

$$\det\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \det(D)\,\det(A - B\,D^{-1}C)$$

Joint-Conditional Gaussian (v)

- Then

$$[\det(2\pi R^{e_y e_y})]^{-1/2}\,[\det(2\pi R^{e_z e_z})]^{1/2}$$
$$= [\det(2\pi R^{e_y e_y})]^{-1/2}\,[\det(2\pi R^{e_y e_y})]^{1/2}\left\{\det[2\pi(R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1}R^{e_y e_x})]\right\}^{1/2}$$
$$= \left\{\det[2\pi(R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1}R^{e_y e_x})]\right\}^{1/2}$$

- So

$$\frac{[\det(2\pi R^{e_z e_z})]^{-1/2}}{[\det(2\pi R^{e_y e_y})]^{-1/2}} = \frac{1}{\left\{\det[2\pi(R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1}R^{e_y e_x})]\right\}^{1/2}}$$

- Finally, $p(\mathbf{x}|\mathbf{y})$ becomes

$$p(\mathbf{x}|\mathbf{y}) = \frac{e^q}{\left\{\det[2\pi(R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1}R^{e_y e_x})]\right\}^{1/2}}$$

- Thus the conditional probability is also Gaussian

Joint-Conditional Gaussian (vi)

- Using the definitions of $\mathbf{e}_x$, $\mathbf{e}_y$ and the identity $(\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y} = -R^{e_x e_y}(R^{e_y e_y})^{-1}$ from Eq. (2c) allows us to write

$$\mathbf{e}_x + (\mathcal{R}^{e_x e_x})^{-1}\mathcal{R}^{e_x e_y}\mathbf{e}_y = \mathbf{x} - \boldsymbol{\mu}_x - R^{e_x e_y}(R^{e_y e_y})^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)$$

  - Then the conditional mean of $\mathbf{x}$ given $\mathbf{y}$ is simply

$$E\left\{\mathbf{x}|\mathbf{y}\right\} = \boldsymbol{\mu}_x + R^{e_x e_y}(R^{e_y e_y})^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)$$

  - Its covariance is given by

$$\operatorname{cov}\left\{\mathbf{x}|\mathbf{y}\right\} = R^{e_x e_x} - R^{e_x e_y}(R^{e_y e_y})^{-1}R^{e_y e_x}$$

  - These equations for the conditional mean and covariance are cornerstones for much of the developments of linear estimation using Gaussian variables

Relation to KF (i)

- The previous derivation can be related to the Kalman filter if we note the following definitions

$$\hat{\mathbf{x}}_k^+ \equiv E\left\{\mathbf{x}|\mathbf{y}\right\}, \quad \hat{\mathbf{x}}_k^- \equiv \boldsymbol{\mu}_x$$
$$P_k^{e_x e_y} \equiv R^{e_x e_y}, \quad P_k^{e_y e_y} \equiv R^{e_y e_y}, \quad K_k \equiv P_k^{e_x e_y}(P_k^{e_y e_y})^{-1} \tag{1}$$
$$\mathbf{e}_k^- \equiv \tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k^- = \mathbf{y} - \boldsymbol{\mu}_y$$

with

$$P_k^{e_x e_y} = E\left\{\left(\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^-\right)\mathbf{e}_k^{-T}\right\}$$
$$P_k^{e_y e_y} = E\left\{\mathbf{e}_k^- \, \mathbf{e}_k^{-T}\right\}$$

- Compute the second expectation equation by first using

$$\begin{aligned}\mathbf{e}_k^- &\equiv \tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k^- \\ &= H_k \mathbf{x} + \mathbf{v}_k - H_k \hat{\mathbf{x}}_k^- \\ &= H_k\left(\mathbf{x} - \hat{\mathbf{x}}_k^-\right) + \mathbf{v}_k\end{aligned}$$

- Then the “innovations covariance” is given by

$$P_k^{e_y e_y} = H_k P_k^- H_k^T + R_k \tag{2}$$

Relation to KF (ii)

- The first expectation equation is the cross-covariance
- From $\hat{\mathbf{x}}_k^+ - \hat{\mathbf{x}}_k^- = K_k \mathbf{e}_k^-$ and Eq. (2) we have

$$\begin{aligned} P_k^{e_x e_y} &= E\left\{K_k \mathbf{e}_k^- \, \mathbf{e}_k^{-T}\right\} \\ &= K_k (H_k P_k^- H_k^T + R_k) \\ &= P_k^- H_k^T \end{aligned}$$

where $K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$ was used

- Then from the definitions in Eq. (1), the Kalman update and covariance expressions are given by

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k \mathbf{e}_k^-$$
$$K_k = P_k^{e_x e_y} (P_k^{e_y e_y})^{-1}$$
$$P_k^+ = P_k^- - K_k P_k^{e_y e_y} K_k^T$$

- Just another form for the Kalman filter equations
- Useful for analysis purposes and in the derivation of the Unscented filter

EKF (Slight Return)

- Truth equations

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k,\, \mathbf{w}_k,\, \mathbf{u}_k,\, k), \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, Q_k)$$

$$\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k,\, \mathbf{u}_k,\, \mathbf{v}_k,\, k), \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, R_k)$$

- Update equations

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k \mathbf{e}_k^-$$

$$P_k^+ = P_k^- - K_k P_k^{e_y e_y} K_k^T$$

- Innovation sequence and gain calculation

residual

$$\mathbf{e}_k^- \equiv \tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k^- = \tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-,\, \mathbf{u}_k, k)$$

$$K_k = P_k^{e_x e_y} (P_k^{e_y e_y})^{-1}$$

$P_k^{e_y e_y} \Rightarrow$ covariance of $\mathbf{e}_k^-$, $P_k^{e_x e_y} \Rightarrow$ cross correlation of $\hat{\mathbf{x}}_k^-$, $\hat{\mathbf{y}}_k^-$

  - Note, we've shown for linear systems that these equations reduce down to the normal KF equations

Unscented Transformation (i)

- Basic premise ⇒ easier to approximate a Gaussian distribution than to approximate a nonlinear function
- Advantages over extended Kalman filter
  - Can be applied to non-differentiable functions
  - No Jacobian or higher derivatives are required
  - Valid to higher-order expansions

Output is still assumed to be Gaussian

- Same basic structure as the extended Kalman filter
  - Different covariance propagation and cross-correlation approach
  - Decomposes the covariance matrix to form "sigma points"
- First introduced by Julier and Uhlmann, 1996
  - Showed several examples of its advantages over the EKF
  - Since that time there have been many applications
    - Quaternion attitude estimation
    - GPS/INS Applications

Unscented Transformation (ii)

random variable

- Suppose we have an arbitrary function $\mathbf{y} = \mathbf{h}(\mathbf{x})$
- Translated sigma points from covariance P_{xx} and mean $\bar{\mathbf{x}}$

$$\boldsymbol{\sigma} \leftarrow 2n \text{ rows or columns from } \pm\sqrt{(n+\lambda)P_{xx}}$$

Matrix SQRT, use cholesky decomposition

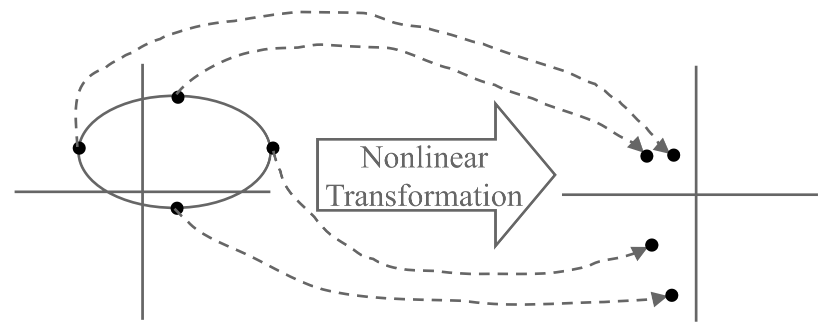
dimension of x

tuning parameters

$$\boldsymbol{\chi}^{(0)} = \bar{\mathbf{x}}, \quad \boldsymbol{\chi}^{(i)} = \boldsymbol{\sigma}^{(i)} + \bar{\mathbf{x}}, \quad i = 1, 2, \ldots, 2n$$

- Transformed sigma points $\boldsymbol{\gamma}^{(i)} = \mathbf{h}[\boldsymbol{\chi}^{(i)}]$
- Predicted mean and covariance

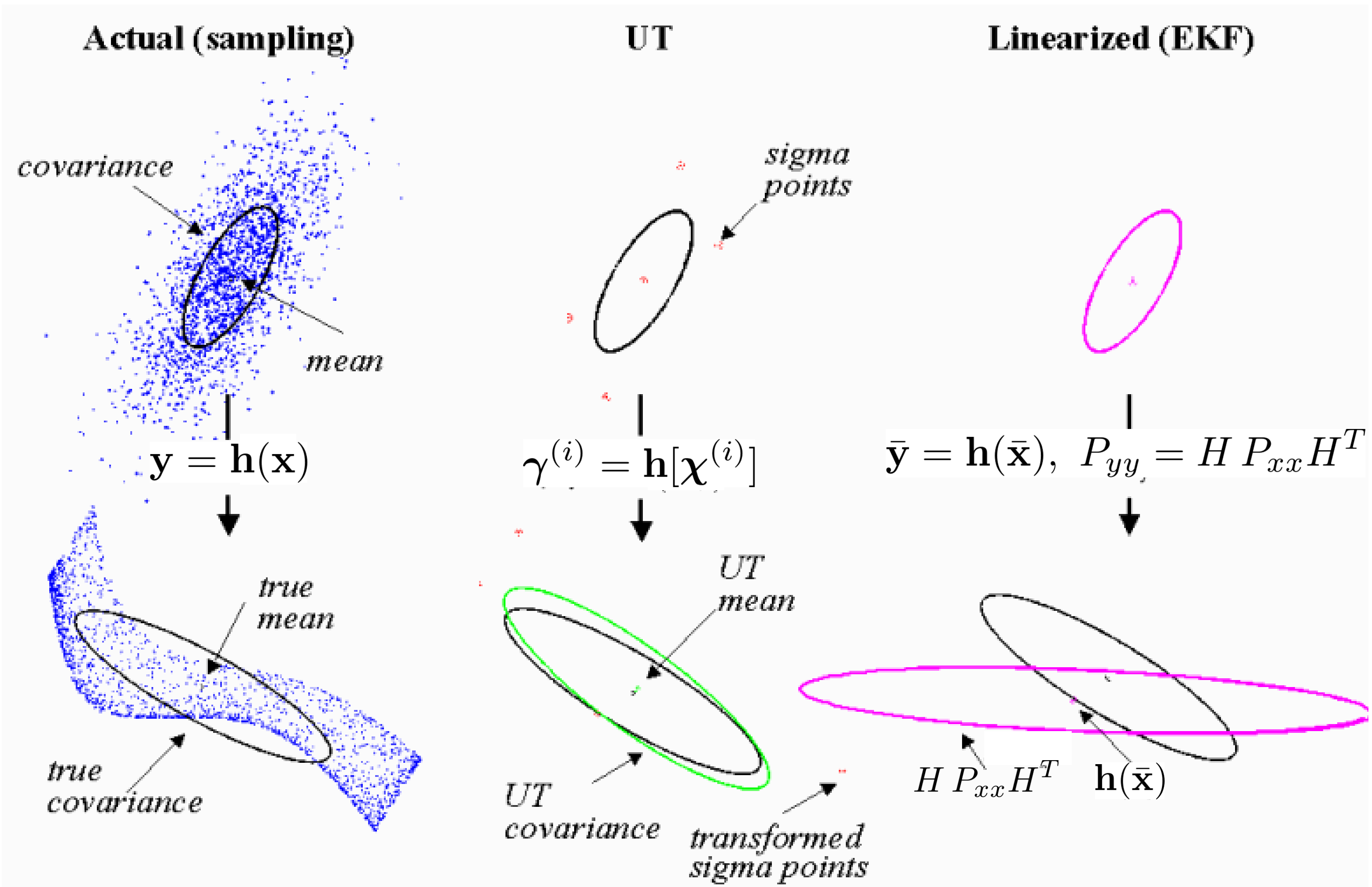
$$\bar{\mathbf{y}} = \frac{1}{n+\lambda}\left\{\lambda\boldsymbol{\gamma}^{(0)} + \sum_{i=1}^{2n}\boldsymbol{\gamma}^{(i)}\right\}$$



$$P_{yy} = \frac{1}{n+\lambda}\left\{\lambda\left[\boldsymbol{\gamma}^{(0)} - \bar{\mathbf{y}}\right]\left[\boldsymbol{\gamma}^{(0)} - \bar{\mathbf{y}}\right]^T + \sum_{i=1}^{2n}\left[\boldsymbol{\gamma}^{(i)} - \bar{\mathbf{y}}\right]\left[\boldsymbol{\gamma}^{(i)} - \bar{\mathbf{y}}\right]^T\right\}$$

Unscented Transformation (iii)

From Wan and van der Merwe



Simple Example

- Julier and Uhlmann, 1996 considered $y = h(x) = x^2$
  - x is normally distributed with mean $\bar{x}$ and variance σ_x^2
- True mean and variance

$$\bar{y} = \bar{x}^2 + \sigma_x^2, \quad \sigma_y^2 = 2\sigma_x^4 + 4\bar{x}^2\sigma_x^2$$

- Linearized mean and variance (mean is biased)

$$\bar{y} = \bar{x}^2, \quad \sigma_y^2 = 4\bar{x}^2\sigma_x^2$$

- Unscented transformation $\Rightarrow$ calculate sigma points

$$\{\chi^{(0)}, \chi^{(1)}, \chi^{(2)}\} = \{\bar{x}, \bar{x} - \sigma, \bar{x} + \sigma\}, \text{ with } \sigma = \sqrt{(n+\lambda)\sigma_x^2}$$

$$\{\gamma^{(0)}, \gamma^{(1)}, \gamma^{(2)}\} = \{\bar{x}^2, \bar{x}^2 - 2\bar{x}\sigma + \sigma^2, \bar{x}^2 + 2\bar{x}\sigma + \sigma^2\}$$

- Unscented mean and variance

$$\bar{y} = \bar{x}^2 + \sigma_x^2, \quad \sigma_y^2 = \lambda\sigma_x^4 + 4\bar{x}^2\sigma_x^2$$

  - Equal to the truth when $\lambda = 2$

Scaled Unscented Filter (i)

- Form an augmented covariance matrix

$$P_k^a = \begin{bmatrix} P_k^- & P_k^{xw} & P_k^{xv} \\ (P_k^{xw})^T & Q_k & P_k^{wv} \\ (P_k^{xv})^T & (P_k^{wv})^T & R_k \end{bmatrix}$$

- Compute sigma points $\Rightarrow$ Cholesky decomposition

$$\boldsymbol{\sigma}_k \leftarrow 2L \text{ columns from } \pm\gamma\sqrt{P_k^a}$$

$$\boldsymbol{\chi}_k^{a(0)} = \hat{\mathbf{x}}_k^a$$

$$\boldsymbol{\chi}_k^{a(i)} = \boldsymbol{\sigma}_k^{(i)} + \hat{\mathbf{x}}_k^a$$

where

$$\mathbf{x}_k^a = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{w}_k \\ \mathbf{v}_k \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{\chi}_k^{x(i)} \\ \boldsymbol{\chi}_k^{w(i)} \\ \boldsymbol{\chi}_k^{v(i)} \end{bmatrix}, \quad \hat{\mathbf{x}}_k^a = \begin{bmatrix} \hat{\mathbf{x}}_k^- \\ \mathbf{0}_{q\times 1} \\ \mathbf{0}_{m\times 1} \end{bmatrix}, \quad \begin{array}{c} \gamma = \sqrt{L+\lambda} \\ \lambda = \alpha^2(L+\kappa) - L \\ 1\times 10^{-4} \le \alpha \le 1 \end{array}$$

- Note, $\alpha = 1$ and $\beta = 0$ gives previous (unscaled) version

Scaled Unscented Filter (ii)

- Transformed set of sigma points

$$\boldsymbol{\chi}_{k+1}^{x(i)} = \mathbf{f}(\boldsymbol{\chi}_k^{x(i)},\, \boldsymbol{\chi}_k^{w(i)},\, \mathbf{u}_k,\, k)$$
$$\boldsymbol{\gamma}_k^{(i)} = \mathbf{h}(\boldsymbol{\chi}_k^{x(i)},\, \mathbf{u}_k,\, \boldsymbol{\chi}_k^{v(i)},\, k)$$

- Weights

$$W_0^{\text{mean}} = \frac{\lambda}{L+\lambda}$$
$$W_0^{\text{cov}} = \frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta)$$
$$W_i^{\text{mean}} = W_i^{\text{cov}} = \frac{1}{2(L+\lambda)}, \quad i = 1,\, 2,\, \ldots,\, 2L$$

- Output and cross-correlation (perform updates)

$$P_k^{yy} = \sum_{i=0}^{2L} W_i^{\text{cov}}\, [\boldsymbol{\gamma}_k^{(i)} - \hat{\mathbf{y}}_k^-]\, [\boldsymbol{\gamma}_k^{(i)} - \hat{\mathbf{y}}_k^-]^T = P_k^{e_y e_y}$$
$$P_k^{e_x e_y} = \sum_{i=0}^{2L} W_i^{\text{cov}}\, [\boldsymbol{\chi}_k^{x(i)} - \hat{\mathbf{x}}_k^-]\, [\boldsymbol{\gamma}_k^{(i)} - \hat{\mathbf{y}}_k^-]^T$$

Scaled Unscented Filter (iii)

- Propagated mean estimate and output

$$\hat{\mathbf{x}}_{k+1}^{-} = \sum_{i=0}^{2L} W_i^{\text{mean}} \, \boldsymbol{\chi}_{k+1}^{x(i)}$$

$$\hat{\mathbf{y}}_{k+1}^{-} = \sum_{i=0}^{2L} W_i^{\text{mean}} \, \gamma_{k+1}^{(i)}$$

- Propagated covariance

$$P_{k+1}^{-} = \sum_{i=0}^{2L} W_i^{\text{cov}} \, [\boldsymbol{\chi}_{k+1}^{x(i)} - \hat{\mathbf{x}}_{k+1}^{-}] \, [\boldsymbol{\chi}_{k+1}^{x(i)} - \hat{\mathbf{x}}_{k+1}^{-}]^T$$

- Measurement error appears linearly? If yes, then

$$P_k^a = \begin{bmatrix} P_k^{-} & P_k^{xw} \\ (P_k^{xw})^T & Q_k \end{bmatrix}, \quad P_k^{e_y e_y} = P_k^{yy} + R_k$$

Scaled Unscented Filter (iv)

- Steps for Unscented Filter
  - Given $\hat{\mathbf{x}}_k^-$, $\hat{\mathbf{y}}_k^-$, P_k^-, first form augmented covariance P_k^a
  - Get sigma points and $\boldsymbol{\chi}_k^{a(i)}$
  - Get output points $\gamma_k^{(i)}$
  - Compute output covariance P_k^{yy} and cross correlation $P_k^{e_x e_y}$
  - Compute innovations covariance $P_k^{e_y e_y}$ and gain K_k
  - Update state and covariance $\Rightarrow$ gives estimate at time t_k
  - Calculate propagated state points $\boldsymbol{\chi}_{k+1}^{x(i)}$
  - Compute propagated state $\hat{\mathbf{x}}_{k+1}^-$ and covariance P_{k+1}^-
  - Form augmented covariance and repeat

UF Assumption

- UF still assumes a Gaussian distribution
  - True for both the prior and posterior distributions
  - Only two variables are required to fully described the pdf: the mean and the covariance
  - "Particles" of the UF exactly match input covariance

$$\boldsymbol{\sigma} \leftarrow 2n \text{ rows or columns from } \pm\sqrt{(n+\lambda)P_{xx}}$$

$$\boldsymbol{\chi}^{(0)} = \bar{\mathbf{x}}, \quad \boldsymbol{\chi}^{(i)} = \boldsymbol{\sigma}^{(i)} + \bar{\mathbf{x}}, \quad i = 1, 2, \ldots, 2n$$

$$\bar{\mathbf{x}} = \frac{1}{n+\lambda}\left\{\lambda\boldsymbol{\chi}^{(0)} + \frac{1}{2}\sum_{i=1}^{2n}\boldsymbol{\chi}^{(i)}\right\}$$

$$P_{xx} = \frac{1}{n+\lambda}\left\{\lambda\left[\boldsymbol{\chi}^{(0)} - \bar{\mathbf{x}}\right]\left[\boldsymbol{\chi}^{(0)} - \bar{\mathbf{x}}\right]^T + \frac{1}{2}\sum_{i=1}^{2n}\left[\boldsymbol{\chi}^{(i)} - \bar{\mathbf{x}}\right]\left[\boldsymbol{\chi}^{(i)} - \bar{\mathbf{x}}\right]^T\right\}$$

  - Only the mean and covariance of the first two moments of the underlying distribution are maintained
    - Accurate to at least 2^{nd}-order in Taylor series expansion (3^{rd}-order for Gaussian inputs)

UF MATLAB Code Example

```
% Input Mean and Covariance
n=4;
x_bar=randn(n,1)
p_xx=randn(n);p_xx=p_xx*p_xx'

% Matrix square root
lambda=abs(randn(1));
psquare=chol((n+lambda)*p_xx)';
sig=real([psquare -psquare]);
chi=sig+kron(x_bar,ones(1,2*n));

% Mean
x_est=1/(n+lambda)*(lambda*x_bar'+0.5*sum(chi,2)')'

% Covariance
pmat=chi-kron(x_est,ones(1,2*n));
p_est=1/(n+lambda)*(lambda*(x_bar-x_est)*(x_bar-x_est)'+0.5*pmat*pmat')
```

MATLAB does cholesky decomp using

```
x_bar =

    0.4998
    1.2781
   -0.5478
    0.2608


p_xx =

    2.3911   -1.5379   -1.2075    1.6667
   -1.5379    6.4358   -3.1783   -1.6417
   -1.2075   -3.1783    6.8739   -1.0540
    1.6667   -1.6417   -1.0540    1.3371


x_est =

    0.4998
    1.2781
   -0.5478
    0.2608


p_est =

    2.3911   -1.5379   -1.2075    1.6667
   -1.5379    6.4358   -3.1783   -1.6417
   -1.2075   -3.1783    6.8739   -1.0540
    1.6667   -1.6417   -1.0540    1.3371
```

Example (i)

- Consider the following state-space model

$$F = \begin{bmatrix} -4 & -3 & -4 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Choosing a sampling interval of 0.1 seconds gives

$e^{F\Delta t}$

$$\Phi = \begin{bmatrix} 0.6583 & -0.2637 & -0.3324 & -0.0820 \\ 0.0820 & 0.9862 & -0.0177 & -0.0044 \\ 0.0044 & 0.0995 & 0.9994 & -0.0002 \\ 0.0002 & 0.0050 & 0.1000 & 1.0000 \end{bmatrix}$$

- Initial conditions for the true and estimated states

$$\mathbf{x}_0 = \begin{bmatrix} 1 & 0 & 2 & 0 \end{bmatrix}^T, \quad \hat{\mathbf{x}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$$

- Initial covariance

$$P_0 = \text{diag}\begin{bmatrix} (2/3)^2 & 0.001 & (4/3)^2 & 0.001 \end{bmatrix}$$

  - First state has a 3σ bound of 2, third state has a 3σ bound of 4, and second and fourth state are known pretty accurately

Example (ii)

- Measurement error covariance

$$R = \begin{bmatrix} 0.01 & 0.005 \\ 0.005 & 0.02 \end{bmatrix}$$

tuning model

- Continuous-time process noise given by $Q = 0.01$, with

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$$

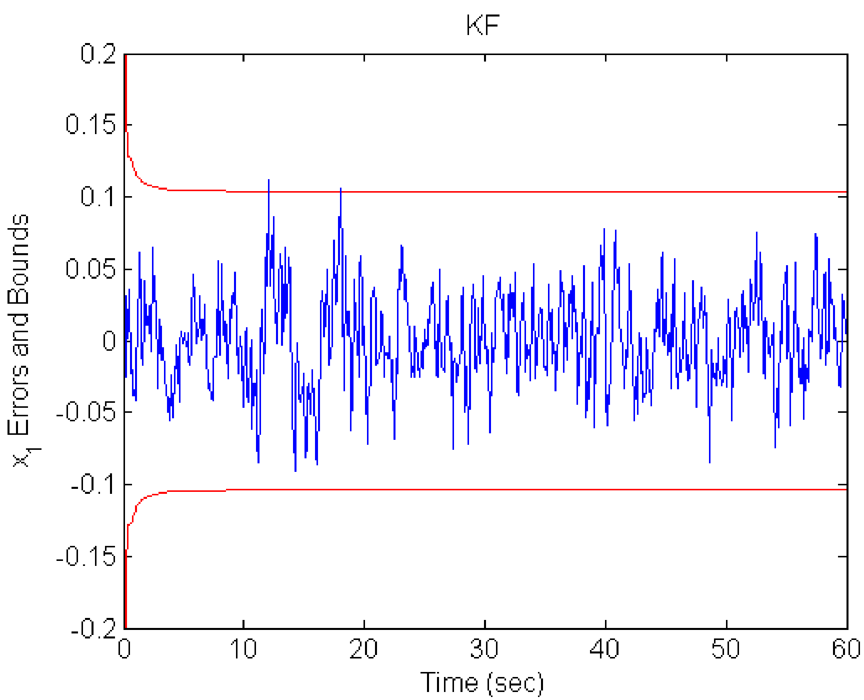
so that

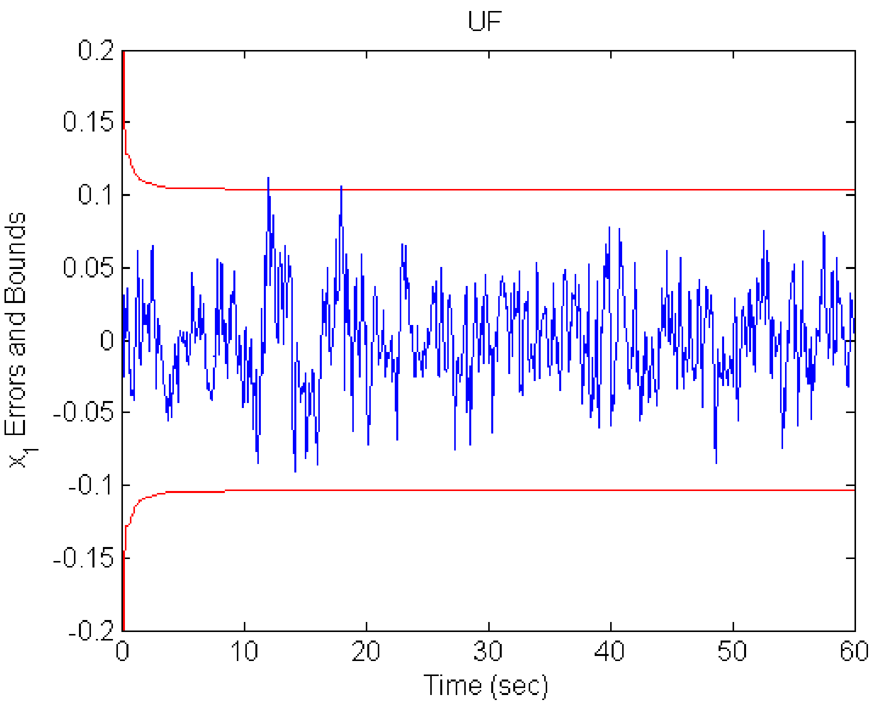
$$G\,Q\,G^T = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- Note, process noise is only added to the first state
  - The rest are just kinematic states, which have no error
- Ran both the standard Kalman filter and Unscented filter
  - Gave nearly identical results to almost within machine precision

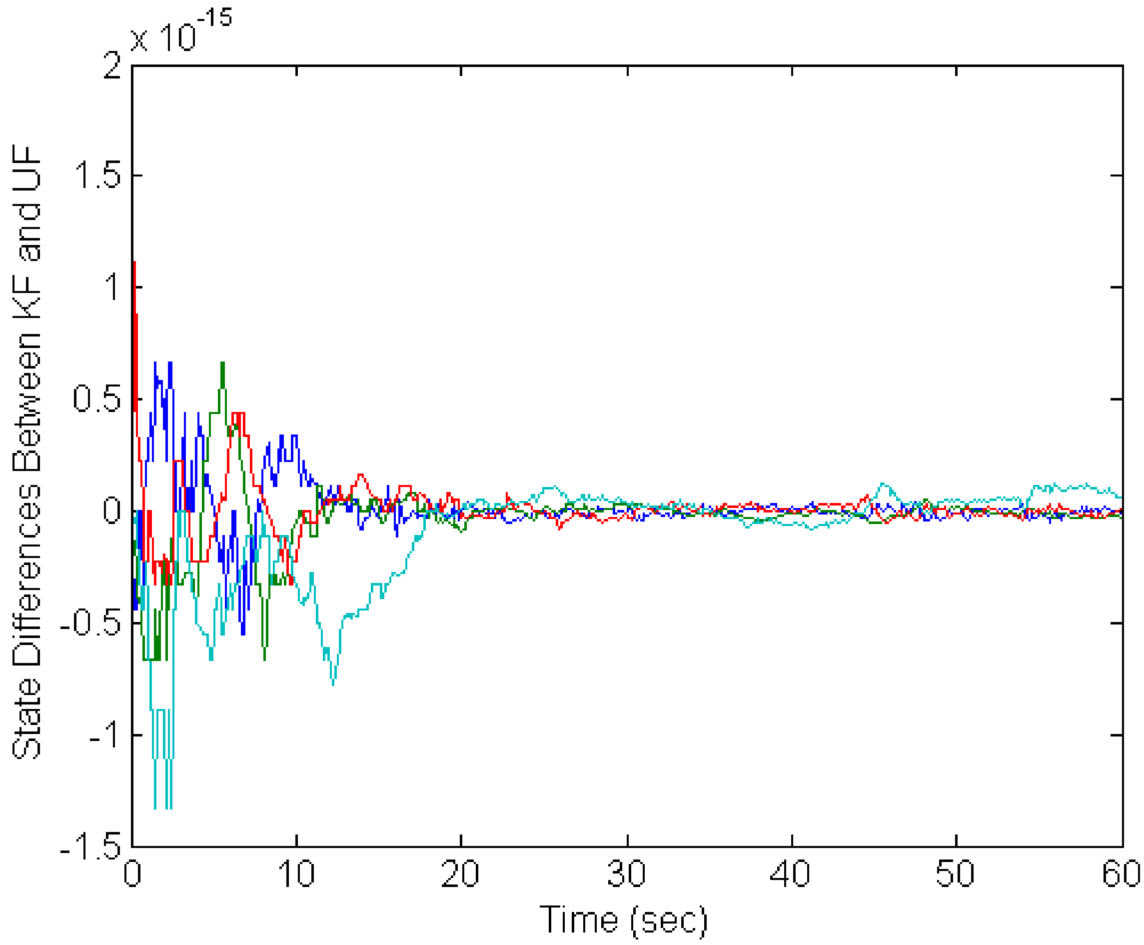
Example (iii)

First State Errors and 3σ Bounds





Example (iv)



x 10^-15
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
10
20
30
40
50
60
Time (sec)
State Differences Between KF and UF

Example (v)

```
% Output
h=[1 0 0 0;0 0 1 0];

% Sampling Interval and Time
dt=0.1;tf=60;t=[0:dt:tf]';m=length(t);

% State Model
f=[-4 -3 -4 -1;eye(3) zeros(3,1)];
n=4;

% Measurement Covariance
r=[0.01 0.005;0.005 0.02];

% Generate Correlated Noise
[tt,lam]=eig(r);
v_uncorr=randn(m,2)*lam.^(0.5);
v=(tt*v_uncorr')';

% Process Noise Spectral Density
q=zeros(n);q(1,1)=1e-2;
```

Example (vi)

```
% Get Discrete-Time Matrices
bmat=expm([-f q;zeros(4) f']*dt);
phi=bmat(n+1:2*n,n+1:2*n)';
qd=phi*bmat(1:n,n+1:2*n);

% Generate Correlated Noise
[tt,lam]=eig(qd);
w_uncorr=randn(m,n)*lam.^(0.5);
w=(tt*w_uncorr')';

% Get Measurements
x=zeros(m,4);x(1,:)=[1 0 2 0];
for i=1:m-1
 x(i+1,:)=(phi*x(i,:)')'+w(i,:);
end
y=(h*x')';ym=y+[v(:,1) v(:,2)];

% Kalman Filter Parameters
xe_kf=zeros(m,4);p_cov_kf=zeros(m,4);
p_kf=diag([(2/3)^2 0.001 (4/3)^2 0.001]);p_cov_kf(1,:)=diag(p_kf)';
```

Example (vii)

```
% Kalman Filter Loop
for i=1:m-1

% Propagation
p_kf=phi*p_kf*phi'+qd;
xe_kf(i+1,:)=(phi*xe_kf(i,:)')';

% Update
gain=p_kf*h'*inv(h*p_kf*h'+r);
p_kf=(eye(4)-gain*h)*p_kf;
p_cov_kf(i+1,:)=diag(p_kf)';
xe_kf(i+1,:)=xe_kf(i+1,:)+(gain*(ym(i+1,:)'-h*xe_kf(i+1,:)'))';

end
```

Example (viii)

```
% Unscented Filter Parameters
xe_uf=zeros(m,4);p_cov_uf=zeros(m,4);
p_uf=diag([(2/3)^2 0.001 (4/3)^2 0.001]);p_cov_uf(1,:)=diag(p_uf)';

alp=1;beta=0;ell=2*n;kap=3+ell;
lam=alp^2*(ell+kap)-ell;
w0m=lam/(ell+lam);
w0c=lam/(ell+lam)+(1-alp^2+beta);
wim=1/(2*(ell+lam));

% No off-diagnoal elements in the augmented covariance,
% so we can take the cholesky of each separately.
% But note that for the filter with augmented state,
% both xx and sigw need to be 4 x 16 instead of 4 x 8,
% which is the case for the non-augmented state.
% Take cholelsky of qd (note it's constant)
psquarew=chol(qd)';
sigw=[zeros(n,2*n) sqrt(ell+lam)*psquarew -sqrt(ell+lam)*psquarew];
```

Example (ix)

Ozzie bear therom

```
% Unscented Filter Loop
for i=1:m-1

% Covariance Decomposition
psquare=chol(p_uf)';
sigv=real([sqrt(ell+lam)*psquare -sqrt(ell+lam)*psquare]);
xx0=xe_uf(i,:)';
xx=[sigv+kron(xe_uf(i,:)',ones(1,2*n)) repmat(xx0,1,2*n)];

% Propagation
xx0=phi*xx0;
xx=phi*xx+sigw;
xe_uf(i+1,:)=w0m*xx0'+wim*sum(xx,2)';

% Covariance
pp0=w0c*(xx0-xe_uf(i+1,:)')*(xx0-xe_uf(i+1,:)')';
pmat=xx-kron(xe_uf(i+1,:)',ones(1,4*n));
p_uf=pp0+wim*pmat*pmat';
```

=ozzie+meg+gabe=family

hi Gabe!

Satisfying

Example (x)

```
% Output
yez=h*xx;ye0=h*xx0;ye=w0m*ye0+wim*sum(yez,2);

% Calculate pyy
pyy0=w0c*(ye0-ye)*(ye0-ye)';pyymat=yez-kron(ye,ones(1,4*n));
pyy=pyy0+wim*pyymat*pyymat';

% Calculate pxy
pxy0=w0c*(xx0-xe_uf(i+1,:)')*(ye0-ye)';
pxy=pxy0+wim*pmat*pyymat';

% Innovations Covariance
pvv=pyy+r;

% Gain and Update
gain=real(pxy*inv(pvv));
p_uf=p_uf-gain*pvv*gain';p_cov_uf(i+1,:)=diag(p_uf)';
xe_uf(i+1,:)=xe_uf(i+1,:)+(gain*(ym(i+1,:)'-ye))';

end
```

Example (xi)

```
% Plot Results
figure(1)
sig3_kf=p_cov_kf.^(0.5)*3;
clf
plot(t,sig3_kf(:,1),'r',t,xe_kf(:,1)-x(:,1),'b',t,-sig3_kf(:,1),'r')
set(gca,'fontsize',12)
axis([0 60 -0.2 0.2])
ylabel('{x_1} Errors and Bounds')
xlabel('Time (sec)')
title('KF')

figure(2)
sig3_uf=p_cov_uf.^(0.5)*3;
clf
plot(t,sig3_uf(:,1),'r',t,xe_uf(:,1)-x(:,1),'b',t,-sig3_uf(:,1),'r')
set(gca,'fontsize',12)
axis([0 60 -0.2 0.2])
ylabel('{x_1} Errors and Bounds')
xlabel('Time (sec)')
title('UF')
```

My Name

Example (xii)

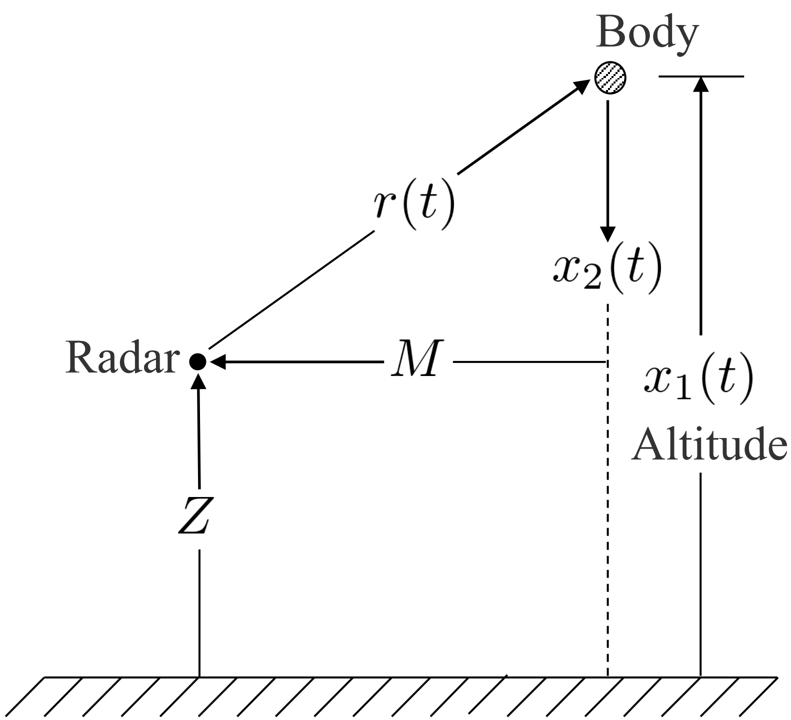
```
figure(3)
clf
plot(t,xe_kf-xe_uf)
set(gca,'fontsize',12)
ylabel('State Differences Between KF and UF')
xlabel('Time (sec)')
```

is . . .

Ozzie Bear

Dog

Free Falling Body Example (i)



$$\dot{x}_1(t) = -x_2(t)$$

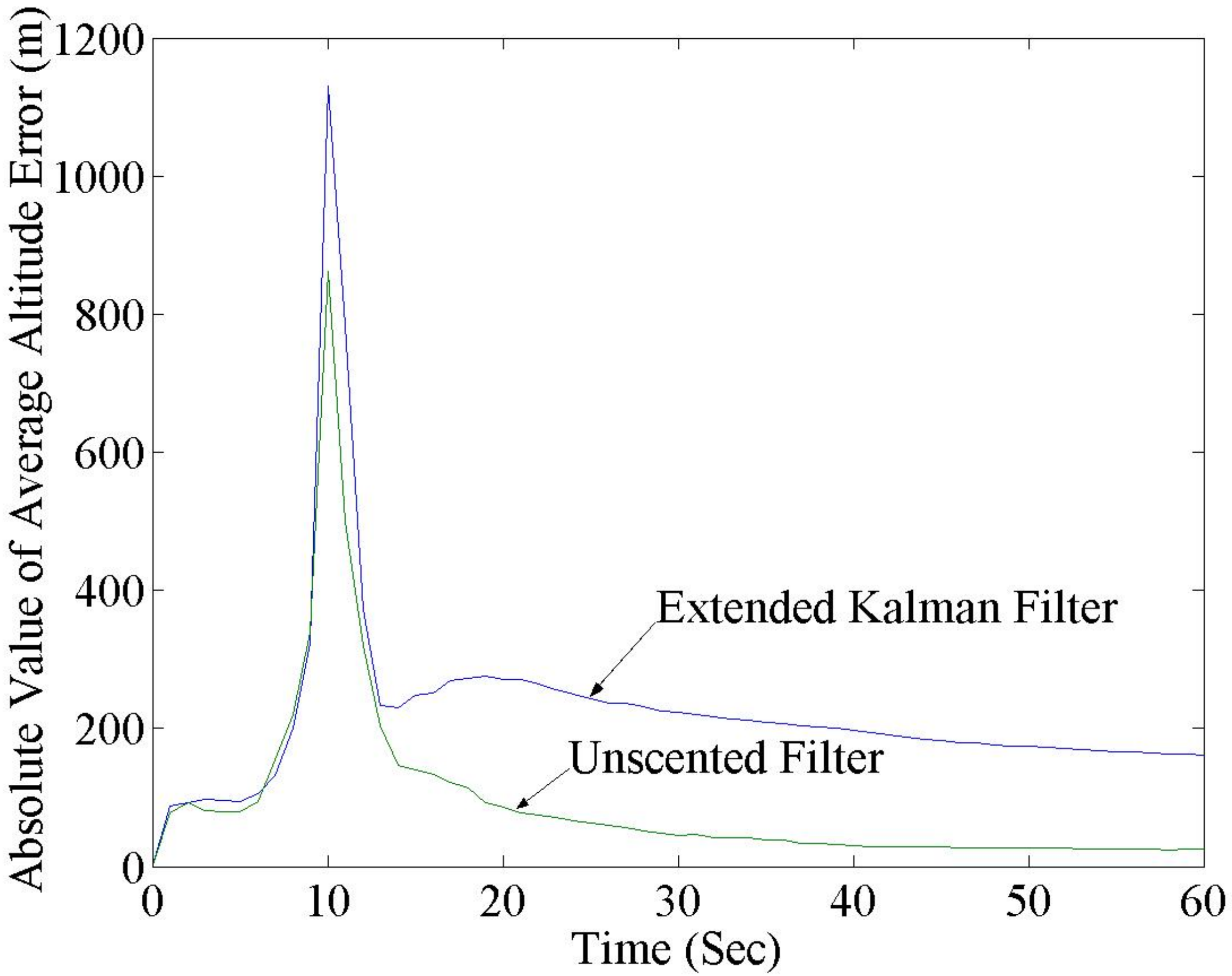
$$\dot{x}_2(t) = -e^{-\alpha\, x_1(t)}\, x_2^2(t)\, x_3(t)$$

$$\dot{x}_3(t) = 0$$

$$\tilde{y}_k = \sqrt{M^2 + (x_{1_k} - Z)^2} + v_k$$

- Estimate ballistic coefficient, x_3, of a vertical falling body
- Measure range by a radar
- M and Z are constants
- α relates air density with altitude (constant)
- No process noise
  - Nonlinearities cause the problems, not process noise
- EKF and UF comparisons
  - Same initial covariances
  - Monte Carlo simulation with 100 runs

Free Falling Body Example (ii)



Free Falling Body Example (iii)

```
% Initialize
dt=1;tf=60;t=[0:dt:tf]';m=length(t); % stores samples every second
runs=100;
xes1=zeros(m,runs);xes2=zeros(m,runs);xes3=zeros(m,runs);
xes1_kf=zeros(m,runs);xes2_kf=zeros(m,runs);xes3_kf=zeros(m,runs);

% Main Loop for Trial Runs
for jj = 1:runs,
    % Allocate Variables
    gam=5e-5;h=100000;mm=100000;
    x=zeros(m,3);x(1,1)=300000;x(1,2)=20000;x(1,3)=1e-3;
    ym=zeros(m,1);r=1e4;ym(1)=sqrt(mm^2+(x(1,1)-h)^2)+sqrt(r)*randn(1);

    % Initial Conditions
    pcov=diag([1e6 4e6 1e-4]);p=zeros(m,3);p(1,:)=diag(pcov)';
    xe=zeros(m,3);xe(1,1)=300000;xe(1,2)=20000;xe(1,3)=3e-5;

   % Kalman Filter Initial Conditions
    pcov_kf=diag([1e6 4e6 1e-4]);p_kf=zeros(m,3);p_kf(1,:)=diag(pcov_kf)';
    xe_kf=zeros(m,3);xe_kf(1,1)=300000;xe_kf(1,2)=20000;xe_kf(1,3)=3e-5;
```

Free Falling Body Example (iv)

```
% Unscented Filter Parameters
  alp=1;beta=2;kap=0;n=3;lam=alp^2*(n+kap)-n;
  w0m=lam/(n+lam);w0c=lam/(n+lam)+(1-alp^2+beta);
  wim=1/(2*(n+lam));yez=zeros(1,6);

  % Interval Set to 1/64 Seconds for Integration
  dt=1/64;tf=60;t=[0:dt:tf]';m=length(t);
  % Main Filter Loop
  for i=1:m-1
     % Truth
     f1=dt*athansfun(x(i,:)',gam);
     f2=dt*athansfun(x(i,:)'+0.5*f1,gam);
     f3=dt*athansfun(x(i,:)'+0.5*f2,gam);
     f4=dt*athansfun(x(i,:)'+f3,gam);
     x(i+1,:)=x(i,:)+1/6*(f1'+2*f2'+2*f3'+f4');
     % Measurement
     ym(i+1)=sqrt(mm^2+(x(i+1,1)-h)^2)+sqrt(r)*randn(1);
  end
  x = x(1:1/dt:end,:);
  ym = ym(1:1/dt:end);
```

Free Falling Body Example (v)

```
% Interval Set to 1 Second for Estimate Propagation and Update
  dt=1;tf=60;t=[0:dt:tf]';m=length(t);
  for i=1:m-1

    %%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Unscented Filter
    % Covariance Decomposition
    psquare=chol(pcov)';
    sigv=real([sqrt(n+lam)*psquare -sqrt(n+lam)*psquare]);
    xx0=xe(i,:)';
    xx=sigv+kron(xe(i,:)',ones(1,2*n));

    % Calculate Mean Through Propagation
    f1=dt*athansfun([xx0 xx],gam);
    f2=dt*athansfun([xx0 xx]+0.5*f1,gam);
    f3=dt*athansfun([xx0 xx]+0.5*f2,gam);
    f4=dt*athansfun([xx0 xx]+f3,gam);
    xx0=xx0+1/6*(f1(:,1)+2*f2(:,1)+2*f3(:,1)+f4(:,1));
    xx=xx+1/6*(f1(:,2:2*n+1)+2*f2(:,2:2*n+1)+2*f3(:,2:2*n+1)+f4(:,2:2*n+1));
    xe(i+1,:)=w0m*xx0'+wim*sum(xx,2)';
```

Free Falling Body Example (vi)

```
% Covariance
    pp0=w0c*(xx0-xe(i+1,:)')*(xx0-xe(i+1,:)')';
    pmat=xx-kron(xe(i+1,:)',ones(1,2*n));
    pcov=pp0+wim*pmat*pmat';

    % Output
    for j = 1:2*n
      yez(j)=sqrt(mm^2+(xx(1,j)-h)^2);
    end
    ye0=sqrt(mm^2+(xx0(1)-h)^2);
    ye=w0m*ye0+wim*sum(yez,2);

    % Calculate pyy
    pyy0=w0c*(ye0-ye)*(ye0-ye)';
    pyymat=yez-ye;
    pyy=pyy0+wim*pyymat*pyymat';

    % Calculate pxy
    pxy0=w0c*(xx0-xe(i+1,:)')*(ye0-ye);
    pxy=pxy0+wim*pmat*pyymat';
```

Free Falling Body Example (vii)

```
% Innovations Covarinace
    pvv=pyy+r;

    % Gain and Update
    gain=real(pxy*inv(pvv));
    pcov=pcov-gain*pvv*gain';
    p(i+1,:)=diag(pcov)';
    xe(i+1,:)=xe(i+1,:)+(gain*(ym(i+1)-ye))';

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Kalman Filter - Sometimes This Diverges
    % State Propagation
    f1=dt*athansfun(xe_kf(i,:)',gam);
    f2=dt*athansfun(xe_kf(i,:)'+0.5*f1,gam);
    f3=dt*athansfun(xe_kf(i,:)'+0.5*f2,gam);
    f4=dt*athansfun(xe_kf(i,:)'+f3,gam);
    xe_kf(i+1,:)=xe_kf(i,:)+1/6*(f1'+2*f2'+2*f3'+f4');

    % Estimate Output
    ye_kf=sqrt(mm^2+(xe_kf(i+1,1)-h)^2);
```

Free Falling Body Example (viii)

```
% Covariance Propagation
    f=exp(-gam*xe_kf(i,1))*[0 -exp(gam*xe_kf(i,1)) 0
      gam*xe_kf(i,2)^2*xe_kf(i,3) -2*xe_kf(i,2)*xe_kf(i,3) -xe_kf(i,2)^2
      0 0 0];
    phi=c2d(f,zeros(3,1),dt);
    pcov_kf=phi*pcov_kf*phi';

    % Update
    h_kf=[(xe_kf(i+1,1)-h)/sqrt(mm^2+(xe_kf(i+1,1)-h)^2) 0 0];
    gain_kf=pcov_kf*h_kf'*inv(h_kf*pcov_kf*h_kf'+r);
    pcov_kf=(eye(3)-gain_kf*h_kf)*pcov_kf;
    p_kf(i+1,:)=diag(pcov_kf)';
    xe_kf(i+1,:)=xe_kf(i+1,:)+(gain_kf*(ym(i+1)-ye_kf))';

  end
```

Free Falling Body Example (ix)

```
% Error for Each Trial Run
   xes1(:,jj)=abs(xe(:,1)-x(:,1));
   xes2(:,jj)=abs(xe(:,2)-x(:,2));
   xes3(:,jj)=abs(xe(:,3)-x(:,3));

   xes1_kf(:,jj)=abs(xe_kf(:,1)-x(:,1));
   xes2_kf(:,jj)=abs(xe_kf(:,2)-x(:,2));
   xes3_kf(:,jj)=abs(xe_kf(:,3)-x(:,3));

end

% Average and 3-Sigma Outlier
xerr=[sum(xes1,2) sum(xes2,2) sum(xes3,2)]/runs;
sig3=p.^(0.5)*3;
xerr_kf=[sum(xes1_kf,2) sum(xes2_kf,2) sum(xes3_kf,2)]/runs;
sig3_kf=p_kf.^(0.5)*3;
```

Free Falling Body Example (x)

```
% Plot Results
plot(t,xerr(:,1),t,xerr_kf(:,1),'--')
set(gca,'Fontsize',12)
ylabel('Absolute Error of Average Altitude Error (M)')
xlabel('Time (Sec)')
legend('Unscented Filter','Extended Kalman Filter')
```

Free Falling Body Example (xi)

```
function f=athansfun(x,gam)

% Function Routine for Athans Problem
[m,n]=size(x);
f=zeros(m,n);
f(1,:)=-x(2,:);
f(2,:)=-exp(-gam*x(1,:)).*(x(2,:).^2).*x(3,:);
f(3,:)=zeros(1,n);
```