Optimal Estimation Methods

(Lecture 21 – Target Tracking & Orbit Determination)

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

Target Tracking

Target Tracking

- Early-day application of the Kalman filter
  - Tracking of aircraft from radar observations
    - Actually done before the Kalman filter was developed (using the Wiener filter), but its structure is well explained using the Kalman filter equations
  - Two main purposes
    - The first involves actual filtering of the radar measurements to obtain accurate range estimates
    - The second involves the estimation of velocity (and possibly acceleration)
  - Need for accurate velocity estimation
    - Predict ahead of time where multiple targets are expected in future radar scans in order to make a correct association of each target
    - A 3σ bound from the error covariance can be used to access the validity of the radar scan at future times
    - Used to ensure that the same target is actually tracked, thus avoiding incorrect target associations of multiple vehicles

α–β Filter (i)

- One of the simplest target trackers is the α–β filter
  - Used to estimate the position and velocity (usually range and range rate) of a vehicle
  - Truth model in continuous time is given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t), \quad \mathbf{x} \equiv \begin{bmatrix} r \\ \dot{r} \end{bmatrix} \tag{1}$$

    where $w(t)$ is the process noise with spectral density q, and the states are position and velocity, denoted by r and $\dot{r}$
  - Note that the first state does not contain any process noise in this formulation
    - This is due to the fact that this state represents a kinematic relationship that is valid in theory and in the real world, since velocity is always the derivative of position
  - Discrete time measurements are assumed

$$\tilde{y}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + v_k \equiv H\mathbf{x}_k + v_k, \quad v_k \sim N(0, \sigma_n^2)$$

α–β Filter (ii)

- Since $F^2 = 0$, the state transition matrix is given by

$$\Phi = I + \Delta t\, F = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

  where Δt is the sampling interval
- The discrete-time process noise covariance is given by

$$\Upsilon\, Q\, \Upsilon^T = \int_0^{\Delta t} \Phi(\tau)\, G\, Q G^T \Phi^T(\tau)\; d\tau$$

  where $G = [0 \;\; 1]^T$
- Carrying out the integral gives

$$\begin{aligned} \Upsilon\, Q\, \Upsilon^T &= q \int_0^{\Delta t} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \tau & 1 \end{bmatrix} \, d\tau \\ &= q \int_0^{\Delta t} \begin{bmatrix} \tau^2 & \tau \\ \tau & 1 \end{bmatrix} \, d\tau \\ &= q \begin{bmatrix} \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^2/2 & \Delta t \end{bmatrix} \qquad (2) \end{aligned}$$

α–β Filter (iii)

- Notice, unlike the continuous-time process noise term given by $q\,G\,G^T$, the discrete-time process noise has nonzero values in all elements
  - This is due to the effect of sampling of a continuous-time process
  - However, if Δt is small, then a first order a good approximation is given by (as stated in the derivation of the discrete-time KF)

$$\Upsilon\,Q\,\Upsilon^T \approx \Delta t\,q\,G\,G^T = q\begin{bmatrix}0 & 0\\ 0 & \Delta t\end{bmatrix}$$

- The discrete-time Kalman filter equations reduce down to

$$\boxed{\begin{aligned}\hat{r}_k^+ &= \hat{r}_k^- + \alpha\,[\tilde{y}_k - \hat{r}_k^-]\\ \dot{\hat{r}}_k^+ &= \dot{\hat{r}}_k^- + \frac{\beta}{\Delta t}\,[\tilde{y}_k - \hat{r}_k^-]\\ \hat{r}_{k+1}^- &= \hat{r}_k^+ + \dot{\hat{r}}_k^+\,\Delta t\\ \dot{\hat{r}}_{k+1}^- &= \dot{\hat{r}}_k^+\end{aligned}}$$

where the gain is given by $K_k = K \equiv [\alpha \;\; \beta/\Delta t]^T$

α–β Filter (iv)

- The gains α and β are often treated as tuning parameters to enhance the tracking performance
  - However, conventional wisdom tells us that tuning these gains individually is incorrect
  - To understand this concept we must remember that the model in Eq. (1) shows a kinematic relationship
  - If α and β are chosen separately, then this kinematic relationship can be lost
  - This means the velocity estimate may not truly be the derivative of the position estimate, even though we know that this relationship is exact
  - A more true-to-physics approach involves tuning the continuous-time process noise parameter q
- From Eq. (2) changes in the velocity over the sampling interval are of the order $\sqrt{q\Delta t}$, which can be used as a guideline in the choice of q

α–β Filter (v)

- Define the following propagated and updated error-covariances

$$P^- \equiv \begin{bmatrix} p_{rr}^- & p_{r\dot{r}}^- \\ p_{r\dot{r}}^- & p_{\dot{r}\dot{r}}^- \end{bmatrix}, \quad P^+ \equiv \begin{bmatrix} p_{rr}^+ & p_{r\dot{r}}^+ \\ p_{r\dot{r}}^+ & p_{\dot{r}\dot{r}}^+ \end{bmatrix}$$

- Next, define the following variable

$$\boxed{S_q = q^{1/2}\,\Delta t^{3/2} \Big/ \sigma_n}$$

- The propagated error-covariance elements are given by

$$\boxed{\begin{aligned} p_{rr}^- &= \sigma_n^2 \left[\left(\frac{\xi}{S_q} \right)^2 - 1 \right] \\ p_{\dot{r}\dot{r}}^- &= \left(\frac{\sigma_n}{\Delta t} \right)^2 \left[S_q^2 \left(\frac{1}{2} - \frac{1}{\xi} \right) + \xi \right] \\ p_{r\dot{r}}^- &= \frac{\sigma_n^2 \xi}{\Delta t} \end{aligned}} \qquad (3)$$

α–β Filter (vi)

where

$$\xi = \frac{1}{2}\left[\left(\frac{S_q^2}{2}+\vartheta\right)+\sqrt{\left(\frac{S_q^2}{2}+\vartheta\right)^2-4S_q^2}\,\right]$$

$$\vartheta = \left[4S_q^2+\frac{S_q^4}{12}\right]^{1/2}$$

- The Kalman gain is given by

$$K \equiv \begin{bmatrix}\alpha \\ \beta/\Delta t\end{bmatrix} = P^- H^T (H\,P^- H^T + R)^{-1} = \frac{1}{p_{rr}^- + \sigma_n^2}\begin{bmatrix}p_{rr}^- \\ p_{r\dot{r}}^-\end{bmatrix} \qquad (4)$$

- This clearly shows that α and β are closely related to one another

α–β Filter (vii)

- To determine this relationship, first will show the relationship between p_{rr}^- and $p_{r\dot{r}}^-$
- Substituting $\xi = \Delta t\, p_{r\dot{r}}^- / \sigma_n^2$ into Eq. (3) and solving the resulting equation for $p_{r\dot{r}}^-$ yields

$$p_{r\dot{r}}^- = \frac{\sigma_n S_q}{\Delta t}\sqrt{p_{rr}^- + \sigma_n^2} \qquad (5)$$

- Solving for p_{rr}^- from the definition of α in Eq. (4) gives

$$p_{rr}^- = \frac{\sigma_n^2 \alpha}{1-\alpha} \qquad (6)$$

- Solving for $p_{r\dot{r}}^-$ from the definition of β in Eq. (4) gives

$$p_{r\dot{r}}^- = \frac{\beta\left(p_{rr}^- + \sigma_n^2\right)}{\Delta t} \qquad (7)$$

- Substituting Eq. (6) into Eq. (7) yields

$$p_{r\dot{r}}^- = \frac{\sigma_n^2 \beta}{\Delta t\,(1-\alpha)} \qquad (8)$$

α–β Filter (viii)

- Substituting Eqs. (6) and (8) into Eq. (5) leads to

$$\frac{\sigma_n^2 \beta}{\Delta t\,(1-\alpha)} = \frac{\sigma_n S_q}{\Delta t}\sqrt{\frac{\sigma_n^2 \alpha}{(1-\alpha)} + \sigma_n^2}$$

$$\frac{\sigma_n^4 \beta^2}{\Delta t^2\,(1-\alpha)^2} = \frac{\sigma_n^2 S_q^2}{\Delta t^2}\left[\frac{\sigma_n^2 \alpha}{(1-\alpha)} + \sigma_n^2\right]$$

$$\frac{\sigma_n^2 \beta^2}{(1-\alpha)^2} = S_q^2\left[\frac{\sigma_n^2 \alpha + \sigma_n^2 (1-\alpha)}{(1-\alpha)}\right]$$

$$\frac{\sigma_n^2 \beta^2}{(1-\alpha)^2} = S_q^2\left[\frac{\sigma_n^2}{(1-\alpha)}\right]$$

- Hence

$$\boxed{\frac{\beta^2}{1-\alpha} = S_q^2} \qquad (9)$$

- The quantity S_q is known as the tracking index, since it is proportional to the ratio of the process noise standard deviation and the measurement noise standard deviation

α–β Filter (ix)

- Kalata's Model

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \Delta t^2/2 \\ \Delta t \end{bmatrix} w_k$$

  - Leads to $S_q = q^{1/2}\Delta t^{1/2}/\sigma_n$
  - Slightly different than the previously shown one (uses $\Delta t^{1/2}$ instead of $\Delta t^{3/2}$)
  - This model assumes that the target undergoes a constant acceleration during the sampling interval and that the accelerations from period to period are independent
    - This model may ignore the kinematic relationship shown by the continuous-time model, and thus is not consistent kinematically

- Gain

$$K \equiv \begin{bmatrix} k_1 \\ \\ k_2 \end{bmatrix} = P^+ H^T R^{-1} = \sigma_n^{-2} \begin{bmatrix} p_{rr}^+ \\ \\ p_{r\dot{r}}^+ \end{bmatrix}$$

where $k_1 = \alpha$ and $k_2 = \beta/\Delta t$

α–β Filter (x)

- Updated covariance is given by

$$p_{rr}^{+} = \sigma_n^2 \left[1 - \left(\frac{S_q}{\xi}\right)^2\right]$$

$$p_{\dot{r}\dot{r}}^{+} = \left(\frac{\sigma_n}{\Delta t}\right)^2 \left[\xi - S_q^2 \left(\frac{1}{\xi} + \frac{1}{2}\right)\right]$$

- Equating the first equation above to $k_1 = \alpha$ gives

$$\boxed{\alpha = 1 - \left(\frac{S_q}{\xi}\right)^2}$$

- Using Eq. (9) directly gives

$$\boxed{\beta = S_q\sqrt{1 - \alpha}}$$

α–β Filter (xi)

- A direct relationship between α and β is possible
  - Substituting $H = [1 \;\; 0]$ into $P^+ = (I - KH)P^-$ gives

$$\begin{bmatrix} p^+_{rr} & p^+_{r\dot{r}} \\ p^+_{r\dot{r}} & p^+_{\dot{r}\dot{r}} \end{bmatrix} = \begin{bmatrix} p^-_{rr}(1-k_1) & p^-_{r\dot{r}}(1-k_1) \\ p^-_{r\dot{r}} - k_2 p^-_{rr} & p^-_{\dot{r}\dot{r}} - k_2 p^-_{r\dot{r}} \end{bmatrix} \qquad (10)$$

  - This must be symmetric so that

$$k_1 = \left(\frac{p^-_{rr}}{p^-_{r\dot{r}}}\right) k_2 \qquad (11)$$

  - Substituting quantities into $P^- = \Phi P^+ \Phi^T + \Upsilon Q \Upsilon^T$ gives

$$\begin{bmatrix} p^-_{rr} & p^-_{r\dot{r}} \\ p^-_{r\dot{r}} & p^-_{\dot{r}\dot{r}} \end{bmatrix} = \begin{bmatrix} p^+_{rr} + 2p^+_{r\dot{r}}\Delta t + p^+_{\dot{r}\dot{r}}\Delta t^2 & p^+_{r\dot{r}} + p^+_{\dot{r}\dot{r}}\Delta t \\ p^+_{r\dot{r}} + p^+_{\dot{r}\dot{r}}\Delta t & p^+_{\dot{r}\dot{r}} \end{bmatrix} + q \begin{bmatrix} \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^2/2 & \Delta t \end{bmatrix}$$

(12)

α–β Filter (xii)

- From Eqs. (10) and (12) the 2-2 element gives

$$k_2 = \frac{q\,\Delta t}{p^-_{r\dot{r}}} \tag{13}$$

- Solving Eq. (11) for p^-_{rr} and using Eq. (13) gives

$$p^-_{rr} = \frac{k_1\,q\,\Delta t}{k_2^2} \tag{14}$$

- From Eqs. (10) and (12) the 1-2 element gives

$$p^-_{\dot{r}\dot{r}} = p^-_{r\dot{r}}\left(\frac{k_1}{\Delta t} + k_2\right) - \frac{q\,\Delta t}{2} \tag{15}$$

- From Eqs. (10) and (12) the 1-1 element, with substitution of Eq. (15), leads to

$$p^-_{rr}k_1 + p^-_{r\dot{r}}\Delta t(k_1 - 2) + \frac{q\,\Delta t^3}{6} = 0 \tag{16}$$

α–β Filter (xiii)

- Solving Eq. (13) for $p^-_{r\dot{r}}$, and substituting the resulting equation and Eq. (14), into Eq. (16) yields

$$k_1^2 \Delta t + k_2 \Delta t^2 (k_1 - 2) + \frac{k_2^2 \Delta t^3}{6} = 0$$

- From $k_1 = \alpha$ and $k_2 = \beta / \Delta t$, this equation reduces down to

$$\alpha^2 + \beta(\alpha - 2) + \frac{\beta^2}{6} = 0$$

- Since β is always positive, which will be proven in the stability analysis, then α and β are related by

$$\boxed{\alpha = -\frac{1}{2}\beta + \frac{1}{2}\sqrt{\beta\left[(\beta/3) + 8\right]}}$$

- This equation clearly shows the relationship between α and β, which can be written without S_q directly

α–β Filter Stability (i)

- Investigate discrete-time stability requirements
  - The eigenvalues of $\Phi(I - K\,H)$ can be found using

$$|zI - \Phi[I - K\,H]| = \det \begin{bmatrix} z + \alpha + \beta - 1 & -\Delta t \\ \beta/\Delta t & z - 1 \end{bmatrix} = 0$$

  - This gives the following characteristic equation

$$z^2 + (\alpha + \beta - 2)z + (1 - \alpha) = 0$$

  - All eigenvalues must lie within the unit circle for a stable system
    - Even though the characteristic equation is second-order in nature, using the unit circle condition directly to prove stability is arduous
    - Jury's test can be used to easily derive the stability conditions for α and β

α–β Filter Stability (ii)

- Consider the following second-order polynomial

$$z^2 + a_1 z + a_2 = 0$$

where $a_1 \equiv \alpha + \beta - 2$ and $a_2 \equiv 1 - \alpha$

- Jury's test for stability for this second-order equation involves satisfying the following three conditions

$$a_2 < 1$$
$$a_2 > a_1 - 1$$
$$a_2 > -(a_1 + 1)$$

- From the definitions of a_1 and a_2, these conditions give $\alpha > 0$, $\beta > 0$, and $2\alpha + \beta < 4$
- From the equation $\alpha = 1 - (S_q/\xi)^2$, since $\alpha > 0$ and $(S_q/\xi)^2 > 0$, then the following conditions must be satisfied for stability

$$0 < \alpha \leq 1$$
$$0 < \beta < 2$$

α–β Filter Stability (iii)

- The stability conditions are valid even if α and β are chosen independently
- If q is tuned to determine α and β, then from the following equations

$$\frac{\beta^2}{1-\alpha} = S_q^2$$

$$\alpha = -\frac{1}{2}\beta + \frac{1}{2}\sqrt{\beta\left[(\beta/3)+8\right]}$$

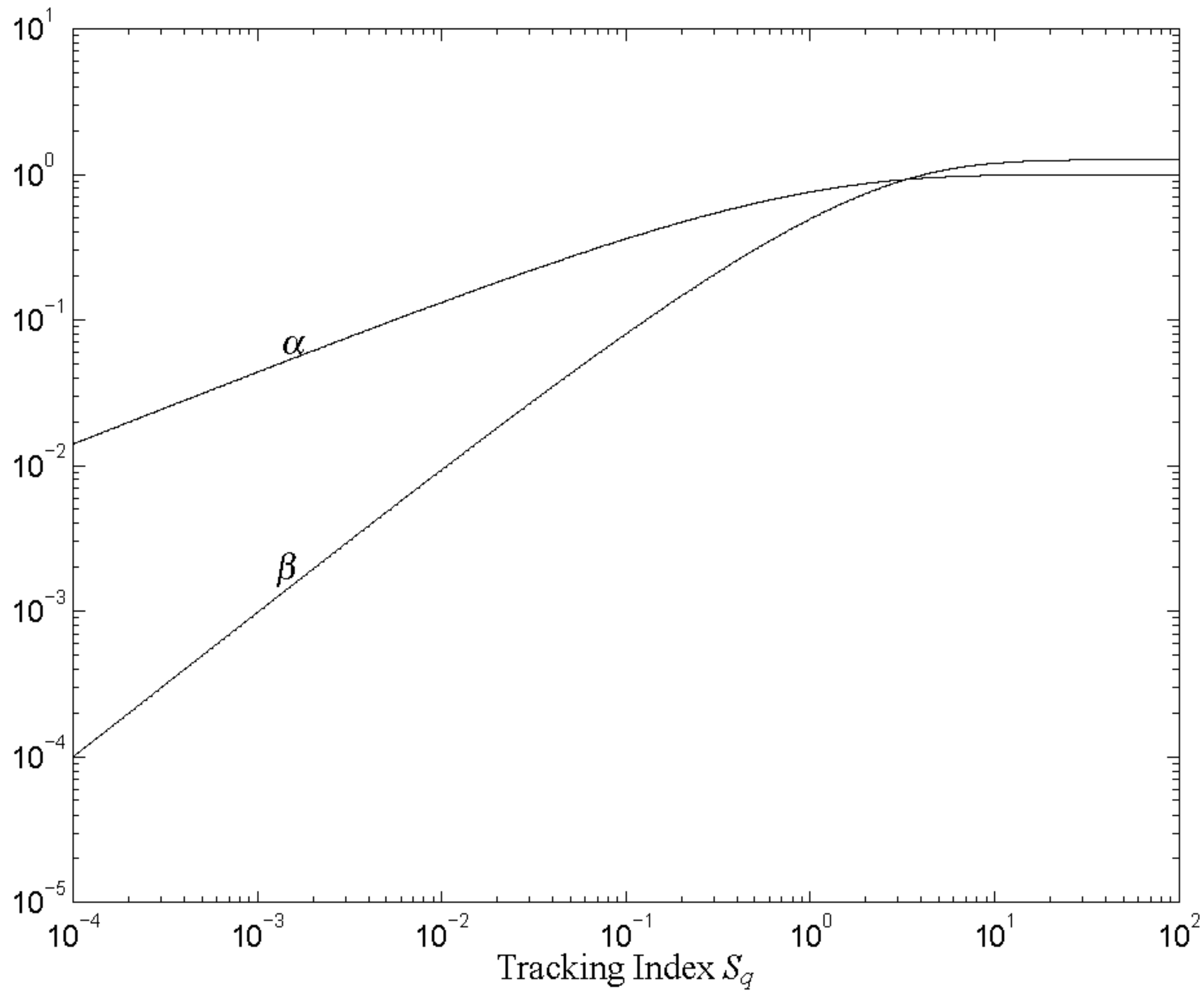
the asymptotic limits are given by

$$\alpha = 1$$

$$\beta = 3 - \sqrt{3} = 1.2679$$

- These limits are clearly within the upper bounds
- Always best to tune q then to tune α and β in order to retain the kinematic relationship (although this is not done in practice usually)

Gains vs. Tracking Index



α–β –γ Filter (i)

- Higher-order filter with truth model given by

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w(t)$$

  - Measurement model given by

$$\tilde{y}_k = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k + v_k \equiv H\mathbf{x}_k + v_k$$

  - The state transition matrix and covariance are given by

$$\Phi = I + \Delta t\, F + \frac{\Delta t^2}{2} F^2 = \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Upsilon\, Q\, \Upsilon^T = q \begin{bmatrix} \Delta t^5/20 & \Delta t^4/8 & \Delta t^3/6 \\ \Delta t^4/8 & \Delta t^3/3 & \Delta t^2/2 \\ \Delta t^3/6 & \Delta t^2/2 & \Delta t \end{bmatrix}$$

α–β –γ Filter (ii)

- Kalman update and propagation equations are given by

$$\boxed{\begin{aligned}\hat{r}_k^+ &= \hat{r}_k^- + \alpha\left[\tilde{y}_k - \hat{r}_k^-\right]\\ \dot{\hat{r}}_k^+ &= \dot{\hat{r}}_k^- + \frac{\beta}{\Delta t}\left[\tilde{y}_k - \hat{r}_k^-\right]\\ \ddot{\hat{r}}_k^+ &= \ddot{\hat{r}}_k^- + \frac{\gamma}{2\Delta t^2}\left[\tilde{y}_k - \hat{r}_k^-\right]\\ \hat{r}_{k+1}^- &= \hat{r}_k^+ + \dot{\hat{r}}_k^+\Delta t + \frac{1}{2}\ddot{\hat{r}}_k^+\Delta t^2\\ \dot{\hat{r}}_{k+1}^- &= \dot{\hat{r}}_k^+ + \ddot{\hat{r}}_k^+\Delta t\\ \ddot{\hat{r}}_{k+1}^- &= \ddot{\hat{r}}_k^+\end{aligned}}$$

where the gain is given by $K_k = K \equiv \left[\alpha \quad \beta/\Delta t \quad \gamma/(2\Delta t^2)\right]^T$

- Changes in the acceleration over the sampling interval are of the order $\sqrt{q\Delta t}$, which can be used as a guideline in the choice of q
  - Can clearly see how this filter can produce better estimates
  - But requires initial acceleration, which may not be known accurately

α–β –γ Filter Stability

- Characteristic equation found by

$$|zI - \Phi[I - K\,H]| = \det \begin{bmatrix} z + \alpha + \beta + \frac{1}{4}\gamma - 1 & -\Delta t & -\frac{1}{2}\Delta t^2 \\ \frac{1}{2\,\Delta t}(2\beta + \gamma) & z - 1 & -\Delta t \\ \frac{1}{2\,\Delta t^2}\gamma & 0 & z - 1 \end{bmatrix} = 0$$

which leads to

$$z^3 + (\alpha + \beta + \frac{1}{4}\gamma - 3)z^2 + (3 - 2\alpha - \beta + \frac{1}{4}\gamma)z + (\alpha - 1) = 0$$

- Jury's test leads to the following stability conditions

$$0 < \alpha \leq 1$$

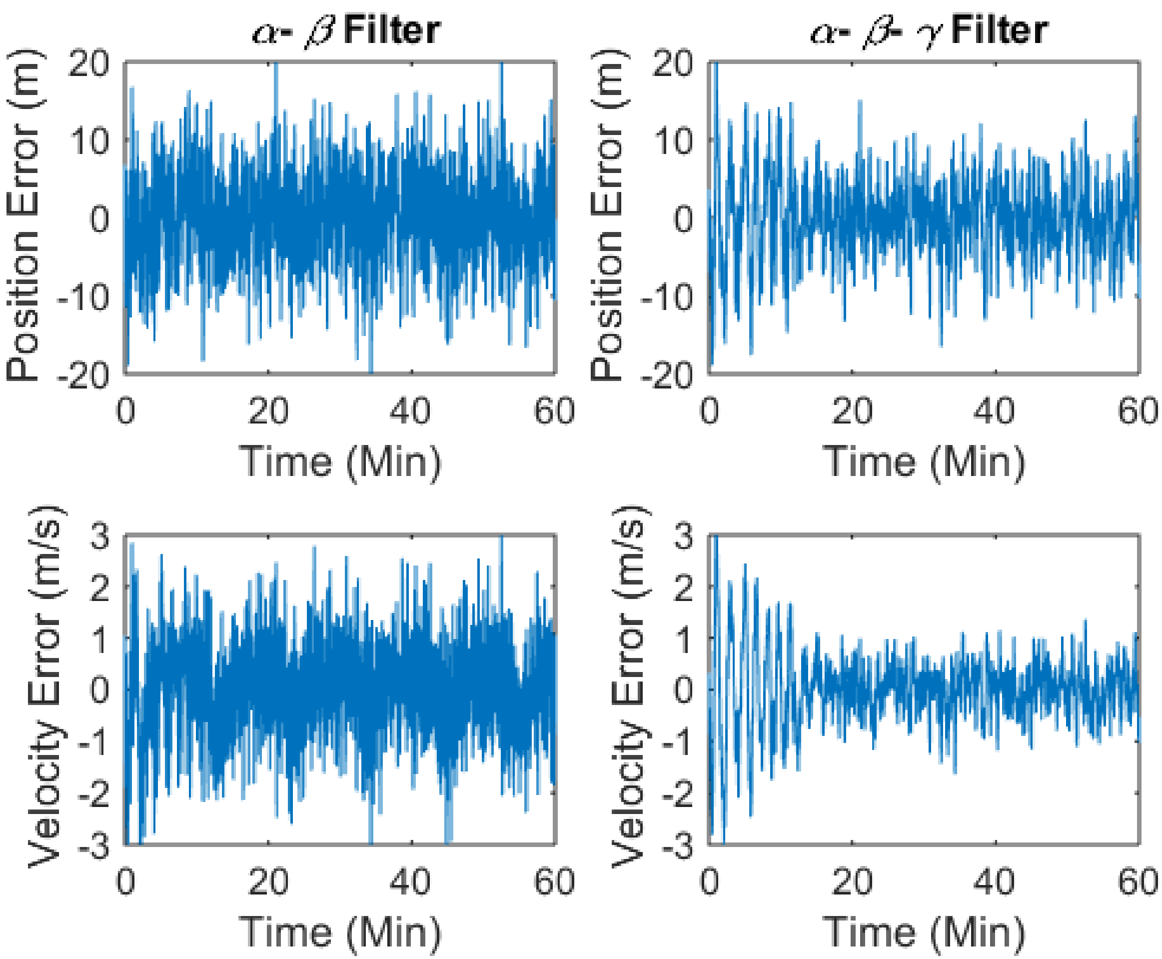
$$0 < \beta < 2$$

$$0 < \gamma < \frac{4\alpha\beta}{2 - \alpha}$$

Example (i)

- Track the vertical position of an aircraft
  - The vertical position has a standard deviation of 10 m for the measurement error
  - Measurements are sampled at 1-second intervals
  - Since the truth is known, then the variance parameter q in both the filters is tuned to ensure the best possible performance
    - This parameter is varied until transients begin to appear in the position errors
  - For the α–β filter the optimal parameter is given by $q = 0.5$
    - Gives $\alpha = 0.31344$ and $\beta = 0.05859$
  - For the α–β –γ filter the optimal parameter is given by $q = 0.0001$
    - Much smaller than α–β filter; affects changes in acceleration, which is smaller in magnitude than changes in velocity
    - Gives $\alpha = 0.18127$ and $\beta = 0.01811$, and $\gamma = 0.00181$

Example (ii)



The $\alpha-\beta-\gamma$ filter gives better estimates, especially for velocity, because it is a higher-order filter than the $\alpha-\beta$ filter

Code (i)

```
% Aircraft Data
cd0=0.0164;cda=0.20;cdde=0;
cy0=0;cyb=-0.9;cydr=0.120;cyda=0;
cl0=0.21;cla=4.4;clde=0.32;
cll0=0;cllb=-.160;clldr=0.008;cllda=0.013;
cm0=0;cma=-1.00;cmde=-1.30;
cn0=0;cnb=0.160;cndr=-0.100;cnda=0.0018;
cmq=-20.5;cllp=-0.340;cllr=0.130;cnp=-0.026;cnr=-0.280;

% Aircraft Data in SI Units
rho=.6536033;s=510.97;cbar=8.321;l=59.74;mass=2831897.6/9.81;
in=diag([24675882 44877565 67384138]);in(1,3)=1315143.1;in(3,1)=in(1,3);
g=9.81;

coef=[cd0;cda;cdde;cy0;cyb;cydr;cyda;cl0;cla;clde;cll0;cllb;clldr;cllda;cm0;cma;cmde;cn0
cnb;cndr;cnda;cmq;cllp;cllr;cnp;cnr];
other=[rho;s;cbar;l;mass;g];
```

Code (ii)

```
% Initial Conditions
w10=0;w20=0;w30=0;
xx0=0;yy0=0;zz0=6096;
vmag=205.13;

% Trim Conditions
qtrim=0.5*rho*vmag^2;
dtrim=cla*cmde-cma*clde;
alptrim=((mass*g/qtrim/s-cl0)*cmde+cm0*clde)/dtrim;
detrim=(-cla*cm0-cma*(mass*g/qtrim/s-cl0))/dtrim;
dragtrim=(cd0+cda*alptrim+cdde*detrim)*qtrim*s;
v10=sqrt(vmag^2/(1+tan(alptrim)^2));v20=0;v30=v10*tan(alptrim);

% Initial Angles
phi0=0;theta0=0*alptrim;psi0=0;

% Steady-State Values
w1_ss=w10;w2_ss=w20;w3_ss=w30;
v_ss=vmag;
```

Code (iii)

```
% True States
dt=1;t=[0:dt:3600]';m=length(t);
x=zeros(m,12);
x(1,1:3)=[v10 v20 v30];
x(1,4:6)=[w10 w20 w30];
x(1,7:9)=[xx0 yy0 zz0];
x(1,10:12)=[phi0 theta0 psi0];

% Control Surface Inputs and Thrust
de=detrim*ones(m,1)+1*pi/180*sin(0.01*t);dr=0*ones(m,1);da=0*ones(m,1);
thrust=dragtrim*ones(m,1);

% Main Loop for Aircraft Simulation
for i=1:m-1,
 f1=dt*air_fun(x(i,:),de(i),dr(i),da(i),thrust(i),coef,other,in,w1_ss,w2_ss,w3_ss,v_ss);
 f2=dt*air_fun(x(i,:)+0.5*f1',de(i),dr(i),da(i),thrust(i),coef,other,in,w1_ss,w2_ss,w3_ss,v_ss);
 f3=dt*air_fun(x(i,:)+0.5*f2',de(i),dr(i),da(i),thrust(i),coef,other,in,w1_ss,w2_ss,w3_ss,v_ss);
 f4=dt*air_fun(x(i,:)+f3',de(i),dr(i),da(i),thrust(i),coef,other,in,w1_ss,w2_ss,w3_ss,v_ss);
 x(i+1,:)=x(i,:)+1/6*(f1'+2*f2'+2*f3'+f4');
end;
```

Code (iv)

```
% Velocity, Angle of Attack and Sideslip
vel=x(:,1:3);
velm=(vel(:,1).^2+vel(:,2).^2+vel(:,3).^2).^(0.5);
alp=atan(vel(:,3)./vel(:,1))*180/pi;
bet=asin(vel(:,2)./velm)*180/pi;

% Angles
w=x(:,4:6)*180/pi;
pos=x(:,7:9);
phi=x(:,10)*180/pi;theta=x(:,11)*180/pi;psi=x(:,12)*180/pi;

% Pre-Allocate Space
pos0=pos(1,3);vel0=vel(1,3);
m=length(pos);
pose=zeros(m,1);pose(1)=pos0;
vele=zeros(m,1);vele(1)=vel0;

% Noise Parameter
sigp=10;
ym=pos(:,3)+sigp*randn(m,1);
```

Code (v)

```
% Alpha-Beta Filter Variables
xe_2=zeros(m,2);xe_2(1,:)=[pos0 vel0];
phi=[1 dt;0 1];h=[1 0];

% Process Noise Tuning and Covariance
q=.5;
qd=q*[1/3*dt^3 0.5*dt^2;0.5*dt^2 dt];
pcov=dare(phi',h',qd,sigp^2,zeros(2,1),eye(2));
gain=pcov*h'*inv(h*pcov*h'+sigp^2);

sig3_alp_bet=diag(pcov).^(0.5)*3
disp(' ')

% Alpha-Beta Filter
for i = 1:m-1
 xe_2(i+1,:)=(phi*xe_2(i,:)'+phi*gain*(ym(i)-xe_2(i,1)))';
end
```

Code (vi)

```
% Alpha-Beta-Gamma Filter Variables
xe_3=zeros(m,3);xe_3(1,:)=[pos0 vel0 0];
phi=[1 dt dt^2/2;0 1 dt;0 0 1];h=[1 0 0];

% Process Noise Tuning and Covariance
q=.0001;
qd=q*[dt^5/20 dt^4/8 dt^3/6;dt^4/8 dt^3/3 dt^2/2;dt^3/6 dt^2/2 dt];
pcov=dare(phi',h',qd,sigp^2,zeros(3,1),eye(3));
gain=pcov*h'*inv(h*pcov*h'+sigp^2);

sig3_alp_bet_gam=diag(pcov).^(0.5)*3

% Alpha-Beta-Gamma Filter
for i = 1:m-1
 xe_3(i+1,:)=(phi*xe_3(i,:)'+phi*gain*(ym(i)-xe_3(i,1)))';
end

% Velocity
zvel=diff(pos(:,3))/dt;zvel(m)=zvel(m-1);
```

Code (vii)

```
% Plot Results
subplot(221)
plot(t/60,pos(:,3)-xe_2(:,1));
set(gca,'fontsize',12)
axis([0 60 -20 20]);
set(gca,'Ytick',[-20 -10 0 10 20]);
set(gca,'Xtick',[0 20 40 60]);
ylabel('Position Error (m)')
xlabel('Time (Min)')
title('{\it \alpha}-{\it \beta} Filter')

subplot(223)
plot(t/60,zvel-xe_2(:,2));
set(gca,'fontsize',12)
axis([0 60 -3 3]);
set(gca,'Ytick',[-3 -2 -1 0 1 2 3]);
set(gca,'Xtick',[0 20 40 60]);
ylabel('Velocity Error (m/s)')
xlabel('Time (Min)')
```

Code (viii)

```
subplot(222)
plot(t/60,pos(:,3)-xe_3(:,1));
set(gca,'fontsize',12)
axis([0 60 -20 20]);
set(gca,'Ytick',[-20 -10 0 10 20]);
set(gca,'Xtick',[0 20 40 60]);
ylabel('Position Error (m)')
xlabel('Time (Min)')
title('{\it \alpha}-{\it \beta}-{\it \gamma} Filter')

subplot(224)
plot(t/60,zvel-xe_3(:,2));
set(gca,'fontsize',12)
axis([0 60 -3 3]);
set(gca,'Ytick',[-3 -2 -1 0 1 2 3]);
set(gca,'Xtick',[0 20 40 60]);
ylabel('Velocity Error (m/s)')
xlabel('Time (Min)')
```

Code (ix)

```
function f=air_fun(x,de,dr,da,thrust,coef,other,in,w1_ss,w2_ss,w3_ss,v_ss);
% Function Routine for General Aircraft Equations

% Inertias
ixx=in(1,1);iyy=in(2,2);izz=in(3,3);ixz=in(1,3);

% Velocities and Euler Angles
v1=x(1);v2=x(2);v3=x(3);
w1=x(4);w2=x(5);w3=x(6);
phi=x(10);theta=x(11);psi=x(12);

% Get Other Constants
f=zeros(12,1);
m=other(5);
g=other(6);

% Speed, Angle of Attack and Sideslip
vm=norm([x(1);x(2);x(3)]);
alp=atan(x(3)/x(1));
bet=asin(x(2)/vm);
```

Code (x)

```
% Dynamic Pressure
q=0.5*other(1)*vm^2;

% General Coefficients
cd=coef(1)+coef(2)*alp+coef(3)*de;
cy=coef(4)+coef(5)*bet+coef(6)*dr+coef(7)*da;
cl=coef(8)+coef(9)*alp+coef(10)*de;

dd=2*vm^2;
cll=coef(11)+coef(12)*bet+coef(13)*dr+coef(14)*da+coef(23)*(w1-w1_ss)*other(4)/2/v_ss...
   +coef(24)*(w3-w3_ss)*other(4)/2/v_ss;
cm=coef(15)+coef(16)*alp+coef(17)*de+coef(22)*(w2-w2_ss)*other(3)/2/v_ss;
cn=coef(18)+coef(19)*bet+coef(20)*dr+coef(21)*da+coef(25)*(w1-w1_ss)*other(4)/2/v_ss...
   +coef(26)*w3*other(4)/2/v_ss;

% Drag, Force and Lift
drag=cd*q*other(2);
yforce=cy*q*other(2);
lift=cl*q*other(2);
```

Code (xi)

```
% Torques
la1=cll*q*other(2)*other(4);
la2=cm*q*other(2)*other(3);
la3=cn*q*other(2)*other(4);

% Forces
force1=-drag*cos(alp)+lift*sin(alp)+thrust*cos(alp);
force2=yforce;
force3=-drag*sin(alp)-lift*cos(alp)+thrust*sin(alp);

% Functions
f(1)=-g*sin(theta)+force1/m+v2*w3-v3*w2;
f(2)=g*cos(theta)*sin(phi)+force2/m+v3*w1-v1*w3;
f(3)=g*cos(theta)*cos(phi)+force3/m+v1*w2-v2*w1;

k1=ixz*(iyy-ixx);
k2=izz*(izz-iyy);
k3=ixz*(izz-iyy);
k4=ixx*(iyy-ixx);
```

Code (xii)

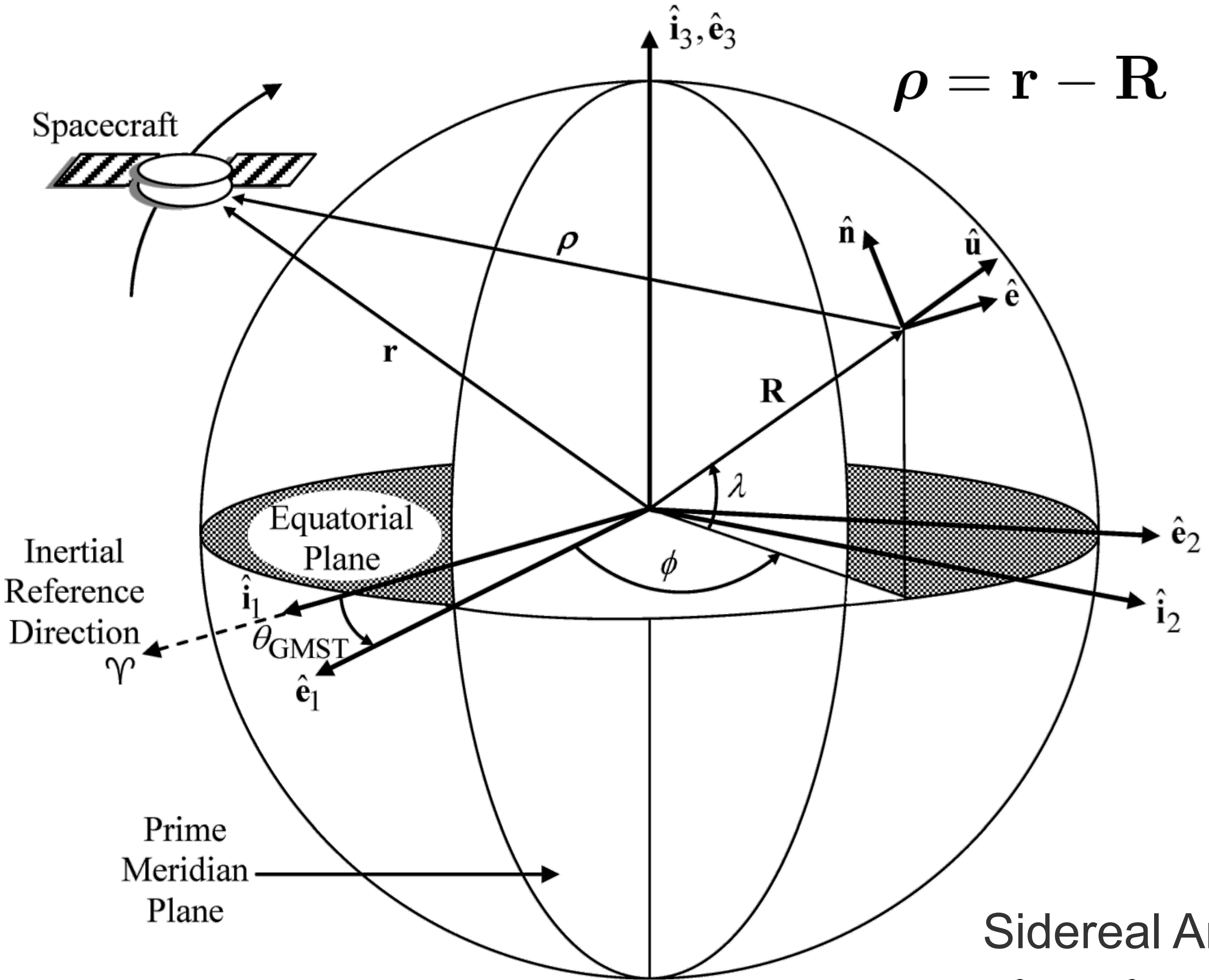
```
f(4)=(ixz*la3+izz*la1-k1*w1*w2-ixz^2*w2*w3+ixz*izz*w1*w2-k2*w2*w3)/(ixx*izz-ixz^2);
f(5)=(la2-(ixx-izz)*w1*w2-ixz*(w1^2-w3^2))/iyy;
f(6)=(ixx*la3+ixz*la1+ixz^2*w1*w2-k3*w2*w3-k4*w1*w2-ixx*ixz*w2*w3)/(izz*ixx-ixz^2);

f(7)=cos(theta)*cos(psi)*v1...
    +(sin(phi)*sin(theta)*cos(psi)-cos(phi)*sin(psi))*v2...
    +(cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi))*v3;
f(8)=cos(theta)*sin(psi)*v1...
    +(sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos(psi))*v2...
    +(cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi))*v3;
f(9)=-sin(theta)*v1+sin(phi)*cos(theta)*v2+cos(phi)*cos(theta)*v3;

f(10)=w1+sin(phi)*tan(theta)*w2+cos(phi)*tan(theta)*w3;
f(11)=cos(phi)*w2-sin(phi)*w3;
f(12)=sin(phi)*sec(theta)*w2+cos(phi)*sec(theta)*w3;
```

Orbit Estimation

Orbit Estimation

- Two main approaches
  - Least Squares
  - Kalman filter
- Least Squares Approach
  - Determination of the planetary objects from telescope/sextant observations was Gauss' motivation to invent least squares!
    - Still used today
    - Goal
      - Given a number of observations during some time interval, determine the orbit position and velocity at some epoch
    - Nonlinear least squares must be used
      - A differential correction is used since differential equations are employed for the orbit estimation
    - Use initial orbit determination methods to initialize the nonlinear least squares process

Earth Observations



$$\boldsymbol{\rho} = \mathbf{r} - \mathbf{R}$$

- λ latitude of observer
- ϕ longitude of observer
- θ_{GMST} sidereal angle (computed knowing time)

- $\mathbf{R}$ position of observer
- $\mathbf{r}$ position of spacecraft
- ρ slant range

Sidereal Angle of Sight

$$\theta = \theta_{\mathrm{GMST}} + \phi$$

Observations

- Slant range in inertial components is given by

$$\boldsymbol{\rho} = \begin{bmatrix} x - ||\mathbf{R}|| \cos\lambda \cos\theta \\ y - ||\mathbf{R}|| \cos\lambda \sin\theta \\ z - ||\mathbf{R}|| \sin\lambda \end{bmatrix}$$

- Conversion from inertial to observer (“up, east and north) system is given by

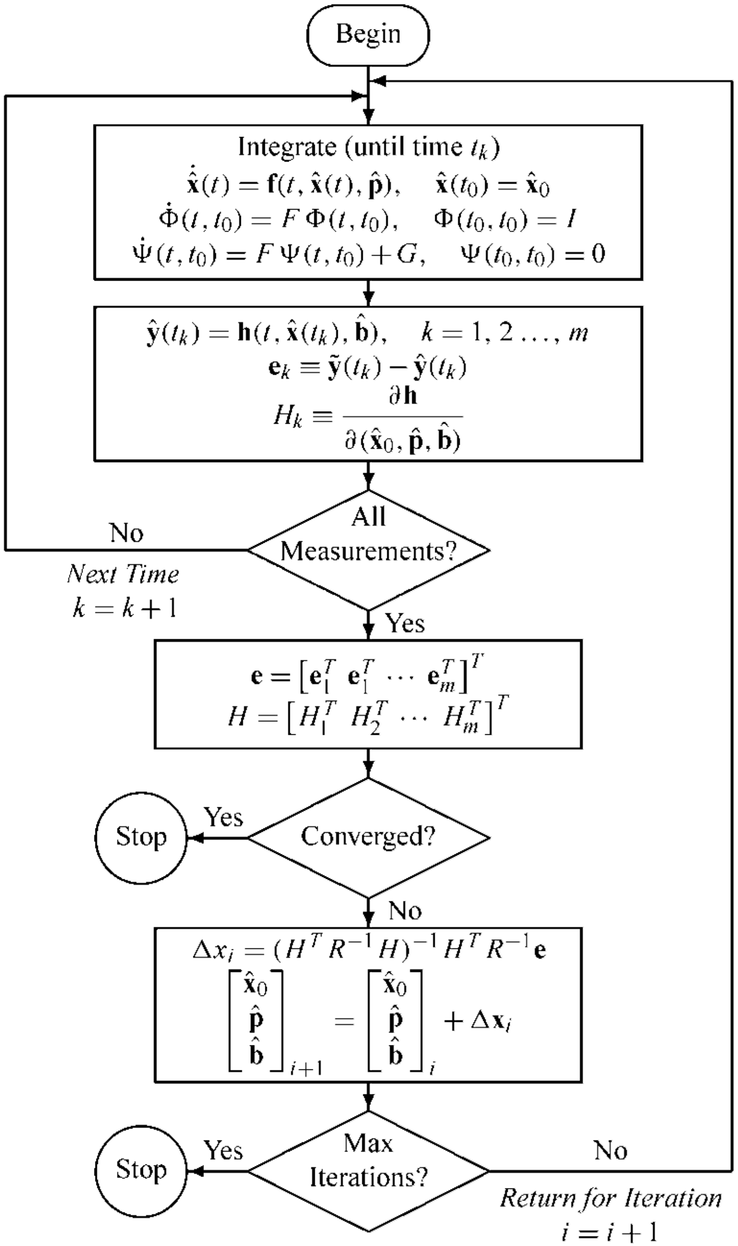
$$\begin{bmatrix} \rho_u \\ \rho_e \\ \rho_n \end{bmatrix} = \begin{bmatrix} \cos\lambda & 0 & \sin\lambda \\ 0 & 1 & 0 \\ -\sin\lambda & 0 & \cos\lambda \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{\rho}$$

- Assume radar with range, azimuth and elevation

$$||\boldsymbol{\rho}|| = (\rho_u^2 + \rho_e^2 + \rho_n^2)^{1/2}$$

$$\text{az} = \tan^{-1}\left(\frac{\rho_e}{\rho_n}\right), \quad \text{el} = \sin^{-1}\left(\frac{\rho_u}{||\boldsymbol{\rho}||}\right)$$

Least Squares Process



$$\boxed{\ddot{\mathbf{r}} = -\frac{\mu}{||\mathbf{r}||^3}\mathbf{r}, \quad \mathbf{r} = [x \;\; y \;\; z]^T}$$

The goal is to determine $\mathbf{x}_0 = \begin{bmatrix} \mathbf{r}_0^T & \dot{\mathbf{r}}_0^T \end{bmatrix}^T$

Also includes other parameters if desired, given by $\mathbf{p}$ (e.g., the parameter μ can also be determined if desired)

Other quantities, such as measurement biases or force model parameters, can be appended to the measurement observation equation through the vector $\mathbf{b}$

Example (i)

- Truth at epoch

$$\mathbf{r}_0 = \begin{bmatrix} 7,000 & 1,000 & 200 \end{bmatrix}^T \text{ km}, \quad \dot{\mathbf{r}}_0 = \begin{bmatrix} 4 & 7 & 2 \end{bmatrix}^T \text{ km/sec}$$

- Measurements
  - The latitude of the observer is given by $\lambda = 5°$
  - Initial sidereal time is given by $\theta_0 = 5°$
  - Measurements are given at 10-second intervals over a 100-second simulation
  - The measurement errors are zero-mean Gaussian with a standard deviation of the range measurement error given by 1 km, and a standard deviation of the angle measurements given by 0.01 degrees
- Herrick-Gibbs estimates and truth (second time-step)

$$\hat{\mathbf{r}} = \begin{bmatrix} 7,038 & 1,070 & 221 \end{bmatrix}^T \text{ km}, \quad \mathbf{r} = \begin{bmatrix} 7,040 & 1,070 & 220 \end{bmatrix}^T \text{ km}$$

$$\dot{\hat{\mathbf{r}}} = \begin{bmatrix} 3.92 & 7.00 & 2.00 \end{bmatrix}^T \text{ km/sec}, \quad \dot{\mathbf{r}} = \begin{bmatrix} 3.92 & 7.00 & 2.00 \end{bmatrix}^T \text{ km/sec}$$

Example (ii)

- Initialized least squares with

$$\hat{\mathbf{r}}_0 = \begin{bmatrix} 6,990 & 1 & 1 \end{bmatrix}^T \text{ km}, \quad \dot{\hat{\mathbf{r}}}_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \text{ km/sec}$$

  - Test robustness of the algorithm

Iteration	Position (km)			Velocity (km/sec)		
0	6,990	1	1	1	1	1
1	7,496	1,329	−178	5.30	6.20	−18.42
2	7,183	609	27	12.66	22.63	12.69
3	6,842	905	490	6.65	13.73	−8.15
4	6,795	963	255	9.33	7.38	1.36
5	6,985	989	199	4.24	7.20	1.89
6	7,000	1,000	200	4.00	7.00	2.00
7	7,000	1,000	200	4.00	7.00	2.00

$$3\boldsymbol{\sigma}_{\hat{\mathbf{r}}} = \begin{bmatrix} 1.26 & 0.25 & 0.51 \end{bmatrix}^T \text{ km}$$

$$3\boldsymbol{\sigma}_{\dot{\hat{\mathbf{r}}}} = \begin{bmatrix} 0.020 & 0.008 & 0.006 \end{bmatrix}^T \text{ km/sec}$$

Iterated Kalman Filter (i)

- Procedure
  - Use the EKF to process the data forward with some initial condition guess
  - Then process the data backward to epoch
  - Initial conditions for the state are then given by previous pass results
    - The backward pass uses the final state from the forward pass for its initial condition
  - The covariance must be reset after each forward or backward pass
    - This is required since no “new” information is given with each pass
  - The algorithm for orbit estimation is essentially equivalent to the nonlinear fixed-point smoother with a covariance reset
  - Results show much better convergence properties than a nonlinear least squares approach

Iterated Kalman Filter (ii)

- Partials
  - State matrix is given by

$$F = \begin{bmatrix} 0_{3\times 3} & I_{3\times 3} \\ F_{21} & 0_{3\times 3} \end{bmatrix}$$

where

$$F_{21} = \begin{bmatrix} \frac{3\mu x^2}{||\mathbf{r}||^5} - \frac{\mu}{||\mathbf{r}||^3} & \frac{3\mu xy}{||\mathbf{r}||^5} & \frac{3\mu xz}{||\mathbf{r}||^5} \\ \frac{3\mu xy}{||\mathbf{r}||^5} & \frac{3\mu y^2}{||\mathbf{r}||^5} - \frac{\mu}{||\mathbf{r}||^3} & \frac{3\mu yz}{||\mathbf{r}||^5} \\ \frac{3\mu xz}{||\mathbf{r}||^5} & \frac{3\mu yz}{||\mathbf{r}||^5} & \frac{3\mu z^2}{||\mathbf{r}||^5} - \frac{\mu}{||\mathbf{r}||^3} \end{bmatrix}$$

Iterated Kalman Filter (iii)

- Output matrix

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} H_{11} & 0_{3\times 3} \end{bmatrix}$$

where

$$H_{11} = \begin{bmatrix} \dfrac{\partial ||\boldsymbol{\rho}||}{\partial x} & \dfrac{\partial ||\boldsymbol{\rho}||}{\partial y} & \dfrac{\partial ||\boldsymbol{\rho}||}{\partial z} \\[2ex] \dfrac{\partial \mathrm{az}}{\partial x} & \dfrac{\partial \mathrm{az}}{\partial y} & \dfrac{\partial \mathrm{az}}{\partial z} \\[2ex] \dfrac{\partial \mathrm{el}}{\partial x} & \dfrac{\partial \mathrm{el}}{\partial y} & \dfrac{\partial \mathrm{el}}{\partial z} \end{bmatrix}$$

Iterated Kalman Filter (iv)

$$\frac{\partial||\boldsymbol{\rho}||}{\partial x} = (\rho_u \cos\phi\cos\Theta - \rho_e \sin\Theta - \rho_n \sin\phi\cos\Theta)/||\boldsymbol{\rho}||$$
$$\frac{\partial||\boldsymbol{\rho}||}{\partial y} = (\rho_u \cos\phi\sin\Theta + \rho_e \cos\Theta - \rho_n \sin\phi\sin\Theta)/||\boldsymbol{\rho}||$$
$$\frac{\partial||\boldsymbol{\rho}||}{\partial z} = (\rho_u \sin\phi + \rho_n \cos\phi)/||\boldsymbol{\rho}||$$

$$\frac{\partial \mathrm{az}}{\partial x} = \frac{1}{(\rho_n^2 + \rho_e^2)}(\rho_e \sin\phi\cos\Theta - \rho_n \sin\Theta)$$
$$\frac{\partial \mathrm{az}}{\partial y} = \frac{1}{(\rho_n^2 + \rho_e^2)}(\rho_e \sin\phi\sin\Theta + \rho_n \cos\Theta)$$
$$\frac{\partial \mathrm{az}}{\partial z} = -\frac{1}{(\rho_n^2 + \rho_e^2)}\rho_e \cos\phi$$

Iterated Kalman Filter (v)

$$\frac{\partial \mathrm{el}}{\partial x} = \frac{1}{||\boldsymbol{\rho}||(||\boldsymbol{\rho}||^2 - \rho_u^2)^{1/2}} \left(||\boldsymbol{\rho}|| \cos\phi \cos\Theta - \rho_u \frac{\partial ||\boldsymbol{\rho}||}{\partial x} \right)$$

$$\frac{\partial \mathrm{el}}{\partial y} = \frac{1}{||\boldsymbol{\rho}||(||\boldsymbol{\rho}||^2 - \rho_u^2)^{1/2}} \left(||\boldsymbol{\rho}|| \cos\phi \sin\Theta - \rho_u \frac{\partial ||\boldsymbol{\rho}||}{\partial y} \right)$$

$$\frac{\partial \mathrm{el}}{\partial z} = \frac{1}{||\boldsymbol{\rho}||(||\boldsymbol{\rho}||^2 - \rho_u^2)^{1/2}} \left(||\boldsymbol{\rho}|| \sin\phi - \rho_u \frac{\partial ||\boldsymbol{\rho}||}{\partial z} \right)$$

Example (i)

- Same starting conditions as least squares

Iteration	Position (km)			Velocity (km/sec)		
0	6,990	1	1	1	1	1
1	7,121	1,046	192	−0.07	5.70	1.67
2	7,000	1,000	200	4.00	7.00	2.00
3	7,000	1,000	200	4.00	7.00	2.00

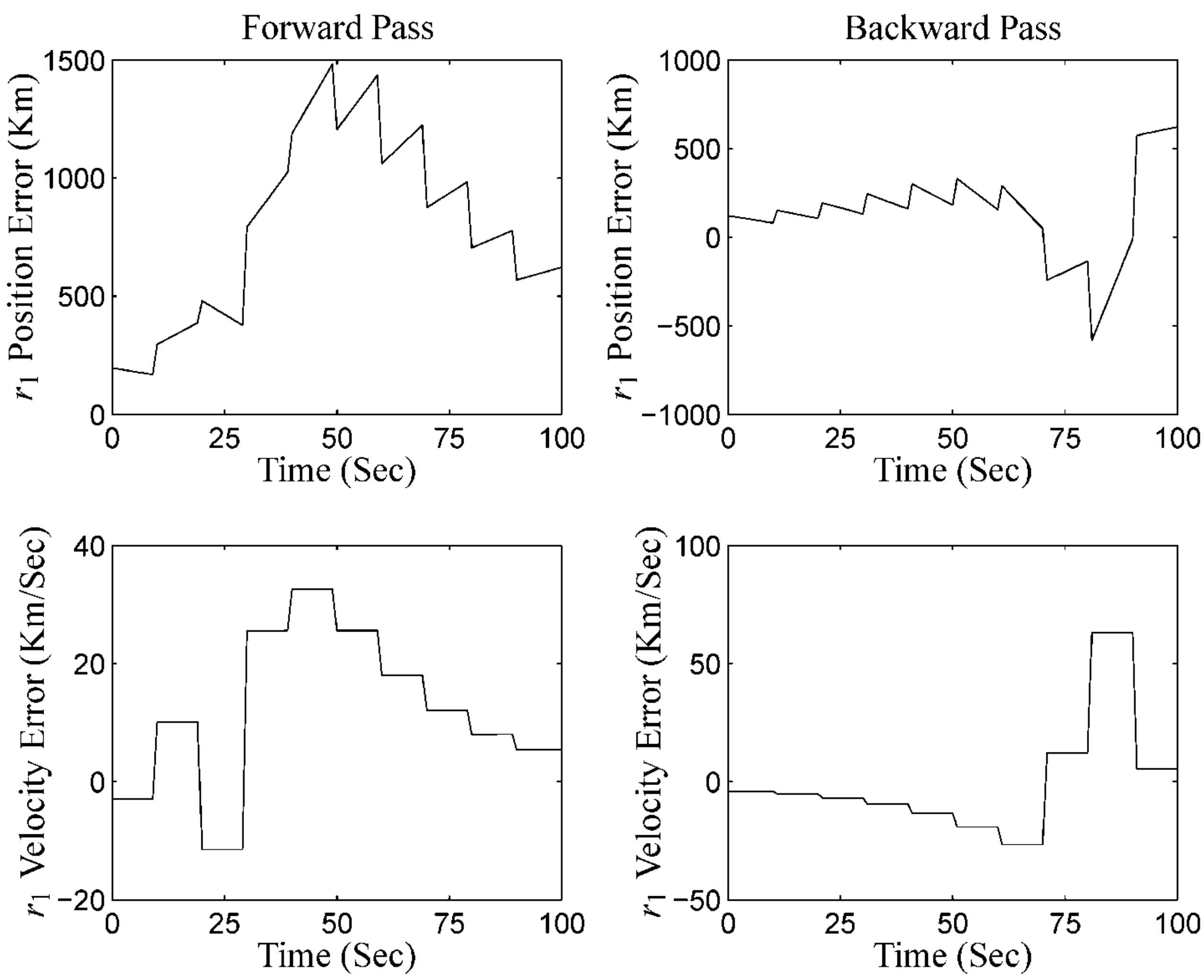
Identical results and covariance as least squares at final iteration

$$3\boldsymbol{\sigma}_{\hat{\mathbf{r}}} = \begin{bmatrix}1.26 & 0.25 & 0.51\end{bmatrix}^T \text{ km}, \quad 3\boldsymbol{\sigma}_{\dot{\hat{\mathbf{r}}}} = \begin{bmatrix}0.020 & 0.008 & 0.006\end{bmatrix}^T \text{ km/sec}$$

- Notes
  - EKF can be executed in real-time to provide a *navigation* solution, i.e., it provides sequential estimates
  - EKF can handle process noise, which is typically needed to handle unmodeled disturbances
  - EKF must append state vector to estimate biases while least squares approach doesn't need to do this

Example (ii)

First EKF Iteration



Code (i)

```
function [xe,xecov]=orbit_kal(x0,t0,tf,dt,y,tm,lat,sid,max,ymcov,p0,q);

% [xe,xecov]=orbit_kal(x0,t0,tf,dt,y,tm,lat,dis,max,ymcov,p0,q);
%
% Determines the initial orbit conditions at Epoch using
% an iterated Kalman filter approach. The inputs are:
%     x0 = initial guess (6x1)
%     t0 = initial time (1x1)
%     tf = final time (1x1)
%     dt = integration interval in sec. (1x1)
%      y = [range azimuth elevation] in km and rad (mx3)
%     tm = measurement times (assumed multiples of dt, and tm(1) = t0) (mx3)
%    lat = geocentric lattitude of radar in rad (1x1)
%    sid = sidereal time in sec (mx1)
%    max = maximum number of iterations
% ymcov = measurement covariance (3x3)
%     p0 = initial error covariance (6x6)
%      q = discrete process noise covariance (6x6)
```

Code (ii)

```
% Find Measurement Times
t=[t0:dt:tf]';
clear k;for i=1:length(tm), k(i)=find(t==tm(i));end;k=k(:);

% Measurements
ym=[y(:,1) y(:,2) y(:,3)];

% Initialize
m=length(t);
ye1=zeros(length(y),1);ye2=zeros(length(y),1);ye3=zeros(length(y),1);
mu=398600.64;
pcov=p0;
kkk=1;
xe=zeros(length(t),6);xe(1,:)=x0(:)';
pf=zeros(length(t),6);pb=zeros(length(t),6);
pf(1,:)=diag(pcov)';
clear xiter piter
rearth=6378;
phia=lat;theta=sid;
```

Code (iii)

```
% Main Loop
for jjjj=1:max,

for i=1:m-1;

if (i==k(kkk)),

xw=xe(i,:);

% Estimated Quantities
rhoud=cos(phia)*cos(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
   +cos(phia)*sin(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk))) ...
   +sin(phia)*(xw(3)-rearth*sin(phia));
rhoed=-sin(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
    +cos(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk)));
rhond=-sin(phia)*cos(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
    -sin(phia)*sin(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk))) ...
    +cos(phia)*(xw(3)-rearth*sin(phia));
rhod=norm([rhoud rhoed rhond]);
```

Code (iv)

```
% Partials
drdx=(rhoud*cos(phia)*cos(theta(kkk))-rhoed*sin(theta(kkk)) ...
    -rhond*sin(phia)*cos(theta(kkk)))/rhod;

drdy=(rhoud*cos(phia)*sin(theta(kkk))+rhoed*cos(theta(kkk)) ...
    -rhond*sin(phia)*sin(theta(kkk)))/rhod;

drdz=(rhoud*sin(phia)+rhond*cos(phia))/rhod;


fac1=1/(1+(rhoed/rhond)^2);
dadx=fac1*(-sin(theta(kkk))+rhoed*sin(phia)*cos(theta(kkk))/rhond)/rhond;
dady=fac1*(cos(theta(kkk))+rhoed*sin(phia)*sin(theta(kkk))/rhond)/rhond;
dadz=fac1*(-rhoed*cos(phia)/rhond)/rhond;

fac2=1/sqrt(1-(rhoud/rhod)^2);
dedx=fac2*(cos(phia)*cos(theta(kkk))-rhoud*drdx/rhod)/rhod;
dedy=fac2*(cos(phia)*sin(theta(kkk))-rhoud*drdy/rhod)/rhod;
dedz=fac2*(sin(phia)-rhoud*drdz/rhod)/rhod;
```

Code (v)

```
% Sensitivity
hpre=[drdx drdy drdz 0 0 0
      dadx dady dadz 0 0 0
      dedx dedy dedz 0 0 0];

% Gain
gain=pcov*hpre'*inv(hpre*pcov*hpre'+ymcov);

% Estimates
ye1=rhod;
ye2=atan2(rhoed,rhond);
ye3=asin(rhoud/rhod);
if (ye2-y(kkk,2) > pi), ye2=ye2-2*pi; end
if (y(kkk,2)-ye2 > pi), ye2=ye2+2*pi; end

ye=[ye1 ye2 ye3]';
```

Code (vi)

```
% Unpdate
 xe(i,:)=xe(i,:)+(gain*(ym(kkk,:)'-ye))';
 kkk=kkk+1;
 pcov=(eye(6)-gain*hpre)*pcov;

end

% Propagate
re=norm(xe(i,1:3));
mu=398600.64;
asub=mu/re^5*[3*xe(i,1)^2-re^2 3*xe(i,1)*xe(i,2) 3*xe(i,1)*xe(i,3)
              3*xe(i,1)*xe(i,2) 3*xe(i,2)^2-re^2 3*xe(i,2)*xe(i,3)
              3*xe(i,1)*xe(i,3) 3*xe(i,2)*xe(i,3) 3*xe(i,3)^2-re^2];
biga=[zeros(3) eye(3);asub zeros(3)];
ad=c2d(biga,zeros(6,1),dt);
pcov=ad*pcov*ad'+q;
```

Code (vii)

```
% Integrate
f1=dt*orbitfun(xe(i,:),mu);
f2=dt*orbitfun(xe(i,:)+0.5*f1',mu);
f3=dt*orbitfun(xe(i,:)+0.5*f2',mu);
f4=dt*orbitfun(xe(i,:)+f3',mu);
xe(i+1,:)=xe(i,:)+1/6*(f1'+2*f2'+2*f3'+f4');

pf(i+1,:)=diag(pcov)';

end

xf=xe;pb(m,:)=pf(m,:);pcov=p0;
kkk=kkk-1;
```

Code (viii)

```
% Backwards
for i=m:-1:2,

if (i==k(kkk)),

% Estimate quantities
xw=xe(i,:);

rhoud=cos(phia)*cos(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
    +cos(phia)*sin(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk))) ...
    +sin(phia)*(xw(3)-rearth*sin(phia));
rhoed=-sin(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
     +cos(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk)));
rhond=-sin(phia)*cos(theta(kkk)).*(xw(1)-rearth*cos(phia)*cos(theta(kkk))) ...
     -sin(phia)*sin(theta(kkk)).*(xw(2)-rearth*cos(phia)*sin(theta(kkk))) ...
     +cos(phia)*(xw(3)-rearth*sin(phia));
rhod=norm([rhoud rhoed rhond]);
```

Code (ix)

```
% Partial Derivatives
drdx=(rhoud*cos(phia)*cos(theta(kkk))-rhoed*sin(theta(kkk)) ...
    -rhond*sin(phia)*cos(theta(kkk)))/rhod;
drdy=(rhoud*cos(phia)*sin(theta(kkk))+rhoed*cos(theta(kkk)) ...
    -rhond*sin(phia)*sin(theta(kkk)))/rhod;
drdz=(rhoud*sin(phia)+rhond*cos(phia))/rhod;

fac1=1/(1+(rhoed/rhond)^2);
dadx=fac1*(-sin(theta(kkk))+rhoed*sin(phia)*cos(theta(kkk))/rhond)/rhond;
dady=fac1*(cos(theta(kkk))+rhoed*sin(phia)*sin(theta(kkk))/rhond)/rhond;
dadz=fac1*(-rhoed*cos(phia)/rhond)/rhond;

fac2=1/sqrt(1-(rhoud/rhod)^2);
dedx=fac2*(cos(phia)*cos(theta(kkk))-rhoud*drdx/rhod)/rhod;
dedy=fac2*(cos(phia)*sin(theta(kkk))-rhoud*drdy/rhod)/rhod;
dedz=fac2*(sin(phia)-rhoud*drdz/rhod)/rhod;

% Sensitivity
hpre=[drdx drdy drdz 0 0 0;dadx dady dadz 0 0 0;dedx dedy dedz 0 0 0];
```

Code (x)

```
% Estimates
ye1=rhod;
ye2=atan2(rhoed,rhond);
ye3=asin(rhoud/rhod);
if (ye2-y(kkk,2) > pi), ye2=ye2-2*pi; end
if (y(kkk,2)-ye2 > pi), ye2=ye2+2*pi; end

% Gain
gain=pcov*hpre'*inv(hpre*pcov*hpre'+ymcov);

% Update
ye=[ye1 ye2 ye3]';
xe(i,:)=xe(i,:)+(gain*(ym(kkk,:)'-ye))';
kkk=kkk-1;
pcov=(eye(6)-gain*hpre)*pcov;

end
```

Code (xi)

```
% Propagate
re=norm(xe(i,1:3));
mu=398600.64;
asub=mu/re^(5)*[3*xe(i,1)^2-re^2 3*xe(i,1)*xe(i,2) 3*xe(i,1)*xe(i,3)
                3*xe(i,1)*xe(i,2) 3*xe(i,2)^2-re^2 3*xe(i,2)*xe(i,3)
                3*xe(i,1)*xe(i,3) 3*xe(i,2)*xe(i,3) 3*xe(i,3)^2-re^2];
biga=[zeros(3) eye(3);asub zeros(3)];

ad=c2d(-biga,zeros(6,1),dt);
pcov=ad*pcov*ad'+q;
pb(i-1,:)=diag(pcov)';

% Integrate
f1=-dt*orbitfun(xe(i,:),mu);
f2=-dt*orbitfun(xe(i,:)+0.5*f1',mu);
f3=-dt*orbitfun(xe(i,:)+0.5*f2',mu);
f4=-dt*orbitfun(xe(i,:)+f3',mu);
xe(i-1,:)=xe(i,:)+1/6*(f1'+2*f2'+2*f3'+f4');

end
```

Code (xii)

```
plast=pcov;
xb=xe;pcov=p0;

xiter(jjjj,:)=xe(1,:);
piter(jjjj,:)=pb(1,:);

xe(1,:)

end

xe=xiter;
xecov=plast;
```

Code (xiii)

```
function f=orbitfun(x,mu);

f=zeros(6,1);
r=sqrt(x(1)^2+x(2)^2+x(3)^2);
r3=r^3;
f(1)=x(4);f(2)=x(5);f(3)=x(6);
f(4)=-mu/r3*x(1);f(5)=-mu/r3*x(2);f(6)=-mu/r3*x(3);
```