DATA FUSION II, Radar Trackers

Date: February 16, 2006 **Time:** 5:00 – 7:30 PM
Instructor: Dr. James K Beard
Credits: 1 **Course Code:** 0901-504-04 **Registration Number:** 13459

Today's topics:

1 Homework and Study Problems from Last Week

1.1 Gelb problem 3-1

The linear variance equation is the differential equation for propagation of errors of a noise-driven system. This is developed in Section 3.7, Propagation of Errors, pp. 75-78. The linear variance equation is given as Equation (3.7-17) at the bottom of page 77:

$$\dot{P}(t) = F(t)\cdot P(t) + P(t)\cdot F(t) + G(t)\cdot Q(t)\cdot G^T(t) \tag{1.1}$$

The problem statement is to verify a solution as given:

$$P(t) = \Phi(t,t_0)\cdot P(t_0)\cdot \Phi^T(t,t_0) + \int_{t_0}^{t} \Phi(t,\tau)\cdot G(\tau)\cdot Q(\tau)\cdot G^T(\tau)\cdot \Phi^T(t,\tau)\cdot d\tau \tag{1.2}$$

The simplest way to do this is to take the derivative of the candidate expression for *P(t)* and show that the right-hand-side of the differential equation is obtained. In doing this, we need the definition, from equations (3.7-13) and (3.7-14) on page 77,

$$\dot{\Phi}(t) = F(t)\cdot\Phi(t) \tag{1.3}$$

and Leibnitz' rule for differentiation of an integral,

$$\frac{d}{dt}\left(\int_{f_1(t)}^{f_2(t)} g(t,x)\cdot dx\right) = g\left(t, f_2(t)\right)\cdot\frac{df_2(t)}{dt} - g\left(t, f_1(t)\right)\cdot\frac{df_1(t)}{dt} + \int_{f_1(t)}^{f_2(t)}\frac{d}{dt}g(t,x)\cdot dx \tag{1.4}$$

Because of the size of the equations, we will take one term at a time. We begin with

$$\begin{aligned}&\frac{d}{dt}\left(\Phi(t,t_0)\cdot P(t_0)\cdot\Phi^T(t,t_0)\right)\\ &\quad = F(t)\cdot\Phi(t,t_0)\cdot P(t_0)\cdot\Phi^T(t,t_0) + \Phi(t,t_0)\cdot P(t_0)\cdot\Phi^T(t,t_0)\cdot F^T(t)\end{aligned} \tag{1.5}$$

Repeating this operation on the integrand for *P(t)* produces the remainder of the terms in $F(t)\cdot P(t) + P(t)\cdot F(t)$. Leibnitz' rule on the integral, as applied to the limits of the integral, produce the term

$$\Phi(t,t)\cdot G(t)\cdot Q(t)\cdot G^T(t)\cdot\Phi^T(t,t) = G(t)\cdot Q(t)\cdot G^T(t) \tag{1.6}$$

and we are done.

1.2 Gelb Problem 2-5

Show that the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \tag{1.7}$$

satisfies the polynomial equation

$$A^2 - 5\cdot A - 2\cdot I = 0 \tag{1.8}$$

and find the eigenvalues. Use the results to show the form of the solution in terms of exponential functions.

Solution: The characteristic equation is found from

$$\left\|\begin{bmatrix} 1-\lambda & 2 \\ 3 & 4-\lambda \end{bmatrix}\right\| = \lambda^2 - 5\cdot\lambda - 2 = 0 \tag{1.9}$$

The Cayley-Hamilton theorem (bottom of page 19) states that a matrix satisfies its own characteristic equation, so we have proven the polynomial matrix equation.

From the polynomial equation, we can use

$$A^2 = 5\cdot A + 2\cdot I \tag{1.10}$$

to reduce any polynomial or series in *A* to a linear combination of *A* and *I*. The eigenvalues are

$$\lambda = \frac{5\pm\sqrt{33}}{2} \tag{1.11}$$

The remaining equation, to find the closed forms for $a_1(t)$ and $a_2(t)$, where

$$\exp(A\cdot t) = a_1(t)\cdot I + a_2(t)\cdot A \tag{1.12}$$

is done as follows. First for a two-by-two matrix, the Cayley-Hamilton theorem gives us

$$A^2 - (\lambda_1 + \lambda_2) \cdot A + \lambda_1 \cdot \lambda_2 \cdot I = 0 \tag{1.13}$$

or,

$$A^2 = (\lambda_1 + \lambda_2) \cdot A - \lambda_1 \cdot \lambda_2 \cdot I \tag{1.14}$$

This equation means that the matrix exponential series as given by Equation (2.1-48) on page 20 can be collapsed to a linear combination of *I* and *A*. Since we are looking at $\exp(A \cdot t)$, we use

$$(A \cdot t)^2 = (\lambda_1 + \lambda_2) \cdot t \cdot (A \cdot t) - \lambda_1 \cdot \lambda_2 \cdot t^2 \cdot I \tag{1.15}$$

Note that the eigenvalues of $(A \cdot t)$ are the eigenvalues of *A* multiplied by *t*.

1.2.1 General Scalar Solution

The Cayley-Hamilton theorem tells us that the matrix itself satisfies its characteristic equation, but we know also that both the eigenvalues also satisfy the characteristic equation. Since the characteristic equation is used to collapse the series for the exponential into a polynomial of order *N-1* (a first-order polynomial for two by two matrices), we also have

$$\exp(\lambda_i \cdot t) = a_1(t) + a_2(t) \cdot \lambda_i, \;\; i = 1, 2 \tag{1.16}$$

We can write this equation down for both eigenvalues and solve for $a_1(t)$ and $a_2(t)$:

$$\begin{aligned} \exp(\lambda_1 \cdot t) &= a_1(t) + a_2(t) \cdot \lambda_1 \\ \exp(\lambda_2 \cdot t) &= a_1(t) + a_2(t) \cdot \lambda_2 \end{aligned} \tag{1.17}$$

is a set of two linear equations in $a_1(t)$ and $a_2(t)$, and can be solved for $a_1(t)$ and $a_2(t)$. The solution is

$$\begin{aligned} a_1(t) &= \frac{\lambda_1 \cdot \exp(\lambda_2 \cdot t) - \lambda_2 \cdot \exp(\lambda_1 \cdot t)}{\lambda_1 - \lambda 2} \\ a_2(t) &= \frac{\exp(\lambda_1 \cdot t) - \exp(\lambda_2 \cdot t)}{\lambda_1 - \lambda_2} \end{aligned} \tag{1.18}$$

These can be substituted into the two linear equations in $a_1(t)$ and $a_2(t)$ to verify that they are correct. We can differentiate the series for $\exp(A \cdot t)$ term by term and show that

$$\frac{d}{dt} \exp(A \cdot t) = A \cdot \exp(A \cdot t) \tag{1.19}$$

and use this to verify the solution we have for $a_1(t)$ and $a_2(t)$. Note that $a_1(t)$ and $a_2(t)$ as we have found them are the same as given in the problem statement when it gives is "the form $a_1(t)$ and $a_2(t)$ (a hint that there may be a sign difference or a proportionality factor between the equations given there and the exact solution as we have found here), but the sign of the form for $a_1(t)$ given in the problem statement is incorrect.

1.2.2 General Matrix Solution

We can extend the result to higher order matrices by simply writing the set of two linear equations in equations in $a_1(t)$ and $a_2(t)$ as a set of *N* linear equations in a set of *N* time functions:

$$\exp\left(\lambda_i \cdot t\right) = \sum_{j=1}^{N} a_j\left(t\right) \cdot \lambda_i^{j-1}, \;\; i = 1, 2 \ldots N \tag{1.20}$$

This set of equations can be written as a vector equal to a matrix times a vector:

$$\begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \ldots & \lambda_1^{N-1} \\ 1 & \lambda_2 & \lambda_2^2 & \ldots & \lambda_2^{N-1} \\ 1 & \lambda_3 & \lambda_3^2 & \ldots & \lambda_2^{N-1} \\ \vdots & & & \ddots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \ldots & \lambda_N^{N-1} \end{bmatrix} \cdot \begin{bmatrix} a_1\left(t\right) \\ a_2\left(t\right) \\ a_3\left(t\right) \\ \vdots \\ a_N\left(t\right) \end{bmatrix} = \begin{bmatrix} \exp\left(\lambda_1 \cdot t\right) \\ \exp\left(\lambda_2 \cdot t\right) \\ \exp\left(\lambda_3 \cdot t\right) \\ \vdots \\ \exp\left(\lambda_N \cdot t\right) \end{bmatrix} \tag{1.21}$$

The set of $a_i(t)$ are found by left-multiplying this equation by the inverse of the matrix. The solution to part (b) of the problem can easily be seen to be given by this equation for $N=2$.

The matrix is a special form, its rows being given by successive powers of different constants. The determinant of such a matrix is called a *Vandermonde determinant* and is very famous, and a lot of papers have been written about it. In particular, a closed form for this determinant is (see *Introduction to Matrix Analysis*, Second Edition, by Richard Bellman, page 193)

$$\left|\lambda_i^{j-1}\right| = \prod_{1 \le i < j \le N} \left(\lambda_j - \lambda_i\right) \tag{1.22}$$

Note that the matrix is singular if any two eigenvalues are equal.

2 The Kalman Filter Equations

2.1.1 Summary of Equations of the Kalman Filter

State Vector $\underline{x}$, measurement vector $\underline{y}$, measurement model

$$\begin{aligned} \underline{y} &= \underline{h}\left(\underline{x}\right) + \underline{v} \\ H &= \frac{\partial \underline{h}\left(\underline{x}\right)}{\partial \underline{x}} \\ R &= \mathrm{Cov}\left(\underline{v}\right) = E\left(\underline{v} \cdot \underline{v}^T\right) \end{aligned} \tag{2.1}$$

State vector extrapolation model

$$\begin{aligned} \dot{\underline{x}} &= \underline{f}\left(\underline{x}\right) + \underline{w} \\ F &= \frac{\partial \underline{f}\left(\underline{x}\right)}{\partial \underline{x}} \\ Q &= \mathrm{Cov}\left(\underline{w}\right) = E\left(\underline{w} \cdot \underline{w}^T\right) \end{aligned} \tag{2.2}$$

Covariance extrapolation approximation

$$\begin{aligned} \tilde{P} &\approx \Phi \cdot P_{\_} \cdot \Phi^T + Q \cdot T \\ \dot{\Phi} &= F \cdot \Phi, \;\; \Phi\left(t_{\_}\right) = I \\ \Phi &\approx I + F \cdot T \end{aligned} \tag{2.3}$$

State vector extrapolation approximation

$$\underline{\tilde{x}} = \Phi \cdot \underline{\hat{x}}_{-} \tag{2.4}$$

Kalman gain

$$K = \tilde{P} \cdot H^T \cdot \left(H \cdot \tilde{P} \cdot H^T + R \right)^{-1} \tag{2.5}$$

or, if we have P available before K is computed,

$$K = P \cdot H^T \cdot R^{-1} \tag{2.6}$$

State vector update

$$\underline{\hat{x}} = \underline{\tilde{x}} + K \cdot \left(\underline{y} - \underline{h}\left(\underline{\tilde{x}} \right) \right) \tag{2.7}$$

Covariance extrapolation short form

$$P = \left(I - K \cdot H \right) \cdot \tilde{P} \quad \text{Short form; NUMERICALLY UNSTABLE} \tag{2.8}$$

Joseph stabilized form for covariance extrapolation (good for most applications)

$$P = \left(I - K \cdot H \right) \cdot \tilde{P} \cdot \left(I - K \cdot H \right)^T + K \cdot R \cdot K^T \tag{2.9}$$

Normal form or information matrix format for covariance extrapolation – use to check for accuracy of Joseph stabilized form:

$$P^{-1} = \tilde{P}^{-1} + H^T \cdot R^{-1} \cdot H \tag{2.10}$$

Other: square root filters (not treated in Gelb; overview in the next course)

2.2 The Covariance Update and Kalman Filter Stability

We offered four forms for the covariance update for the Kalman filter. We will mention them in turn and give comments.

2.2.1 The Short Form

The short form, given in many Kalman filter references is

$$P = \left(I - K \cdot H \right) \cdot \tilde{P} \tag{2.11}$$

This equation is derived using sophisticated matrix algebra manipulations (see Gelb page 109, Equation (4.2.16) where this form is presented with the comment “after some manipulation”).

This form is not numerically stable because its use tends to cause numerical noise to accumulate as it is applied for successive updates. In many cases, the range errors in a tracker will become small for targets in track for a long time while the azimuth errors will not, so that the covariance matrix P will have some very small elements as well as some very large elements. This type of matrix is called *ill conditioned* because simple numerical operations with them are prone to numerical error. Use of the short form of the covariance update with ill-conditioned covariance matrices will cause the result to lose its symmetry and often to lose the property of positive-definiteness. If the covariance matrix has a zero or negative eigenvalue, the tracker will become unstable – tracker errors will grow exponentially with successive updates.

The Kalman filter did not fall into regular use after its introduction in 1960 because of difficulties in practical use. IEEE papers declaring the Kalman filter unusable because of numerical stability appeared in print as late as 1975. The principal cause of these difficulties was the use of the short form of the covariance update. For a generation (1975 through about 1990), few practitioners used the short form and the Kalman filter

became ubiquitous. In recent years, these lessons have been forgotten. Whenever you hear about a Kalman filter in development having numerical or intermittent performance problems, you will, with probability one, find that their development software is using the short form of the covariance update.

You should never use the short form of the covariance update in software. Its only virtue is a few less floating point operations that more suitable alternatives, an advantage that was dubious in 1975 and is totally meaningless today. It's use in algebraic manipulations is, of course, correct, but its history as the root of most numerical difficulties in implementation of Kalman filters is a red flag to examine any results for numerical issues before reduction to practice.

2.2.2 The Joseph Stabilized Form

The Joseph stabilized form is

$$P = \left(I - K \cdot H\right) \cdot \tilde{P} \cdot \left(I - K \cdot H\right)^T + K \cdot R \cdot K^T \tag{2.12}$$

This is the equation that is obtained when the covariance propagation equation is used with the Kalman update equation, and holds for any value of the Kalman gain *K* that is used, not just the optimum. As such it can be used when the Kalman gain is modified from the results of the standard equation for any reason. In addition, it always produces a symmetrical result and it is known to be numerically stable when used with poorly conditioned matrices.

You should always use the Joseph stabilized form for your fist cut implementations. You may never need to upgrade your covariance update. Unless your tracker has special problems or issues, the Joseph stabilized form for the covariance update will provide an adequate method for implementation.

2.2.3 The Normal Form

Substituting the optimum Kalman gain into the Joseph stabilized form results in

$$P = \tilde{P} - \tilde{P} \cdot H^T \cdot \left(H \cdot \tilde{P} \cdot H^T + R\right)^{-1} \cdot H \cdot \tilde{P} \tag{2.13}$$

The Matrix Inversion Lemma shows that this is equivalent to

$$P^{-1} = \tilde{P}^{-1} + H^T \cdot R^{-1} \cdot H \tag{2.14}$$

This form is very stable for poorly conditioned covariance matrices and should be used as a check if numerical issues are suspected when the Joseph stabilized form is in use. If switching to the normal form solves the problem, then you know that the tracker you are working with has a very poorly conditioned covariance matrix.

The normal form is suitable for general use in trackers because the order of the covariance matrix is not large, and inversion of matrices on the order of three-by-three to 12 by 12 or even larger, at rates seen in radar updates, is not a significant computer load in comparison to other tracker functions such as data association.

This type of update is called the normal form because it occurs in estimation theory when the state vector does not change (the system transition matrix Φ is *I* and the plant noise *Q* is zero) and new data is presented in batches. The inverse of the covariance matrix is

simply incremented with each update. Sometimes the inverse of the covariance matrix is called the *information matrix* for this reason.

2.2.4 Square Root Filters

Square root filters use a factorization of the covariance matrix of the form

$$P = S \cdot S^T \tag{2.15}$$

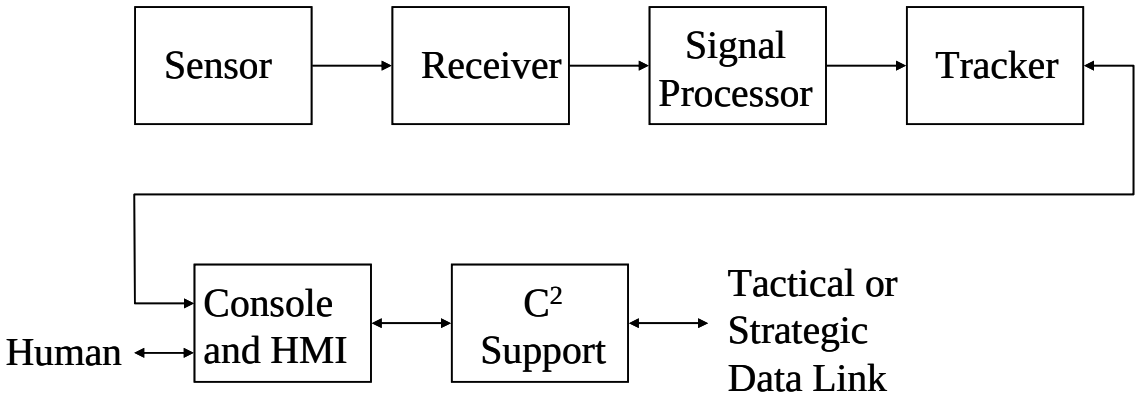
where the matrix *S* is upper triangular. Variations include collection of the diagonal terms of *S* into a diagonal matrix D so that

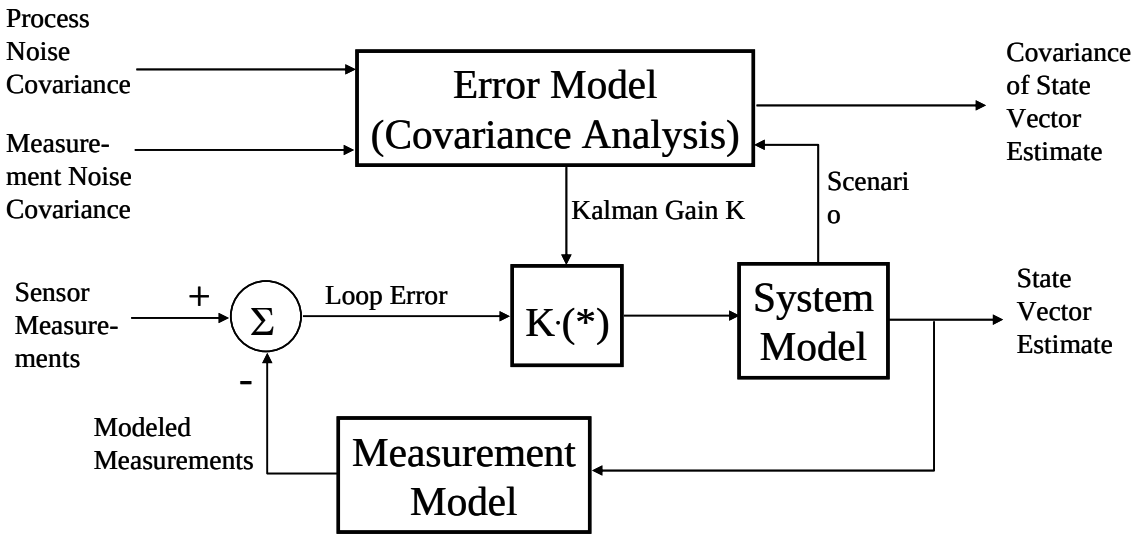
$$P = U \cdot D \cdot U^T \tag{2.16}$$

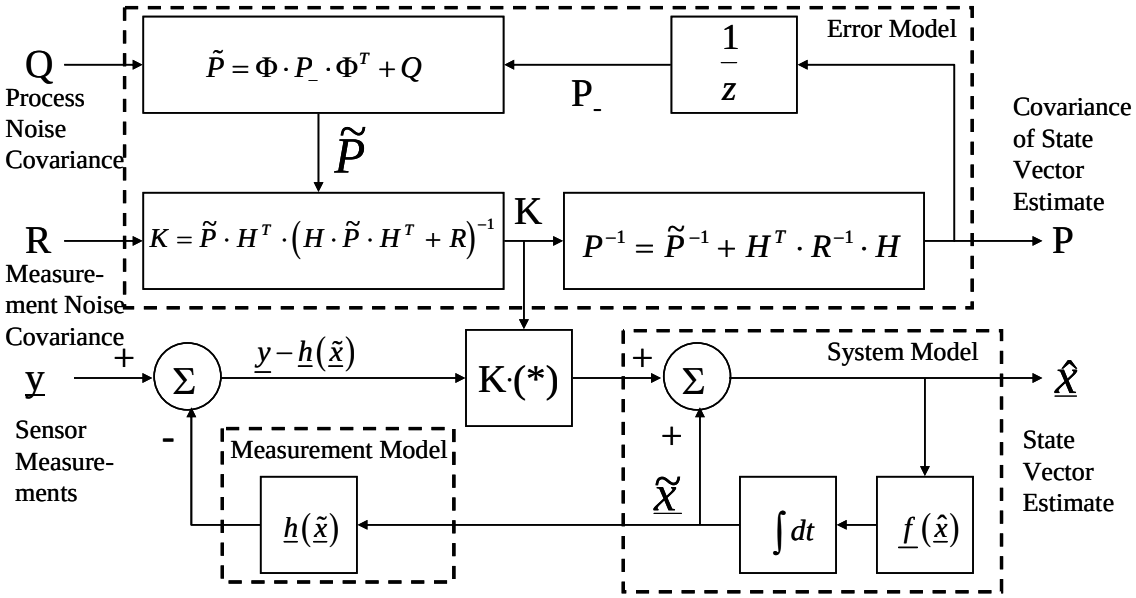
and the diagonal elements of the upper triangular matrix *U* are all ones, and the square root information matrix. The square root filters are useful in providing robust performance when the covariance matrix is poorly conditioned. The computation penalty of their use is negligible but their algebra is not as simple as that of the Kalman filter equations. *Square root filters should be used when robustness is a driving requirement in trackers that have poorly conditioned covariance matrices.*

Square root filters will be presented in Data Fusion III.

3 The Kalman Filter Data Flow




Process Noise Covariance
Measure-ment Noise Covariance
Error Model (Covariance Analysis)
Covariance of State Vector Estimate
Kalman Gain K
Scenari o
Sensor Measure-ments
+
Σ
-
Loop Error
K·(*)
System Model
State Vector Estimate
Modeled Measurements
Measurement Model



4 Tracker Architectures

4.1.1 Top Level Requirements

Tracker architectures are developed according to the principles of system engineering, and as such are requirements driven. The top-level (user) requirements include

- Accept the radar measurements that are available (primary site radar, or PSR)
- Accept inputs from the transponder subsystem (secondary site radar, or SSR)
- Produce target tracks to the scan converter (unless we are talking about the tracker in the scan converter)
- Produce outputs acceptable to the ATC consoles
- Produce C2 outputs to other users

4.1.2 Development Requirements

Implied requirements, or requirements on the tracker functions that are determined through the analysis and design process, include

- Data accuracy consistent with the needs of the ATC users
- Proper association of SSR data with PSR tracks
- No spurious tracks
- No loss of track or corrupted tracks in dense target environments

4.1.3 Design Requirements

Lower-level requirements, or detailed requirements, are determined as the design process proceeds. This process includes modeling and simulation that includes actual tracker

designs, and these requirements document the results of this process. These requirements documents provide an output of the development process and an input to the design process.

These design requirements define a set of software modules to be implemented, their interfaces with other software, and the data that is interchanged. When data enters or exits a system block, the requirements of that data interchange are documented as part of an interface design document (IDD) that specifies the data, its physical units, the accuracy and format of the numbers, and the frequency that the data is sent or made available.

A tracker architecture is a list of tracker functions and their relationships to each other, and the data flow in the tracker as a whole. Tracker functions include

- Translation of coordinate systems from those used in input data
- Estimation of errors and correlations in data inputs
- Proper association of input data with the correct track files
- Update of track files, including estimation of errors in positions in radar contacts
- Proper formatting of data for users, including coordinate transformations and reduction of error information to formats appropriate for users

All these functions are coordinated by a tracker manager function. This is the highest level function in the tracker, and is the tracker to the rest of the software.

4.1.4 Separation of Function

Normally a radar tracker will have separate simple trackers for range, azimuth, and, if information is present, attitude. Occasionally performance requirements such as association reliability in dense environments will require that the target be tracked in two or even three dimensions. Non-FAA requirements such as trackers in fighter aircraft or site defense radars may require that body angles of the radar contact be part of the tracker. As such a tracker may have for, six, eight, or more states. When this is done, the underlying two-state range, bearing, and altitude trackers remain in place to support the association function and provide a backup for the higher-performing but less robust sophisticated multistate Kalman filters. The Kalman filters may be used to provide a radar contact data base through their successful associations for a particular contact that is used in a batch processing estimation for the highest possible performance.

4.1.5 The Association Function

The critical function in the tracker is the association function. The radar contact data or transponder response is matched up with the track files on the basis of the data present in the input, and the information available in the track file. The information on errors in this data is critical to the reliability of the association process, particularly when traffic is dense. Data misassociation results in corrupted tracks, spurious tracks, and dropping of legitimate tracks. A dense traffic condition is the most difficult for reliable association and the situation where ATC demands on robustness are most urgent.

Data association is nearly always done through a nearest-neighbor approach. The radar track that predicts a target range or position closest to that given by the radar contact is assumed to be the correct track file. When multiple radar contacts and multiple track files are present, this sorting process can become less clear. A score is used to make the final decision in these cases.

The score function is computed by extrapolating the target position for each candidate track file to the time of the radar contact, including the covariance of the estimate. Then, a nearest-neighbor number is computed. A threshold is used to prune the possibilities, and the score is used to make any final decisions if more than one radar return passes the threshold test.

There are three score functions that are most often used to provide a nearest-neighbor number:

Hyperpolygon

$$J_{HP} = \sum_{i=1}^{M} \frac{\left| y_i - h_i\left(\underline{\tilde{x}}\right)\right|}{\sqrt{\sigma^2\left(y_i\right) + \sigma^2\left(h_i\left(\underline{\tilde{x}}\right)\right)}} < thr_{HP} \tag{4.1}$$

Hypercube

$$J_{HC} = \underset{1\le i\le M}{\text{Max}}\left(\frac{\left| y_i - h_i\left(\underline{\tilde{x}}\right)\right|}{\sqrt{\sigma^2\left(y_1\right) + \sigma^2\left(h_i\left(\underline{\tilde{x}}\right)\right)}} \right) < thr_{HC} \tag{4.2}$$

Hypersphere (Bhatacharya distance)

$$J_{HS} = \left(\underline{y} - \underline{h}\left(\underline{\tilde{x}}\right)\right)^T \cdot \left(H \cdot \tilde{P} \cdot H^T + R\right)^{-1} \cdot \left(\underline{y} - \underline{h}\left(\underline{\tilde{x}}\right)\right) < thr_{HS} \tag{4.3}$$

For one measurement, all of them are the same – the absolute value of the distance between two points. For more than one measurement, they differ; the unit hyperpolygon $\left(thr_{HP} = 1\right)$ is contained within the unit sphere $\left(thr_{SP} = 1\right)$, which is contained in the unit cube $\left(thr_{HC} = 1\right)$. For two dimensions, the hyperpolygon is a diamond that with vertices on the unit circle, and the hypercube is a square; the hypersphere is a circle. For three dimensions, the hyperpolygon is an octagon contained inside the unit sphere with vertices on it at the axis crossings, and the unit sphere is contained in the unit cube.

For the number of measurements *M* used in association less than three, any of them will do. However, for *M>2* the value of the hypercube and hyperpolygon as a score function becomes less attractive because of increasing statistical probability of poorer fits getting scores equivalent to better fits. A measure of this is the relative volume incoxed in each threshold. All volumes are multiplied by their respective threshold values to the power of M.

Hypersphere volume

$$V_{HS} = \frac{\pi^{M/2}}{\Gamma\left(\frac{M}{2}+1\right)} \tag{4.4}$$

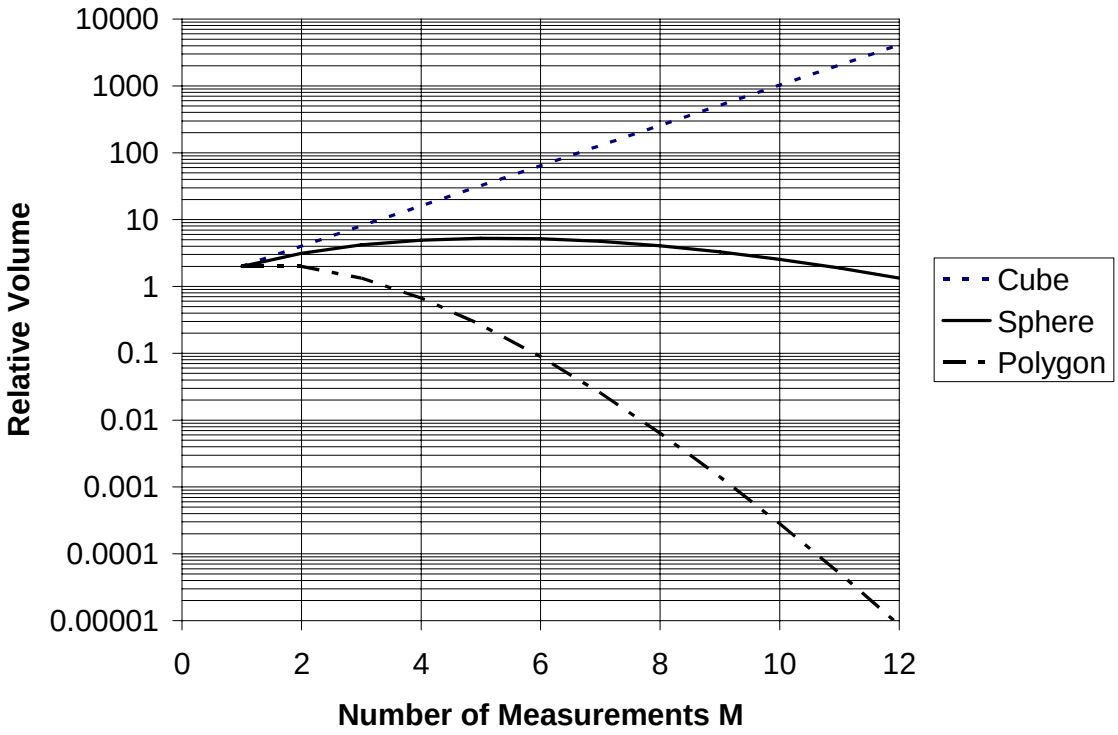
Hyperpolygon volume

$$V_{HP} = \frac{2^M}{\Gamma(M+1)} \tag{4.5}$$

Hypercube volume

$$V_{HC} = 2^M \tag{4.6}$$

Below we have plots of volumes contained within the unit hypercube, hypersphere, and hyperpolygon, respectively. We see that as the number of measurements *M* increases, the use of the Bhatacharya distance becomes increasingly important for robust association.



4.1.6 The Input and Output Functions

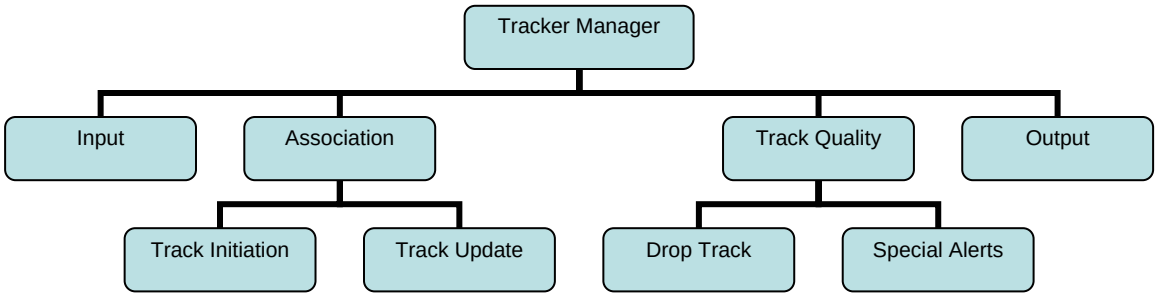
Data input to the tracker is nearly always a "data push" – the data comes in and the tracker is notified to process it. Data output can be "data push" to the ATC system or "data pull" – a request for information on a specific radar contact.

Input includes formatting the radar contact data, including coordinate transformations and ancillary data such as SSR transponder aircraft type data.

5 The Radar Tracker Software Diagram

A simple example of a hierarchical tracker function diagram is shown below. The top level function is the tracker manager. It controls functions of input, association, track quality computations, and output, or servicing data-pull operations. The association function in turn controls the functions of track update and track initiation. The track

quality function controls the functions of dropping tracks and special alerts, such as collision alerts.



6 Data Fusion Considerations

Data fusion is implicit because the PSR and SSR are very different sensors. The PSR is usually near the periphery of the airport, or even off-site. Occasionally the PSR is located on a nearby tower or building, not on the airport grounds. Also, some larger airports use two PSRs. The SSR is often located on the control tower itself, although occasionally it is part of the PSR installation. Thus the radar contact range and azimuth data must be modified to take into account the differences in positions of the sensors.

The information is different in other ways:

- Altitude information is available from the SSR but not from the PSR.
- Aircraft display graphics information such as identification is available from the SSR only.
- Aircraft azimuth information from both PSR and SSR is less accurate in terms of position than range data, particularly at longer ranges.

The data fusion operation uses the Bhatacharya distance. This computation requires the entire covariance matrix. Often only the diagonal terms are provided in software designs because the requirements for robust association and implied requirements for the entire covariance matrix are not provided to the software designers. *Always provide the entire covariance matrix when estimation errors are required.* Symmetry of the covariance matrix can be exploited to minimize the amount of data provided in data headers and resulting bus bandwidth usage.

7 The Quiz

The quiz will be open book, open notes, take-home. You will have one week to complete the quiz. Please return your quiz to Fred Snyder before close of business on Thursday, February 23.

The quiz will be passed out at the end of class. Please work alone, although you may ask me questions by email. I will reply when appropriate by email to all so that everyone has the same clarifications (and to avoid redundant questions by clarifying just once when needed).

Some portions of the quiz will be two questions, and you will decide which to answer. Only one answer will be graded. When this is done, you will have a choice between an analysis question and an essay question. You don't have to decide until you hand in the paper, but you must designate which answer you are submitting on your quiz. I will ignore the other answer, if present.