

Signal Processing for Intelligent Sensor Systems

DAVID C. SWANSON

Signal Processing and Communications Series



Signal Processing for Intelligent Sensor Systems

DAVID C. SWANSON

*The Pennsylvania State University
University Park, Pennsylvania*



MARCEL DEKKER, INC.

NEW YORK • BASEL

ISBN: 0-8247-9942-9

This book is printed on acid-free paper.

Headquarters

Marcel Dekker, Inc.
270 Madison Avenue, New York, NY 10016
tel: 212-696-9000; fax: 212-685-4540

Eastern Hemisphere Distribution

Marcel Dekker AG
Hutgasse 4, Postfach 812, CH-4001 Basel, Switzerland
tel: 41-61-261-8482; fax: 41-61-261-8896

World Wide Web

<http://www.dekker.com>

The publisher offers discounts on this book when ordered in bulk quantities. For more information, write to Special Sales/Professional Marketing at the headquarters address above.

Copyright © 2000 by Marcel Dekker, Inc. All Rights Reserved.

Neither this book nor any part may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, microfilming, and recording, or by any information storage and retrieval system, without permission in writing from the publisher.

Current printing (last digit):

10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

processing except for the fact that the signals represent a time integral of the data. However, this integration is well-known to suppress random noise, making frequency domain adaptive processing very attractive to problems where the signals of interest are sinusoids in low signal-to-noise ratio (SNR). This scenario applies to a wide range of adaptive signal processing problems. However, the frequency domain adaptive filter transfer functions operations are eventually converted back to the time domain to give the filter coefficients. Spectral leakage from sinusoids which may not be perfectly aligned to a frequency bin must be eliminated using the technique presented in Section 5.4 to insure good results. Presentation of the Kalman filter, IIR and multichannel forms for adaptive LMS and lattice filters, and frequency domain processing complement the applications of signal processing presented throughout the rest of this book.

10.1 ADAPTIVE KALMAN FILTERING

Consider the problem of tracking the extremely high-velocity flight path of a low-flying satellite from its radio beacon. It is true that for a period of time around the launching of Sputnik, the world's first satellite, many ham radio enthusiasts provided very useful information on the time and position of the satellite from ground detections across the world. This information could be used to refine an orbit state model which in turn is used to predict the satellites position at future times. The reason for near public panic was that a satellite could potentially take pictures of sensitive defense installations or even deliver a nuclear bomb, which would certainly wreck one's day. So with public and government interest in high gear, the space race would soon join the arms race, and networks of tracking radio stations were constructed not only to detect and track enemy satellites, but also to communicate with one's own satellites. These now-familiar satellite parabolic dishes would swivel as needed to track and communicate with a satellite passing overhead at incredible speeds of over 17,500 miles per hour (for an altitude of about 50 miles). At about 25,000 miles altitude, the required orbital velocity drops to about 6500 mi/hr and the satellite is *geosynchronous*, meaning that it stays over the same position on the ground.

For clear surveillance pictures, the satellite must have a very low altitude. But, with a low altitude, high barometric pressure areas on the ground along the flight path will correspond to the atmosphere extending higher up into space. The variable atmosphere, as well as gravity variations due to the earth not being perfectly spherical, as well as the affects of the moon and tides, will cause the satellite's orbit to change significantly. A receiver dish can be pointed in the general direction of the satellite's path, but the precise path is not completely known until the tracking algorithm "locks in" on the trajectory. Therefore, today adaptive tracking systems around the world are used to keep up to date the trajectories of where satellites (as well as orbital debris) are at all times.

Tracking dynamic data is one of the most common uses of the Kalman filter. However, the distinction between Kalman filtering and the more general recursive least-squares becomes blurred when the state vector is replaced by other basis functions. In this text we specifically refer to the Kalman filter as a recursive least squares estimator of a state vector, most commonly used for a series of differentials depicting the state of a system. All other "non-state vector" basis functions in this

text are referred to in the more general terms of a recursive least squares algorithm. The Kalman filter has enabled a vast array of 20th Century mainstay technologies such as: world-wide real-time communications for telephones, television, and the Internet; the global positioning system (GPS) which is revolutionizing navigation and surveying; satellite wireless telephones and computer networks; industrial and process controls; and environmental forecasting (weather and populations, diseases, etc); and even monetary values in stock and bond markets around the world. The Kalman filter is probably even more ubiquitous than the FFT in terms of its value and uses to society.

The Kalman filter starts with a simple state equation relating a column vector of n_z measurements, $z(t)$, at time t , to a column vector of n_x states, $x(t)$, as seen in Eq. (10.1.1). The measurement matrix, $H(t)$, has n_z rows and n_x columns and simply relates the system states linearly to the measurements. $H(t)$ usually will not change with time but it is left as a time variable for generality. The column vector $w(t)$ has n_z rows and represents the measurement noise, the expected value of which will play an important part in the sensitivity of the Kalman filter to the measurement data.

$$z(t) = H(t)x(t) + w(t) \quad (10.1.1)$$

It is not possible statistically to reduce the difference between $z(t)$ and $Hx(t)$ below the measurement noise described by the elements of $w(t)$. But, the states in $x(t)$ will generally have significantly less noise than the corresponding measurements due to the “smoothing” capability of the Kalman filter. We can therefore define a quadratic cost function in terms of the state to measurement error to be minimized by optimizing the state vector.

$$J(N) = \frac{1}{2} \left[\begin{array}{c} z(1) \\ z(2) \\ \vdots \\ z(N) \end{array} - \begin{array}{c} H(1) \\ H(2) \\ \vdots \\ H(N) \end{array} x \right]^H \left[\begin{array}{ccccc} R(1) & 0 & \dots & 0 & 0 \\ 0 & R(2) & 0 & \dots & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & 0 & R(N) \end{array} \right]^{-1} \times \left[\begin{array}{c} z(1) \\ z(2) \\ \vdots \\ z(N) \end{array} - \begin{array}{c} H(1) \\ H(2) \\ \vdots \\ H(N) \end{array} x \right] \quad (10.1.2)$$

The cost function is defined over N iterations of the filter and can be more compactly written as

$$J(N) = \frac{1}{2} [z^N - H^N x]^H [R^N]^{-1} [z^N - H^N x] \quad (10.1.3)$$

where $J(N)$ is a scalar, z^N is Nn_z rows by 1 column ($Nn_z \times 1$), H^N is ($Nn_z \times n_x$), R^N is ($Nn_z \times Nn_z$), and x is ($n_x \times 1$). Clearly, R^N can be seen as $E\{w(t)w(t)^H\}$ giving a diagonal matrix for uncorrelated Gaussian white noise. Instead of solving for the FIR filter coefficients which best relates an input signal and output signal in the block

least-squares algorithm presented in Section 8.1, we are solving for the optimum state vector which fits the measurements $z(t)$. Granted, this may not be very useful for sinusoidal states unless the frequency is very low such that the N measurements cover only a small part of the wavelength. It could also be said that for digital dynamic system modeling, one must significantly over sample the time updates for Kalman filters to get good results. This was presented for a mass-spring dynamic system in Section 4.1.

The cost function is a quadratic function of the state vector to be optimized as well as the error. Differentiating with respect to the state, we have

$$\begin{aligned}\frac{\partial J(N)}{\partial x} &= \frac{1}{2} [H^N]^H [R^N]^{-1} [z^N - H^N x] - \frac{1}{2} [z^N - H^N x]^H [R^N]^{-1} [H^N] \\ &= [H^N]^H [R^N]^{-1} [H^N] x - [H^N]^H [R^N]^{-1} z^N\end{aligned}\quad (10.1.4)$$

where a second derivative with respect to x is seen to be positive definite, indicating a “concave up” error surface. The minimum error is found by solving for the least squared error estimate for the state vector based on N observations, $x'(N)$, which gives a zero first derivative in Eq. (10.1.4).

$$x'(N) = \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1} [H^N]^H [R^N]^{-1} z^N \quad (10.1.5)$$

The matrix term in the braces which is inverted in Eq. (10.1.5) can be seen as the covariance matrix of the state vector x prediction error. The state prediction error is simply

$$\begin{aligned}e(N) &= x - x'(N) = x - \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1} [H^N]^H [R^N]^{-1} [H^N x + w^N] \\ &= x - [H^N]^{-1} [R^N] [H^N]^{-H} [H^N]^H [R^N]^{-1} [H^N x + w^N] \\ &= \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1} [H^N]^H [R^N]^{-1} w^N\end{aligned}\quad (10.1.6)$$

making the covariance matrix of the state prediction error

$$\begin{aligned}P(N) &= E\{e(N)e(N)^H\} \\ &= \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1} [H^N]^H [R^N]^{-1} R^N [R^N]^{-1} H^N \\ &\quad \times \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1} \\ &= \left\{ [H^N]^H [R^N]^{-1} [H^N] \right\}^{-1}\end{aligned}\quad (10.1.7)$$

For a recursive update, we now augment the vectors and matrices in (10.1.2)–(10.1.3) for the $N+1$ st iteration and write the state prediction error as

an inverse covariance matrix.

$$\begin{aligned}
 P(N+1)^{-1} &= [H^{N+1}]^H [R^{N+1}]^{-1} [H^{N+1}] \\
 &= [H^{NH} H(N+1)^H] \begin{bmatrix} R^N & 0 \\ 0 & R(N+1) \end{bmatrix}^{-1} \begin{bmatrix} H^N \\ H(N+1) \end{bmatrix} \\
 &= [H^N]^H [R^N]^{-1} [H^N] + H(N+1)^H R(N+1)^{-1} H(N+1) \\
 &= P(N)^{-1} + H(N+1)^H R(N+1)^{-1} H(N+1)
 \end{aligned} \tag{10.1.8}$$

Equation (10.1.8) is of the form where the matrix inversion lemma (Section 9.1) can be applied to produce a recursive update for $P(N+1)$, rather than its inverse.

$$\begin{aligned}
 P(N+1) &= P(N) - P(N)H^H(N+1)[R(N+1) + H(N+1)P(N)H^H(N+1)]^{-1} \\
 &\quad \times H(N+1)P(N)
 \end{aligned} \tag{10.1.9}$$

The quantity in the square brackets which is inverted is called the covariance of the measurement prediction error. The measurement prediction error is found by combining Eqs (10.1.1) and (10.1.5).

$$\begin{aligned}
 \zeta(N+1) &= z(N+1) - z'(N+1) \\
 &= z(N+1) - H(N+1)x'(N) \\
 &= H(N+1)x + w(N+1) - H(N+1)x'(N) \\
 &= H(N+1)[x - x'(N)] + w(N+1) \\
 &= H(N+1)e(N) + w(N+1)
 \end{aligned} \tag{10.1.10}$$

The covariance of the measurement prediction error, $S(N+1)$, (also known as the covariance of the innovation), is seen to be

$$\begin{aligned}
 S(N+1) &= E\{\zeta(N+1)\zeta^H(N+1)\} \\
 &= E\{[H(N+1)e(N) + w(N+1)][e^H(N)H^H(N+1) + w^H(N+1)]\} \\
 &= H(N+1)P(N)H^H(N+1) + R(N+1)
 \end{aligned} \tag{10.1.11}$$

We can also define the update gain, or Kalman gain, for the recursion as

$$K(N+1) = P(N)H^H(N+1)[S(N+1)]^{-1} \tag{10.1.12}$$

where Eqs (10.1.11) and (10.1.12) are used to simplify and add more intuitive meaning to the state error covariance inverse update in Eq. (10.1.9).

$$P(N+1) = P(N) - K(N+1)S(N+1)K^H(N+1) \tag{10.1.13}$$

There are several other forms the update for the inverse state error covariance. The

state vector update is

$$\begin{aligned}
 x'(N+1) &= P(N+1)[H^{N+1}]^H[R^{N+1}]^{-1}z^{N+1} \\
 &= P(N+1)[H^{NH}H(N+1)^H]\begin{bmatrix} R^N & 0 \\ 0 & R(N+1) \end{bmatrix}^{-1}\begin{bmatrix} z^N \\ z(N+1) \end{bmatrix} \\
 &= P(N+1)[H^N]^H[R^N]^{-1}z^N \\
 &\quad + P(N+1)H(N+1)^HR(N+1)^{-1}z(N+1) \\
 &= [I - K(N+1)H(N+1)]P(N)[H^N]^H[R^N]^{-1}z^N \\
 &\quad + P(N+1)H(N+1)^HR(N+1)^{-1}z(N+1) \\
 &= [I - K(N+1)H(N+1)]x'(N) + K(N+1)z(N+1)
 \end{aligned} \tag{10.1.14}$$

where it is straightforward to show that $P(N+1) = [I - K(N+1)H(N+1)]P(N)$ and that the Kalman gain can be written as $K(N+1) = P(N+1)H^H(N+1)R^{-1}(N+1)$. The update recursion for the state vector can now be written in the familiar form of old state vector minus the Kalman gain times an error.

$$x'(N+1) = x'(N) + K(N+1)\zeta(N+1) \tag{10.1.15}$$

All we have done here is apply the recursive least squares algorithm to a state vector, rather than an FIR coefficient vector, as seen in Section 9.1. The basis function for the filter here is the measurement matrix defined in Eq. (10.1.1), and the filter outputs are the measurement predictions defined in Eq. (10.1.10). The RLS algorithm solves for the state vector which gives the least-squared error for measurement predictions. As seen in Eq. (10.1.5), the least-squares solution normalizes the known measurement error R^N , and as seen in Eq. (10.1.7), as the number of observations N becomes large, the variance of the state error becomes smaller than the variance of the measurement error. This smoothing action of the RLS algorithm on the state vector is a valuable technique to obtain good estimates of a state trajectory, such as velocity or acceleration, given a set of noisy measurements. A summary of the RLS state vector filter is given in Table 1.

The Kalman filter is really just a recursive kinematic time update form of an RLS filter for dynamic state vectors. Examples of dynamic state vectors include the position, velocity, acceleration, jerk, etc., for a moving object such as a satellite or aircraft. However, any dynamic data, such as process control information (boiler temperature, pressure, etc.), or even mechanical failure trajectories can be tracked using a dynamic state Kalman filter. The Kalman filter updates for the dynamic state vector are decidedly different from RLS, which is why we will refer to the dynamic state vector RLS filter specifically as Kalman filtering. It is assumed that an underlying kinematic model exists for the time update of the state vector. Using Newtonian physics, the kinematic model for a position-velocity-acceleration type state vector is well known. The modeled state vector update can be made for time $N+1$ using information available at time N .

$$x(N+1) = F(N+1)x(N) + v(N) + G(N)u(N) \tag{10.1.16}$$

Table 1 The RLS State Vector Filter

Description	Equation
Basis function (measurement matrix)	$H(N+1)$
Kalman gain update	$K(N+1) = \frac{P(N)H^H(N+1)}{R(N+1) + H(N+1)P(N)H^H(N+1)}$ $= P(N)H^H(N+1)[S(N+1)]^{-1}$
Inverse autocorrelation matrix update	$P(N+1) = (H^{N+1H} [R^N]^{-1} H^{N+1})^{-1}$ $= P(N) - K(N+1)S(N+1)K^H(N+1)$
Measurement prediction error	$\zeta(N+1) = z(N+1) - z'(N+1)$ $= z(N+1) - H(N+1)x'(N)$
Optimal state vector update using prediction error	$x'(N+1) = x'(N) + K(N+1)\zeta(N+1)$

The term $v(N)$ in Eq. (10.1.16) represents the *process noise* of the underlying kinematic model for the state vector updates, while $G(N)u(N)$ represent a control signal input, which will be left to Part V of this text. The matrix $F(N+1)$ is called *the state transition matrix* and is derived from the kinematic model. As with the measurement matrix $H(N+1)$, usually the state transition matrix is constant with time, but we allow the notation to support a more general context. Consider the case of a state vector comprised of a position, velocity, and acceleration. Proceeding without the control input, we note that our best “a priori” (meaning before the state is updated with the least-squared error recursion) update for the state at time $N+1$, given information at time N is

$$\begin{aligned}
 x'(N+1|N) &= F(N)x(N|N) \\
 &= \begin{bmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{N|N} \\ \dot{x}_{N|N} \\ \ddot{x}_{N|N} \end{bmatrix}
 \end{aligned} \tag{10.1.17}$$

where T is the time interval of the update in seconds. The process noise is not included in the a priori state vector prediction since the process noise is assumed uncorrelated from update to update. However, the state prediction error variance must include the process noise covariance. A typical kinematic model for a position, velocity, and acceleration state vector would assume a random change in acceleration from state update to state update. This corresponds to a constant white jerk variance (not to be confused with a relentless obnoxious Caucasian (see Section 4.2)) which results in an acceleration which is an integral of the zero-mean Gaussian white jerk. Assuming a piecewise constant acceleration with a white jerk allows the state transition in Eq. (10.1.17) where the process noise variance for the acceleration is $E\{v(N)v^H(N)\} = \sigma_v^2$. The acceleration process variance scales to velocity by a factor of T , and to position by a factor of $\frac{1}{2}T^2$ leading to the process noise covariance

$Q(N)$ in Eq. (10.1.18).

$$Q(N) = \begin{bmatrix} \frac{1}{2}T^2 \\ T \\ 1 \end{bmatrix} \sigma_v^2 \begin{bmatrix} \frac{1}{2}T^2 & T & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^3 & T^2 & T \\ \frac{1}{2}T^2 & T & 1 \end{bmatrix} \sigma_v^2 \quad (10.1.18)$$

The process noise Q , state transition F and measurement matrix H are usually constants, but our notation allows time variability. If one were to change the update rate T for the Kalman filter, Q must be re-scaled. There are quite a number of process noise assumptions which can be implemented to approximate the real situation for the state kinematics. Bar-Shalom and Li (2) provide a detailed analysis of various process noise assumptions. An a priori state error covariance can be written as

$$P'(N+1|N) = F(N)P(N|N)F^H(N) + Q(N) \quad (10.1.19)$$

The a priori state error estimate is then used to update the innovation covariance

$$S(N+1) = H(N+1)P'(N+1|N)H^H(N+1) + R(N+1) \quad (10.1.20)$$

and finally to update the Kalman gain.

$$K(N+1) = P'(N+1|N)H^H(N+1)[S(N+1)]^{-1} \quad (10.1.21)$$

Given the latest measurement $z(N+1)$ at time $N+1$, an a posteriori update of the least-squared error state vector and state vector error covariance are completed as given in Eqs (10.1.22) and (10.1.23).

$$x(N+1|N+1) = x'(N+1|N) + K(N+1)\{z(N+1) - H(N+1)x'(N+1|N)\} \quad (10.1.22)$$

$$P(N+1|N+1) = P'(N+1|N) - K(N+1)S(N+1)K^H(N+1) \quad (10.1.23)$$

Equations (10.1.17)–(10.1.23) constitute a typical Kalman filter for tracking a parameter along with its velocity and acceleration as a function of time. There are countless applications and formulations of this fundamental processing solution which can be found in many excellent texts in detail well beyond the scope of this book. But, it is extremely useful to see a derivation of the Kalman filter in the context of the more general RLS algorithm, as well as its similarities and differences with other commonly used adaptive filtering algorithms.

One of the more useful aspects of Kalman filtering is found not in its ability to smooth the state vector (by reducing the affect of measurement noise), but rather in its predictive capabilities. Suppose we are trying to intercept an incoming missile and are tracking the missile trajectory from a series of position measurements obtained from a radar system. Our anti-missile requires about 30 sec lead time to fly into the preferred intercept area. The preferred intercept area is determined by the highest probability of hit along a range of possible intercept positions defined by the trajectories of the two missiles. Equations (10.1.17)–(10.1.19) are used to determine the incoming missile trajectory and variance of the trajectory error, which in three dimensions is an ellipsoid centered around a given predicted future position

for the incoming missile. The further out in time one predicts the incoming missile's position, the larger the size of prediction error ellipsoid becomes. The state error prediction can be seen in this case as a three-dimensional Gaussian probability density "cloud", where the ellipsoid is a one-standard deviation contour. Using Baye's rule and other hypothesis scoring methods, one can solve the kinematic equations in a computer for the launch time giving rise to the most likely intercept. This is not always easy nor straightforward, since the tracking filter is adaptive and the states and measurement noise are dynamically changing with time.

Some of the best evidence of the difficulty of missile defense was seen with the Patriot system during the Gulf War of 1991. To be fair, the Patriot was designed to intercept much slower aircraft, rather than ballistic missiles. But, in spite of the extreme difficulty of the task, the missile defense system brought to the attention of the general public the concept of an intelligent adaptive homing system using Kalman filters. Track-intercept solutions are also part of aircraft collision avoidance systems and may soon be part of intelligent highway systems for motor vehicles. Using various kinematic models, Kalman filters have been used routinely in financial markets, prediction of electrical power demand, and in biological/environmental models for predicting the impact of pollution and species populations. If one can describe a phenomena using differential equations and measure quantities systematically related to those equations, tracking and prediction filters can be of great utility in the management and control of the phenomena.

Table 2 summarizes the Kalman filter. The type of measurements used determine the measurement matrix $H(N+1)$ and measurement noise covariance $R(N+1)$. The state kinematic model determines the process noise covariance $Q(N+1)$ and state transition matrix $F(N+1)$. As noted earlier, H , R , Q , F , σ_v^2 , σ_w^2 , and T are typically constant in the tracking filter algorithm. As can be seen from Table 2, the Kalman filter algorithm is very straightforward. One makes a priori state and state covariance predictions, measures the resulting measurement prediction error, and adjusts the state updates according to the error and the define measurement and process noise variances. However, the inverse state error covariance update given in the table can be reformulated to promote lower numerical roundoff error. The expression in Eq. (10.1.24) algebraically equivalent to that given in

$$P(N+1|N) = [I - K(N+1)H(N+1)]P(N+1|N)[I - K(N+1)H(N+1)]^H + K(N+1)R(N+1)K(N+1)^H \quad (10.1.24)$$

Table 2, but its structure, known as Joseph form, promotes symmetry and reduces numerical noise.

An example of the Kalman filter is presented following the simple rocket height tracking example given in Section 4.2 for the α - β - γ tracker. As will be seen below, the α - β - γ tracking filter is a simple Kalman filter where the Kalman gain has been fixed to the optimal value assuming a constant state error covariance. In the rocket example, we have height measurements every 100 msec with a standard deviation of 3 m. The maximum acceleration (actually a deceleration) occurs at burnout due to both gravity and drag forces and is about 13 m/sec². Setting the process noise standard deviation to 13 is seen as near optimal for the α - β - γ tracking filter, allowing

$Q(N)$ in Eq. (10.1.18).

$$Q(N) = \begin{bmatrix} \frac{1}{2}T^2 \\ T \\ 1 \end{bmatrix} \sigma_v^2 \begin{bmatrix} \frac{1}{2}T^2 & T & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{4}T^4 & \frac{1}{2}T^3 & \frac{1}{2}T^2 \\ \frac{1}{2}T^3 & T^2 & T \\ \frac{1}{2}T^2 & T & 1 \end{bmatrix} \sigma_v^2 \quad (10.1.18)$$

The process noise Q , state transition F and measurement matrix H are usually constants, but our notation allows time variability. If one were to change the update rate T for the Kalman filter, Q must be re-scaled. There are quite a number of process noise assumptions which can be implemented to approximate the real situation for the state kinematics. Bar-Shalom and Li (2) provide a detailed analysis of various process noise assumptions. An a priori state error covariance can be written as

$$P'(N+1|N) = F(N)P(N|N)F^H(N) + Q(N) \quad (10.1.19)$$

The a priori state error estimate is then used to update the innovation covariance

$$S(N+1) = H(N+1)P'(N+1|N)H^H(N+1) + R(N+1) \quad (10.1.20)$$

and finally to update the Kalman gain.

$$K(N+1) = P'(N+1|N)H^H(N+1)[S(N+1)]^{-1} \quad (10.1.21)$$

Given the latest measurement $z(N+1)$ at time $N+1$, an a posteriori update of the least-squared error state vector and state vector error covariance are completed as given in Eqs (10.1.22) and (10.1.23).

$$x(N+1|N+1) = x'(N+1|N) + K(N+1)\{z(N+1) - H(N+1)x'(N+1|N)\} \quad (10.1.22)$$

$$P(N+1|N+1) = P'(N+1|N) - K(N+1)S(N+1)K^H(N+1) \quad (10.1.23)$$

Equations (10.1.17)–(10.1.23) constitute a typical Kalman filter for tracking a parameter along with its velocity and acceleration as a function of time. There are countless applications and formulations of this fundamental processing solution which can be found in many excellent texts in detail well beyond the scope of this book. But, it is extremely useful to see a derivation of the Kalman filter in the context of the more general RLS algorithm, as well as its similarities and differences with other commonly used adaptive filtering algorithms.

One of the more useful aspects of Kalman filtering is found not in its ability to smooth the state vector (by reducing the affect of measurement noise), but rather in its predictive capabilities. Suppose we are trying to intercept an incoming missile and are tracking the missile trajectory from a series of position measurements obtained from a radar system. Our anti-missile requires about 30 sec lead time to fly into the preferred intercept area. The preferred intercept area is determined by the highest probability of hit along a range of possible intercept positions defined by the trajectories of the two missiles. Equations (10.1.17)–(10.1.19) are used to determine the incoming missile trajectory and variance of the trajectory error, which in three dimensions is an ellipsoid centered around a given predicted future position

for the incoming missile. The further out in time one predicts the incoming missile's position, the larger the size of prediction error ellipsoid becomes. The state error prediction can be seen in this case as a three-dimensional Gaussian probability density "cloud", where the ellipsoid is a one-standard deviation contour. Using Baye's rule and other hypothesis scoring methods, one can solve the kinematic equations in a computer for the launch time giving rise to the most likely intercept. This is not always easy nor straightforward, since the tracking filter is adaptive and the states and measurement noise are dynamically changing with time.

Some of the best evidence of the difficulty of missile defense was seen with the Patriot system during the Gulf War of 1991. To be fair, the Patriot was designed to intercept much slower aircraft, rather than ballistic missiles. But, in spite of the extreme difficulty of the task, the missile defense system brought to the attention of the general public the concept of an intelligent adaptive homing system using Kalman filters. Track-intercept solutions are also part of aircraft collision avoidance systems and may soon be part of intelligent highway systems for motor vehicles. Using various kinematic models, Kalman filters have been used routinely in financial markets, prediction of electrical power demand, and in biological/environmental models for predicting the impact of pollution and species populations. If one can describe a phenomena using differential equations and measure quantities systematically related to those equations, tracking and prediction filters can be of great utility in the management and control of the phenomena.

Table 2 summarizes the Kalman filter. The type of measurements used determine the measurement matrix $H(N+1)$ and measurement noise covariance $R(N+1)$. The state kinematic model determines the process noise covariance $Q(N+1)$ and state transition matrix $F(N+1)$. As noted earlier, H , R , Q , F , σ_v^2 , σ_w^2 , and T are typically constant in the tracking filter algorithm. As can be seen from Table 2, the Kalman filter algorithm is very straightforward. One makes a priori state and state covariance predictions, measures the resulting measurement prediction error, and adjusts the state updates according to the error and the define measurement and process noise variances. However, the inverse state error covariance update given in the table can be reformulated to promote lower numerical roundoff error. The expression in Eq. (10.1.24) algebraically equivalent to that given in

$$P(N+1|N) = [I - K(N+1)H(N+1)]P(N+1|N)[I - K(N+1)H(N+1)]^H + K(N+1)R(N+1)K(N+1)^H \quad (10.1.24)$$

Table 2, but its structure, known as Joseph form, promotes symmetry and reduces numerical noise.

An example of the Kalman filter is presented following the simple rocket height tracking example given in Section 4.2 for the α - β - γ tracker. As will be seen below, the α - β - γ tracking filter is a simple Kalman filter where the Kalman gain has been fixed to the optimal value assuming a constant state error covariance. In the rocket example, we have height measurements every 100 msec with a standard deviation of 3 m. The maximum acceleration (actually a deceleration) occurs at burnout due to both gravity and drag forces and is about 13 m/sec². Setting the process noise standard deviation to 13 is seen as near optimal for the α - β - γ tracking filter, allowing

Table 2 The Kalman Filter

Description	Equation
Basis function (measurement matrix)	$H(N+1)$
State transition matrix	$F(N+1)$
Process noise covariance matrix	$Q(N+1)$
Measurement noise covariance matrix	$R(N+1)$
A priori state prediction	$x'(N+1 N) = F(N)x(N N)$
A priori state error prediction	$P'(N+1 N) = F(N)P(N)F^H(N) + Q(N)$
A priori measurement prediction	$z'(N+1) = H(N+1)x'(N+1 N)$
Innovation covariance update	$S(N+1) = H(N+1)P'(N+1 N)H^H(N+1) + R(N+1)$
Kalman gain update	$K(N+1) = P'(N+1 N)H^H(N+1)[S(N+1)]^{-1}$
Measurement prediction error	$\zeta(N+1) = z(N+1) - H(N+1)x'(N+1 N)$
Optimal state vector update using prediction error	$x(N+1 N+1) = x'(N+1 N) + K(N+1)\zeta(N+1)$
Inverse state covariance matrix update	$P(N+1 N+1) = P'(N+1 N) - K(N+1)S(N+1)K^H(N+1)$

non-adaptive tracking of the changes in acceleration, velocity, and position of the rocket's height. The adaptive Kalman gain, on the other hand, depends on the ratio of the state prediction error covariance to the measurement noise covariance. The Kalman gain will be high when the state error is large and the measurements accurate (σ_w and R small), and the gain will be low when the state error is small and/or the measurements inaccurate. The adaptive gain allows the Kalman filter to operate with a lower process noise than the non-adaptive α - β - γ tracking filter, since the gain will automatically increase or decrease with the state prediction error. In other words, if one knows the maneuverability of the target (the maximum acceleration, for example), and the measurements are generally noisy, the α - β - γ tracking filter is probably the best choice, since the measurement noise will not effect the filter gain too much. But if one does not know the target maneuverability, the Kalman filter offers fast convergence to give the least-squared error state vector covariance. However, when the measurements are noisy, the Kalman filter also suffers from more noise in the state vector estimates when the same process noise is used as in the α - β - γ tracking filter. Figures 1 and 2 illustrate the noise sensitivity for the rocket example given in Section 4.2.

Comparing the filter gains used in Figures 1 and 2 we see that the α - β - γ tracking filter used [0.2548 0.3741 0.2746] while the Kalman filter calculated a gain vector of [0.5046 1.7545 3.0499]. The higher Kalman gain fit the state vector more accurately to the measurements, but resulted in more velocity and acceleration noise. In the simulation, if we let the measurement noise approach zero, the responses of

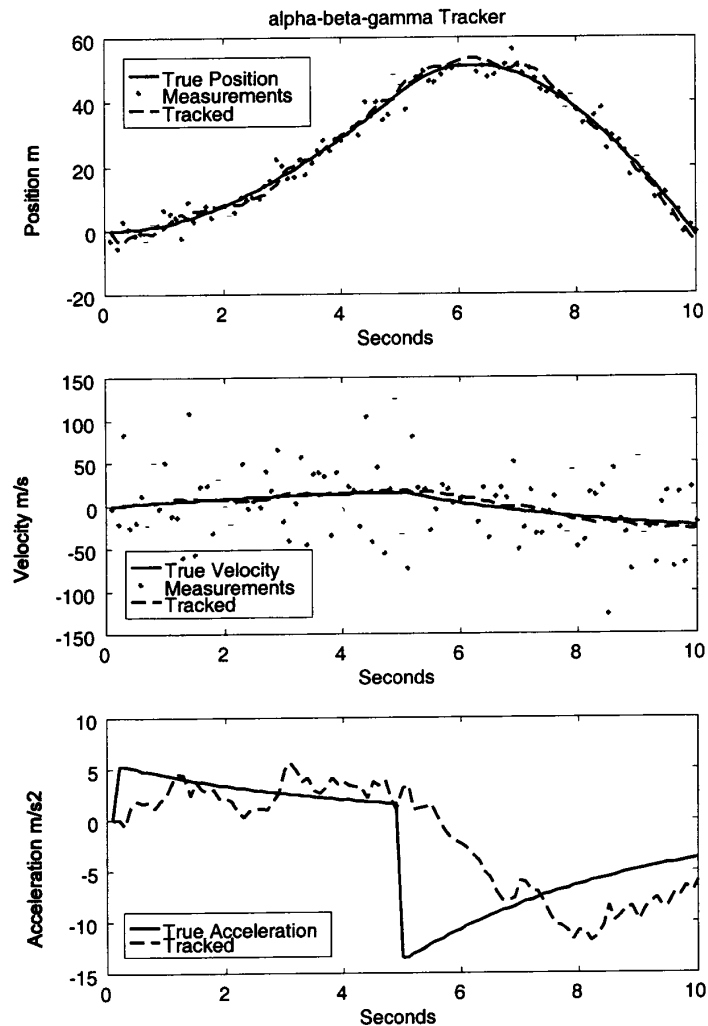


Figure 1 Non-adaptive α - β - γ tracking filter (from Section 4.2) where the measurement noise is 3 and process noise 13.

the two filters become identical and the gain vector converges to $[1 \ 1/T \ 1/(2T^2)]$, or $[1 \ 20 \ 200]$ when $T = 100$ msec. By comparing the gain vectors for a given measurement noise (and process noise for the α - β - γ tracking filter), we can adjust the Kalman filter process noise so that the filter gains are roughly the same when the measurement noise is still a relatively high 3 m. We find that the Kalman filter process noise can be reduced to about 1.0 to give a Kalman gain vector of $[0.2583 \ 0.3851 \ 0.2871]$, which is very close to the gain vector for the α - β - γ tracking filter. Figure 3 shows

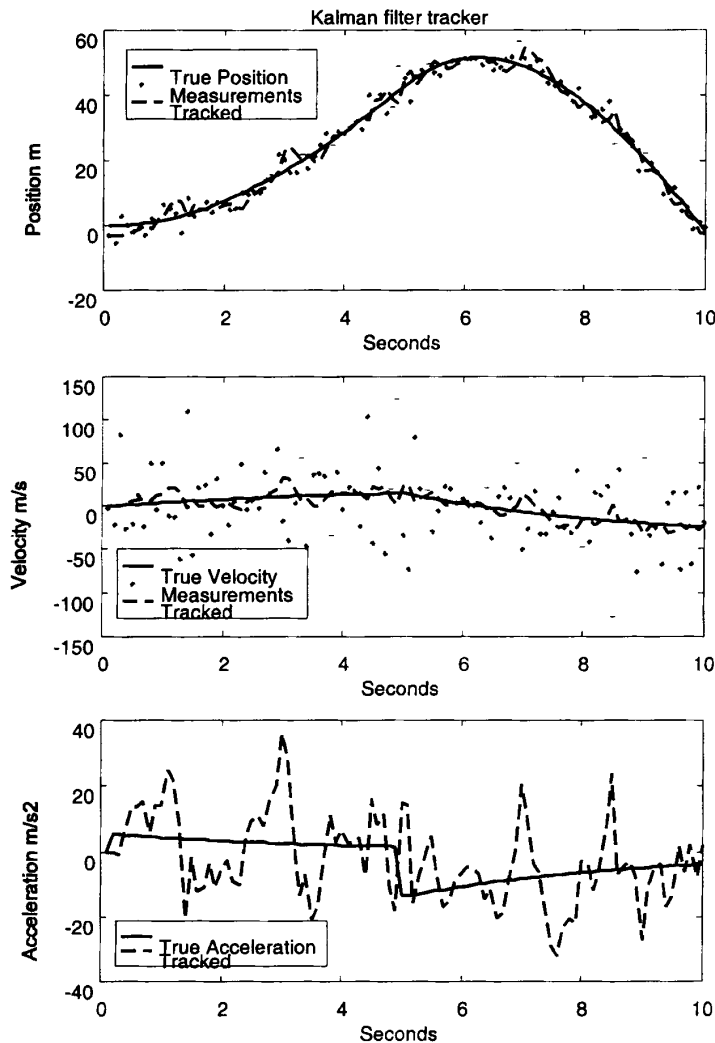


Figure 2 Kalman filter for the same measurement and process noise as seen in Figure 1.

the Kalman filter tracking results which are nearly identical to the α - β - γ tracking filter with a high process noise of 13.

The significance of determining the Kalman filter process noise which gives the same filter gain as the α - β - γ tracking filter when the measurement noise is high is that this gives an indication of the minimum process noise needed to track the target maneuvers without significant overshoot. Note that for a near zero measurement noise, the gain vectors are only identical for the same process noise. Given a reasonable minimum process noise for the Kalman filter, it gives superior tracking per-

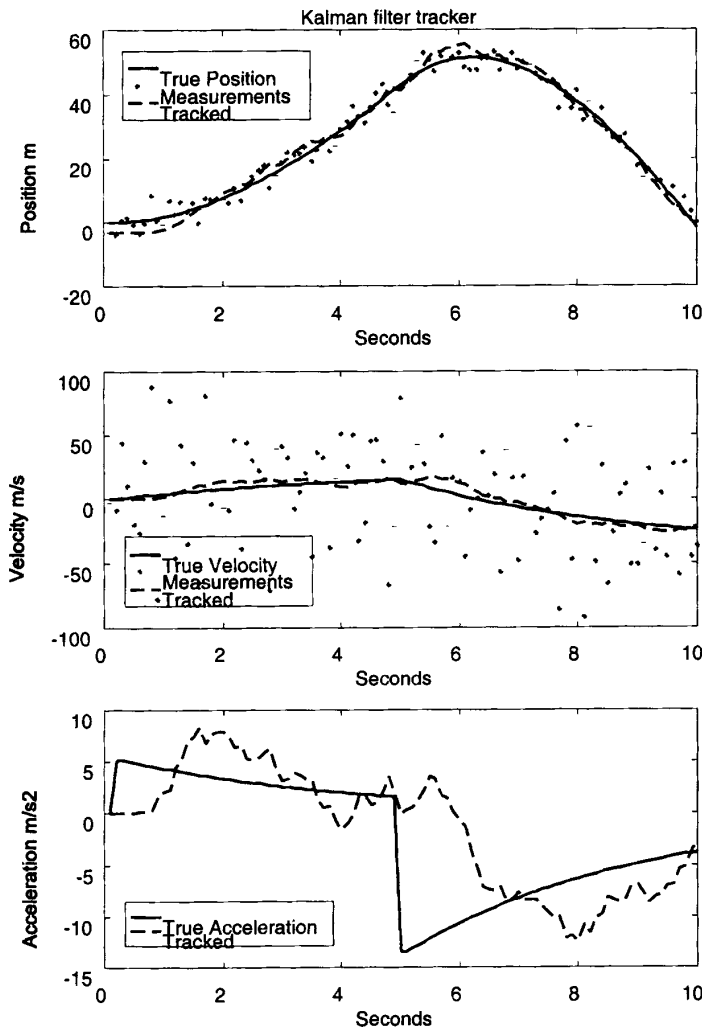


Figure 3 Kalman filter results using the same measurement data as seen in Figures 1–2, but with the process noise of 1.

formance to the α - β - γ tracking filter when the measurement noise is low as seen in Figures 4 and 5.

The examples in Figures 1–5 and in Section 4.2 indicate that the α - β - γ tracking filter is the algorithm of choice when the measurements are noisy. Unfortunately, there is not an algebraic expression which can tell us simply how to set the process noise in a Kalman filter for optimum performance. The reason is that the Kalman gain is not only a function of measurement noise, process noise, and time update

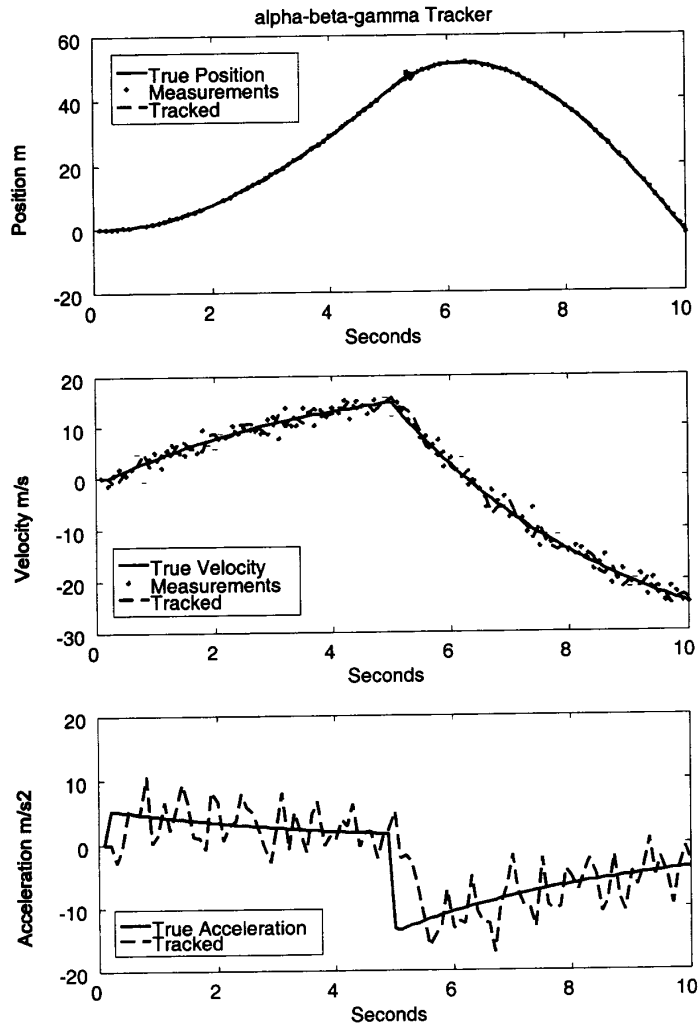


Figure 4 Low measurement noise (0.1) improves the α - β - γ tracking filter with the same process noise of 13.

(as is the case with the α - β - γ tracking filter), but it also depends on the state prediction error covariance, which is a function of target maneuvers. In most situations, one knows a reasonable expected target track maneuverability and the expected measurement noise a priori. This information allows one to execute simulations for a covariance analysis of the Kalman filter's performance on the expected data. With the measurement noise small (or near zero), one can set the Kalman filter process noise to almost any large value and the filter will perform nearly perfectly. With

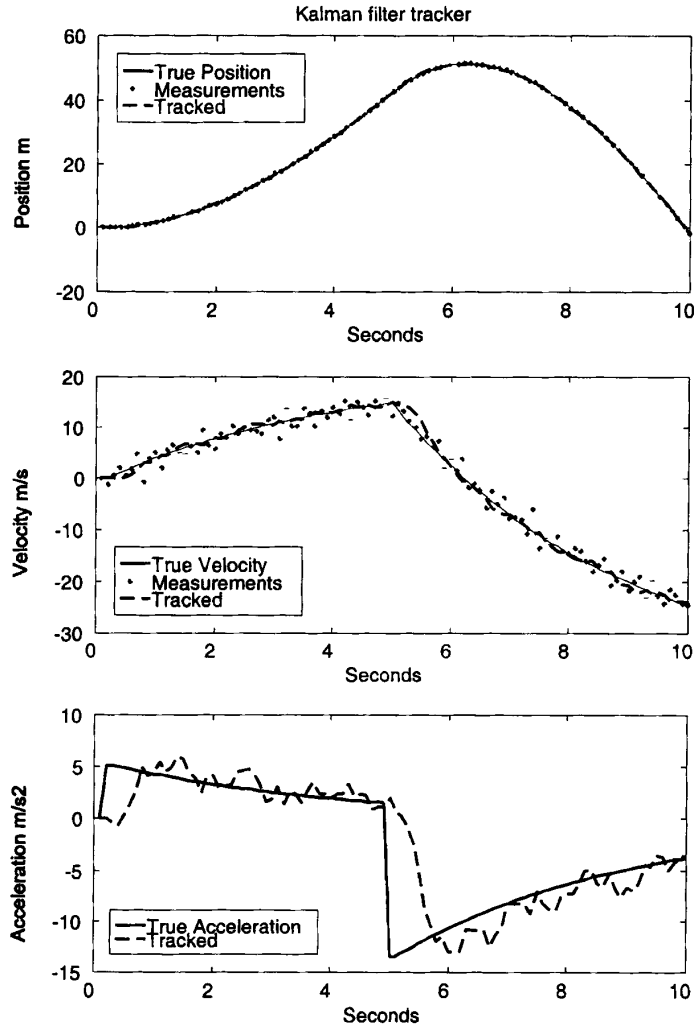


Figure 5 The Kalman filter with 0.1 measurement noise and a process noise of 1 gives improved performance over the α - β - γ tracking filter for the same data and a process noise of 13.

high measurement noise, it is easier to determine the minimum Kalman process noise empirically for a given target maneuver. Setting the process noise to the maximum expected acceleration is a good starting point (and ending point for the α - β - γ tracking filter), but the Kalman filter generally allows a lower process noise due to the enhanced ability of the adaptive updates to help the state error converge rapidly to target maneuvers.

The α - β - γ tracking filter is a Kalman filter with fixed state error covariance. In Section 4.2 we introduced the α - β - γ tracking filter as an important application of a state-variable filter and its derivation was referenced to this Section. Its derivation is rather straightforward, but algebraically tedious. We show the approach of Bar-Shalom (3) to demonstrate the technique. But more importantly, the solution for the α - β - γ tracking filter shows that α , β , and γ cannot be chosen independently. For a given amount of position noise reduction (as determined by $\alpha < 1$), β and γ are systematically determined from the measurement error σ_w and update time T . Setting α therefore also sets the process noise, and the state prediction error covariances to constant "steady-state" values. However, the filter will only converge to the prescribed steady-state error covariance if all the kinematic assumptions are true. If the tracking target makes an unexpected maneuver, the state error will increase for a while as the state vectors "overshoot" the measurements temporarily. This can be seen in Figure 4.5 where a "sluggish" track is produced from using too small a process noise in the α - β - γ tracking filter.

Combining Eqs (10.1.19), (10.1.21) and (10.1.23) we can write an expression for the updated state prediction error covariance P assuming steady-state conditions.

$$P = F^{-1}(P' - Q)F^{-H} = (I - KH)P' \quad (10.1.25)$$

If we simplify the problem by assuming only the α and β tracking gains and a piecewise constant acceleration model, we have the following matrix equation to solve.

$$\begin{bmatrix} 1 & -T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p'_{11} - \frac{T^4}{4} \sigma_v^2 & p'_{12} - \frac{T^3}{2} \sigma_v^2 \\ p'_{21} - \frac{T^3}{2} \sigma_v^2 & p'_{22} - T^2 \sigma_v^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -T & 1 \end{bmatrix} = \begin{bmatrix} 1 - k_1 & 0 \\ -k_2 & 1 \end{bmatrix} \begin{bmatrix} p'_{11} & p'_{12} \\ p'_{21} & p'_{22} \end{bmatrix} \quad (10.1.26)$$

Equation (10.1.26) is best solved by simply equating terms. After some simplification, we have

$$k_1 p'_{11} = 2T p'_{12} - T^2 p'_{22} + \frac{T^4}{4} \sigma_v^2 \quad (10.1.27)$$

$$k_1 p'_{12} = T^2 p'_{22} - \frac{T^3}{2} \sigma_v^2 \quad (10.1.28)$$

and

$$k_2 p_{12} = T^2 \sigma_v^2 \quad (10.1.29)$$

Solving for the predicted state covariance elements we have

$$p_{11} = \frac{k_1}{1 - k_1} \sigma_w^2 \quad (10.1.30)$$

and

$$p'_{12} = p'_{21} = \frac{k_2}{1 - k_1} \sigma_w^2 \quad (10.1.31)$$

Finally, solving for the p'_{22} term we have

$$p'_{22} = \left(\frac{k_1}{T} + \frac{k_2}{2} \right) p'_{12} \quad (10.1.32)$$

Combining Eqs (10.1.27)–(10.1.32) and after some algebraic cancellations, we arrive at the following bi-quadratic equation for the filter gains.

$$k_1^2 - 2Tk_2 + Tk_1k_2 + \frac{T^2}{4}k_2^2 = 0 \quad (10.1.33)$$

The filter gains in terms of α and β are simply $k_1 = \alpha$ and $k_2 = \beta/T$. Equation (10.1.33) reduces to

$$\alpha^2 - 2\beta + \alpha\beta + \frac{\beta^2}{4} = 0 \quad (10.1.34)$$

Solving for α in terms of β using the quadratic formula we have

$$\alpha = \sqrt{2\beta} - \frac{\beta}{2} \quad (10.1.35)$$

and solving for β in terms of α we have

$$\beta = 4 - 2\alpha - 4\sqrt{1 - \alpha} \quad (10.1.36)$$

For the α - β - γ filter, it can be shown (with some algebraic complexity) that $\gamma = \beta^2/\alpha$. Another important relation is seen when equating terms for p'_{12} .

$$\begin{aligned} p'_{12} &= \frac{T^2\sigma_v^2}{k_2} = \frac{k_2}{1 - k_1} \sigma_w^2 \\ \frac{T^2\sigma_v^2}{\frac{\beta}{T}} &= \frac{\frac{\beta}{T}}{1 - \alpha} \sigma_w^2 \end{aligned} \quad (10.1.37)$$

Equation (10.1.37) leads to an expression for the track maneuverability index λ_M .

$$\lambda_M = \frac{T^2\sigma_v}{\sigma_w} \quad (10.1.38)$$

The results of the derivation above are also presented in Eqs (4.2.7)–(4.2.11). The maneuverability index determines the “responsiveness” of the tracking filter to the measurements. For the fixed gain α - β - γ filter one estimates the maximum target acceleration to determine the process noise standard deviation σ_v . Given the measurement noise standard deviation σ_w and track update time T , the maneuverability index λ_M is determined as well as α , β , and γ (see Section 4.2).

The α - β - γ filter is an excellent algorithm choice for cases where the kinematics are known and the measurements are relatively noisy. Given unknown target maneuverability and/or low noise measurements, the adaptive Kalman filter is a better tracking algorithm choice, because the process noise can be significantly lowered without sacrificing responsiveness to give more noise reduction in the tracking state outputs. While derivation of the α - β - γ filter is quite tedious, knowledge of the Kalman filter equations allows a straightforward solution. Tracking filters are an extremely important and powerful signal processing tool. Whenever one measures a quantity with a known measurement error statistic, and observes the quantity changing deterministically over time, one is always interested in predicting when that quantity reaches a certain value, and with what statistical confidence.

10.2 IIR FORMS FOR LMS AND LATTICE FILTERS

The least-squared error system identification and signal modeling algorithms presented in Chapters 8 and 9 were limited to FIR filter structures. However, modeling systems and signals using infinite impulse response (IIR) digital filter is quite straightforward. Because an IIR filter with adaptive coefficients has the potential to become unstable, one must carefully constrain the adaptive IIR filter algorithm. If we are performing a system identification where both the input and output signals are completely known, stability is less of a concern provided that the unknown system to be identified is stable. When only the output signal is available and we are trying to model the signal as the output of an IIR filter driven by white noise, the stability issues can be considerably more difficult. Recall from Section 3.1 that FIR filters are often referred to as moving average, or MA, and are represented on the complex z -plane as a polynomial where the angles of the zeros determine the frequencies where the FIR filter response is attenuated, or has a spectral dip. In Section 3.2 the IIR filter is presented as a denominator polynomial in the z -domain. The angles of the zeros of the IIR polynomial determine the frequencies where the filter's response is amplified, or has a spectral peak. The IIR zeros are called the filter's poles because they represent the frequencies where the response has a peak. IIR filters with only a denominator polynomial feedback the past values of the output signal in the calculation of the current output, and as a consequence, are often called autoregressive, or AR filters.

The most general form of an IIR filter has both a numerator polynomial and a denominator polynomial, and is usually referred to as a pole-zero, or ARMA filter. The zeros of the numerator polynomial are the filter zeros and the zeros of the denominator polynomial are the filter poles. For the IIR filter to be stable, the magnitude of all the poles must be less than unity. This insures that the output feedback of some particular output signal sample eventually reverberates out to zero as it is fed back in the autoregressive generation of the current IIR output. Therefore, the denominator part of a stable IIR filter must be a minimum phase polynomial with all its zeros (the system poles) inside the unit circle.

Figure 6 depicts system identification of some unknown system represented by the ratio of z -domain polynomials $B[z]/A[z]$ where the zeros of $B[z]$ are the system zeros and the zeros of $A[z]$ are the system poles. The adaptive system identification is executed by a pair of adaptive filters which are synchronized by a common error signal $e[n]$, and the input $x[n]$ and output $y[n]$. Recall from Eq. (3.2.10) the difference