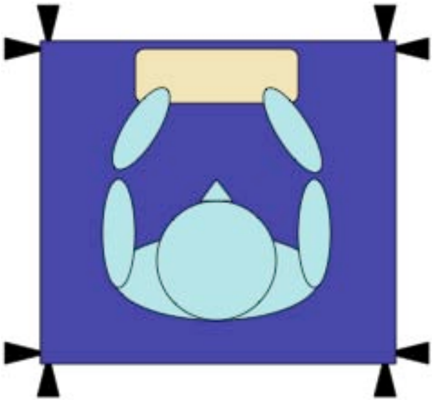
# SBE 2006 Dynamics Homework Solutions

***Preamble***

Imagine you are a test director at Johnson Space Center (JSC). You are interested in testing a training version of SAFER (Simplified Aid For EVA Rescue) to be used on the Precision Air Bearing Floor (PABF).

The training version of the SAFER is designed to work only in one plane (i.e., the plane of motion parallel to the plane of the floor). The SAFER can be controlled in two ways: translation or rotation, depending on which jets are active at any time. Balanced jets in one direction will result in a force applied in the opposite direction. Counteracting jets on either side of the SAFER will result in a torque about the center of mass of the SAFER. The SAFER uses “bang-off-bang” control, meaning each jet is either fully on or fully off. A diagram of the training SAFER can be found below.

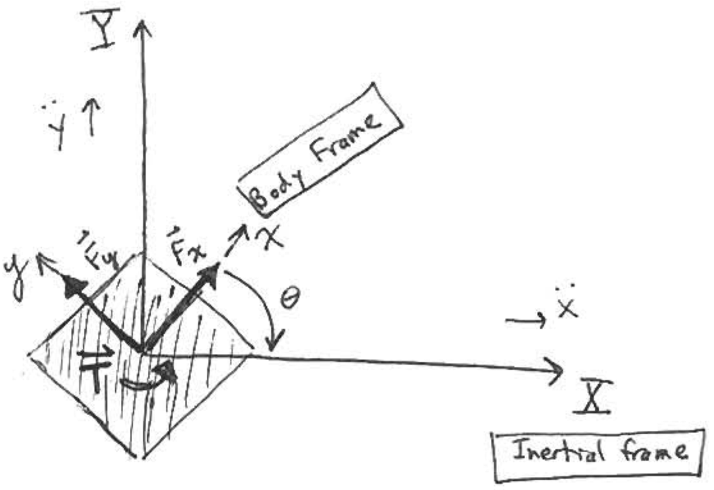


In order to evaluate the performance of this SAFER model, you want to track the motion of a subject wearing the SAFER training unit. To collect data for your test, you have setup a small, motion tracking system. The tracking system uses magnetic markers on the SAFER and provides noisy measurements of the planar X and Y positions as well as a rotation angle, theta, of the SAFER.

This problem explores the dynamics of this situation, simulates it and leads you through the development of a simple Kalman filter for tracking the position, velocity, angle and angular rate of a subject using this training version of the SAFER.

PROBLEM 1.

SAFER Dynamics



Assumptions:
- No friction
- Neglecting gravity

Applied inputs: Body force $\vec{F}_{body} = \begin{bmatrix} \vec{F}_x \\ \vec{F}_y \end{bmatrix}$

Torque $\vec{T}$

Dynamics:

$$\boxed{\vec{F}_{inert} = \begin{bmatrix} F_X \\ F_Y \end{bmatrix}}$$

(1) Rotate body force into inertial frame using the direction cosine matrix, $\underline{\underline{C}}$

$$\vec{F}_{inert} = \underline{\underline{C}} \, \vec{F}_{body} \text{ , where } \underline{\underline{C}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

(2) Apply Newton's laws:

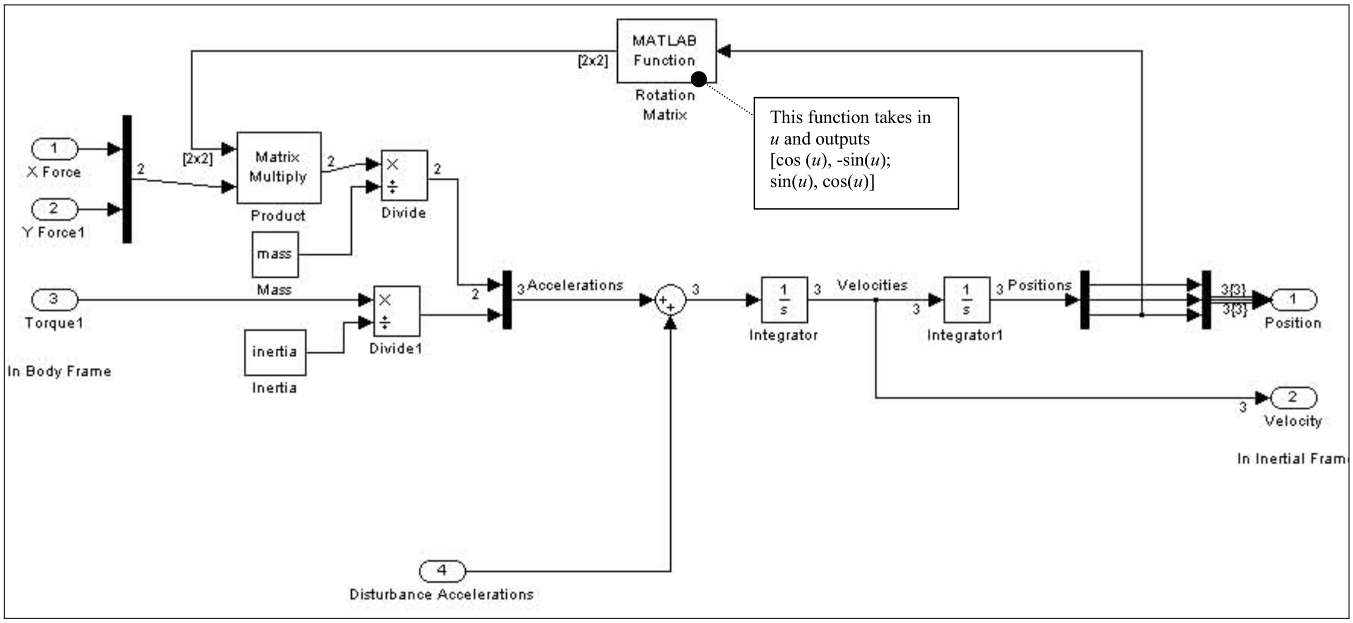
$$F_X = m\ddot{X} \Rightarrow \ddot{X} = \frac{1}{m} F_X$$

$$F_Y = m\ddot{Y} \Rightarrow \ddot{Y} = \frac{1}{m} F_Y$$

$$T = I\ddot{\theta} \Rightarrow \ddot{\theta} = \frac{1}{I} T$$

**Problem 2.**

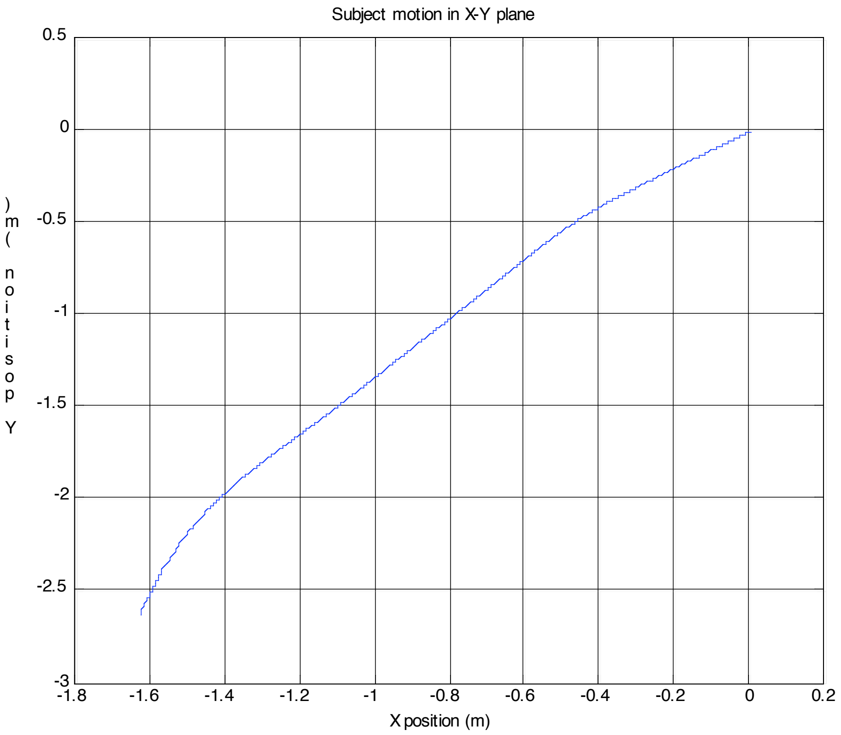
Simulink Model of SAFER plant dynamics

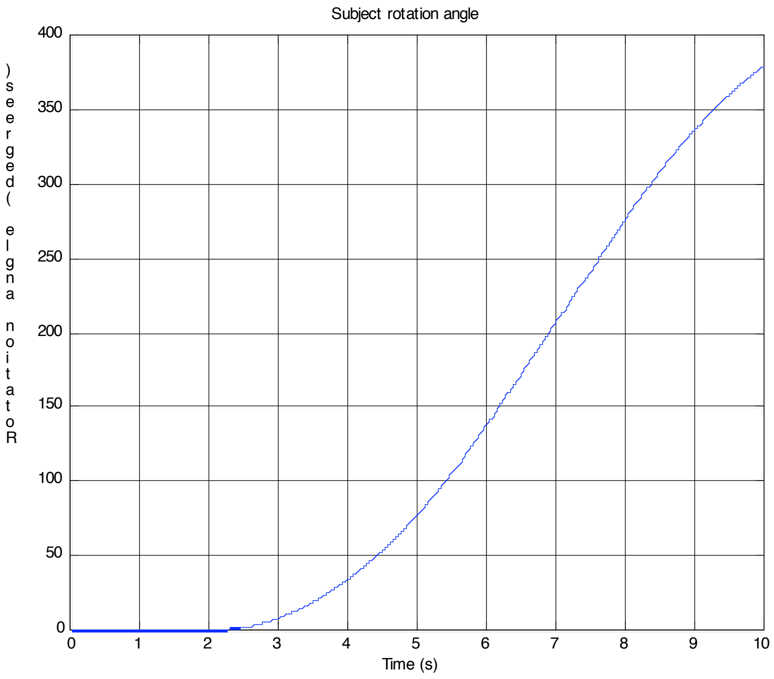


This function takes in *u* and outputs [cos (*u*), -sin(*u*); sin(*u*), cos(*u*)]

## Problem 2.

Plots of SAFER position and rotation angle





**Problems 3-5.**

After adding process noise and measurement noise into the Simulink model, we find a set of data that is less useful than the original ideal situation of Problem 2.

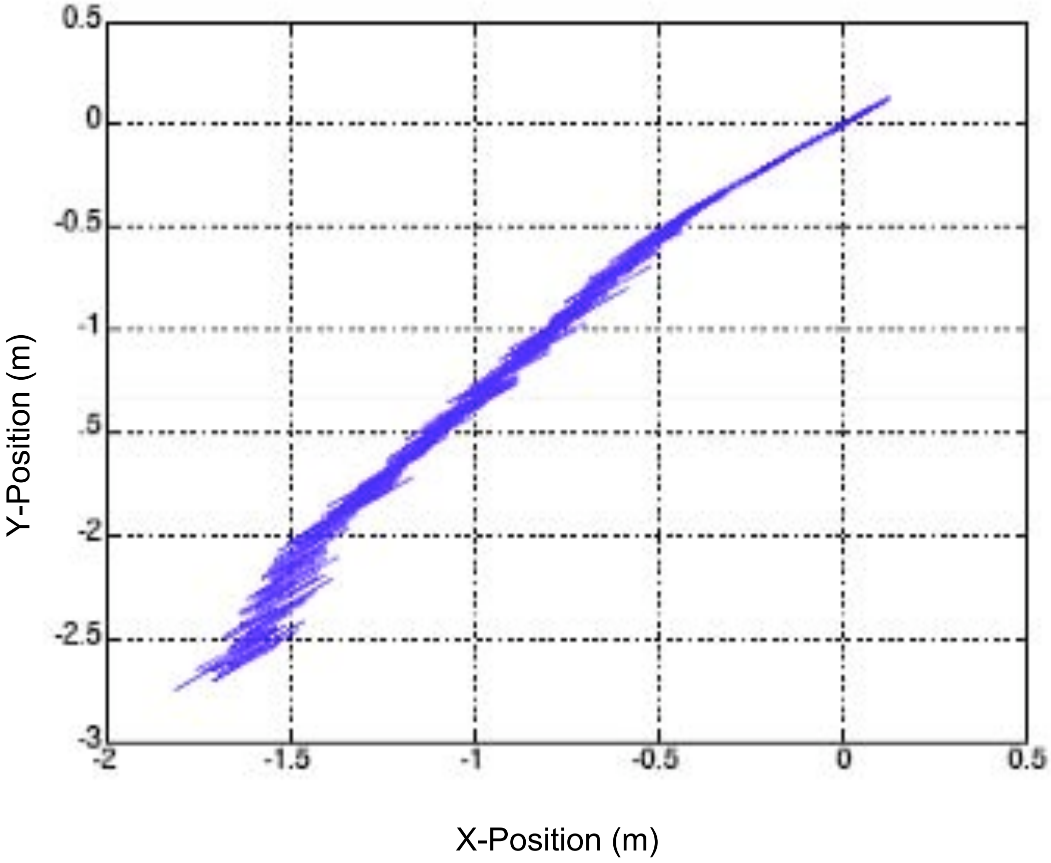


Figure 3.1 XY-position measurement of SAFER unit, with both dynamics disturbances (process noise) and sensor chatter (measurement noise)

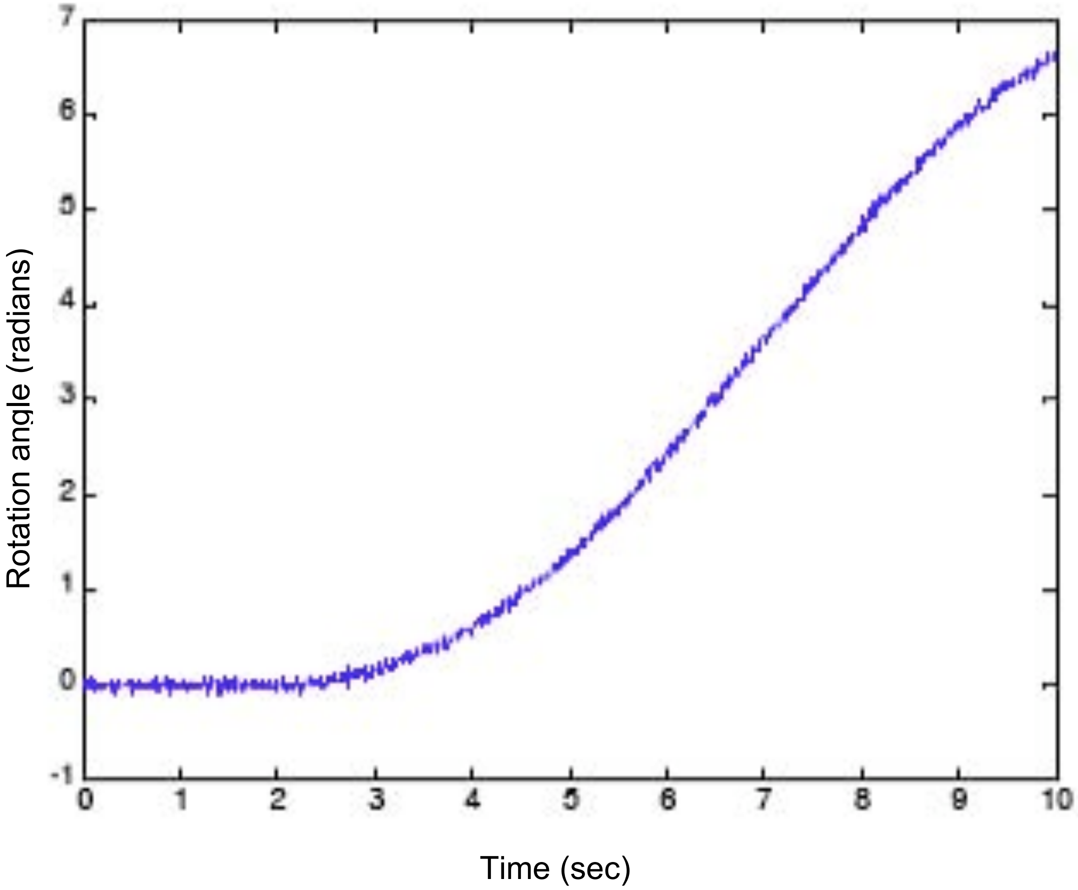


Figure 3.2. Rotation angle of SAFER unit, over time with both dynamics disturbances (process noise) and sensor chatter (measurement noise)

**Problem 6.**

To determine the matrices needed to create the Kalman filter, we want our dynamics to be represented in the form $\dot{x} = Ax + Bu$, which is equivalent to:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \left(\dfrac{\cos\theta}{mass}\right) & \left(\dfrac{-\sin\theta}{mass}\right) & 0 \\ \left(\dfrac{\sin\theta}{mass}\right) & \left(\dfrac{\cos\theta}{mass}\right) & 0 \\ 0 & 0 & \left(\dfrac{1}{Inertia}\right) \end{bmatrix} \begin{bmatrix} F_{\dot{x}} \\ F_{\dot{y}} \\ \tau \end{bmatrix}$$

We will also need to determine the measurement matrix, H; the measurement covariance matrix, R; the continuous process noise covariance matrix, Q; the initial state covariance matrix estimate, P; and the initial state vector estimate, X.

The measurement update and time update of the Kalman filter will be carried out according to the following matrix equations (Ferguson, 2006, SBE Dynamics Lecture Notes):

*Measurement Update*

$$\begin{aligned} S_k &= H_k P_k^- H_k^T + R_k \\ K_k &= P_k^- H_k^T S_k^{-1} \\ \hat{X}_k^+ &= \hat{X}_k^- + K_k \left( z_k - H_k \hat{X}_k^- \right) \\ P_k^+ &= \left( \mathbf{1} - K_k H_k \right) P_k^- \end{aligned}$$

*Time Update*

$$\begin{aligned} \hat{X}_{k+1}^- &= \Phi_k \hat{X}_k^+ \\ P_{k+1}^- &= \Phi_k P_k^+ \Phi_k^T + Q_k \end{aligned}$$

We must put these into Matlab form, in a discrete time-step loop. To apply the Kalman filter to our simulated noisy data, we can use the following Matlab code:

```
% Clean up the environment
close all
clear all

% Constants
rad2deg = 180/pi;
deg2rad = pi/180;

% Set the time vector
start_time = 0;
stop_time = 10;
time_step = 0.01;
time_vec = start_time:time_step:stop_time;

% Force and torque values
on_force_val = 10;
on_torque_val = 3;

% Mass properties
mass = 125;  % kg
inertia = 10;

% Buid the force and torque vectors as functions of time
x_forcev = zeros(size(time_vec));
y_forcev = zeros(size(time_vec));
torquev = zeros(size(time_vec));
for i = 1:length(time_vec)
    if (time_vec(i) >= 7)
        x_forcev(i) = x_forcev(i) + on_force_val;
    end
if (time_vec(i) <= 3)
        x_forcev(i) = x_forcev(i) - on_force_val;
    end
    if ((time_vec(i) >=2)&(time_vec(i) <= 8))
        torquev(i) = torquev(i) + on_torque_val;
    end
    if (time_vec(i) >= 6)
        torquev(i) = torquev(i) - on_torque_val;
    end
    if (time_vec(i) <= 4)
        y_forcev(i) = y_forcev(i) - on_force_val;
    end
    if (time_vec(i) >= 8)
        y_forcev(i) = y_forcev(i) + on_force_val;
    end

end

% Format the forces and torques for use with the source block
torque = [time_vec' torquev'];
x_force = [time_vec' x_forcev'];
y_force = [time_vec' y_forcev'];

% Process noise variances
% Note that these are squares of standard deviations, so they look pretty
% small
```

```
pos_proc_var = 0.01;
ang_proc_var = 0.001;

% Measurement noise variances
% Note that these are squares of standard deviations, so they look pretty
% small
pos_meas_var = 0.005;
ang_meas_var = 0.001;

% Run the dynamics from the Simulink model
% Check the simulation configuration dialog of the model to ensure that the
% following things are set:
%
% Start time:  start_time
% Stop time:  stop_time
% Type:  Fixed-step
% Solver:  ode3(Bogacki-Sharnpine)
% Periodic sample time constraint:  Unconstrained
% Fixed-step size (fundamental sample time):  time_step
% Tasking mode for periodic sample times:  Auto
% Higher priority value indicates higher task priority:  Unchecked
% Automatically handle data transfers between tasks:  Unchecked
%
% After running the simulation, the following variables are available:
%
% truth
% meas
%
% Both are structures that contain the data we're interested in
% The structure is automatically created by Simulink and is needlessly
% complicated … They are extracted below for simplicity
sim(‘your_simulink_model’)

% Extract the truth and the measurements from the Simulink data structures
meas_vecs = meas.signals.values;
truth_vecs = truth.signals.values;

% We now proceed with the setup of the Kalman Filter
% We assume that the state we want to estimate takes the following form:
% X = [x y theta x_dot y_dot theta_dot]'
%
% The measurement vector has the following form:
% y = [x_hat y_hat theta_hat]

% Form the measurement covariance matrix
R = diag([pos_meas_var; pos_meas_var; ang_meas_var]);

% Form the continuous process noise covariance matrix
% Note that process noise is only added to the velocity states because
% we're really adding this to the state equation:
%
% X_dot = Phi*X + B*u + Q
Q = diag([0, 0, 0, pos_proc_var, pos_proc_var, ang_proc_var]);

% Put the input vectors into a matrix for easy use
input_vectors = [x_forcev', y_forcev', torquev'];
```

```
% Form the measurement matrix such that
% y = H*X
H = [1 0 0 0 0 0;
     0 1 0 0 0 0;
     0 0 1 0 0 0];

% Form the continuous dynamics matrix
A = [0 0 0 1 0 0;
     0 0 0 0 1 0;
     0 0 0 0 0 1;
     0 0 0 0 0 0;
     0 0 0 0 0 0;
     0 0 0 0 0 0];

% The continuous B matrix for X_dot = AX + Bu depends on the state (angle)
% So, it needs to be computed at every time step

% Define the initial error variances on the estimate we start with
init_pos_err_var = 0.025;
init_vel_err_var = 1;
init_ang_err_var = 0.01;
init_ang_rate_err_var = 0.02;

% Initialize the state covariance estimate
P_init = diag([init_pos_err_var, init_pos_err_var, init_ang_err_var, init_vel_err_var, init_vel_err_var,
init_ang_rate_err_var]);
P = P_init

% Initialize the state estimate
X_init = truth_vecs(1,:)' + sqrtm(P)*randn(6,1);
X = X_init;

% Setup storage matrices for our estimates
Pdiag_store = [];
X_store = [];

% Run the filter
for i = 1:length(time_vec)
    % Store the P diagonal
    Pdiag_store = [Pdiag_store, diag(P)];

    % Do meas update
    K = P*H'*inv(H*P*H' + R);
    X = X + K*(meas_vecs(i,:)' - H*X);
    P = (eye(size(P)) - K*H)*P;

    % Store the estimate
    X_store = [X_store, X];

    % Do time update
    %First compute the continuous B matrix at this time step, recalling that X(3) is the current theta angle
    B = [0, 0, 0;
           0, 0, 0;
```

```
            0, 0, 0;
            (1/mass)*cos(X(3)), (-1/mass)*sin(X(3)), 0;
            (1/mass)*sin(X(3)), (1/mass)*cos(X(3)), 0;
            0, 0, (1/inertia)];

    [Phi, Bk, Qk] = get_discrete_dyn(A, Q, B, time_step);
    X = Phi*X + Bk*input_vectors(i,:)';
    P = Phi*P*Phi' + Qk;
end

% Compute the errors
X_diff = truth_vecs' - X_store;
```

```
%%%%%%%%%%%%%%%%%%%

% Here is the get_discrete_dyn function
% It needs to be stored as a separate .m file, with the filename get_discrete_dyn.m

function [Phi_full, Bk, Qk] = get_discrete_dyn(A, Q, B, time_step)

% Build Spectral Densities
full_state_size = length(A);
S = [-A Q; zeros(size(A)) A'];
C = expm(S*time_step);

% Discretize A and B into Phi and Bk
csys = ss(A, B, [], []);
dsys = c2d(csys, time_step, 'zoh');
Phi_full = dsys.A;
Bk = dsys.B;

% Form Qk
Qk = Phi_full*C(1:full_state_size,full_state_size+1:2*full_state_size);
```

## Problem 7.

To create the plots of estimation error and covariance bounds, we can use the following Matlab code.

```
% Make some plots
% The truth position
figure
plot(truth_vecs(:,1), truth_vecs(:,2))
grid
title('Subject motion in X-Y plane')
xlabel('X position (m)');
ylabel('Y position (m)');

% The truth angle
figure
plot(time_vec, truth_vecs(:,3).*rad2deg)
grid
title('Subject rotation angle')
xlabel('Time (s)');
ylabel('Rotation angle (degrees)');

% Position estimation errors and covariance bounds
figure
subplot(3,1,1)
plot(time_vec, X_diff(1,:), time_vec, sqrt(Pdiag_store(1,:)), 'k--', time_vec, -sqrt(Pdiag_store(1,:)), 'k--')
title('Position Estimation Errors and Covariance Bounds');
ylabel('X Position (m)');
grid
subplot(3,1,2)
plot(time_vec, X_diff(2,:), time_vec, sqrt(Pdiag_store(2,:)), 'k--', time_vec, -sqrt(Pdiag_store(2,:)), 'k--')
ylabel('Y Position (m)');
xlabel('Time (s)')
grid
subplot(3,1,3)
plot(time_vec, X_diff(3,:).*rad2deg, time_vec, sqrt(Pdiag_store(3,:)).*rad2deg, 'k--', time_vec, -
sqrt(Pdiag_store(3,:)).*rad2deg, 'k--')
ylabel('Rotation Angle (degrees)');
xlabel('Time (s)')
grid

% Rate estimation errors and covariance bounds
figure
subplot(3,1,1)
plot(time_vec, X_diff(4,:), time_vec, sqrt(Pdiag_store(4,:)), 'k--', time_vec, -sqrt(Pdiag_store(4,:)), 'k--')
title('Velocity Estimation Errors and Covariance Bounds');
ylabel('X Velocity (m/s)');
grid
subplot(3,1,2)
plot(time_vec, X_diff(5,:), time_vec, sqrt(Pdiag_store(5,:)), 'k--', time_vec, -sqrt(Pdiag_store(5,:)), 'k--')
ylabel('Y Velocity (m/s)');
xlabel('Time (s)')
grid
subplot(3,1,3)
plot(time_vec, X_diff(6,:).*rad2deg, time_vec, sqrt(Pdiag_store(6,:)).*rad2deg, 'k--', time_vec, -
sqrt(Pdiag_store(6,:)).*rad2deg, 'k--')
```

```
ylabel('Rotation Rate (degrees/s)');
xlabel('Time (s)')
grid
```

The error in our position estimation and its covariance bounds are shown in Figure 7.1.

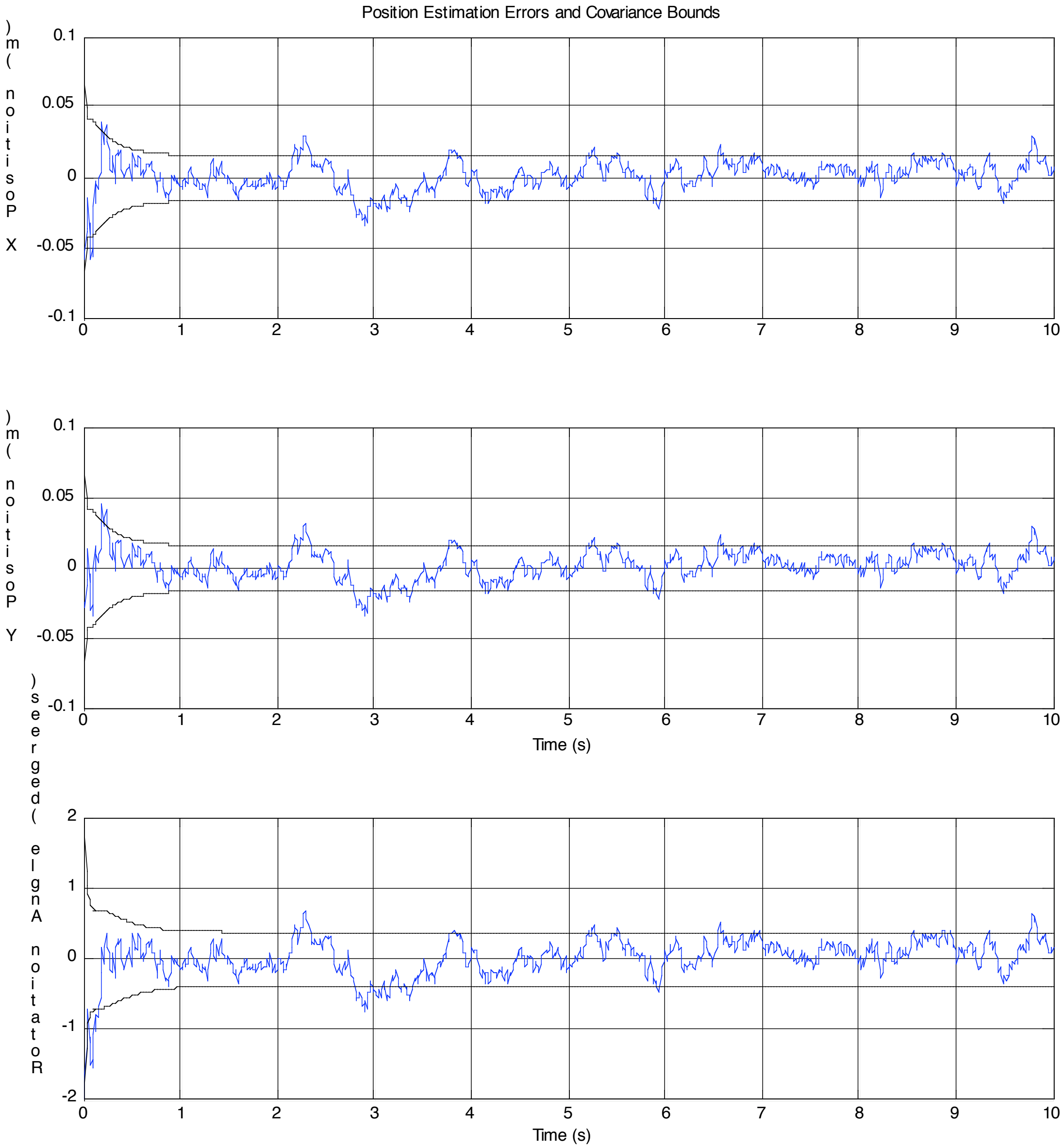


Figure 7.1 Position estimation errors and covariance bounds

The error in our velocity estimation and its covariance bounds are shown in Figure 7-2.

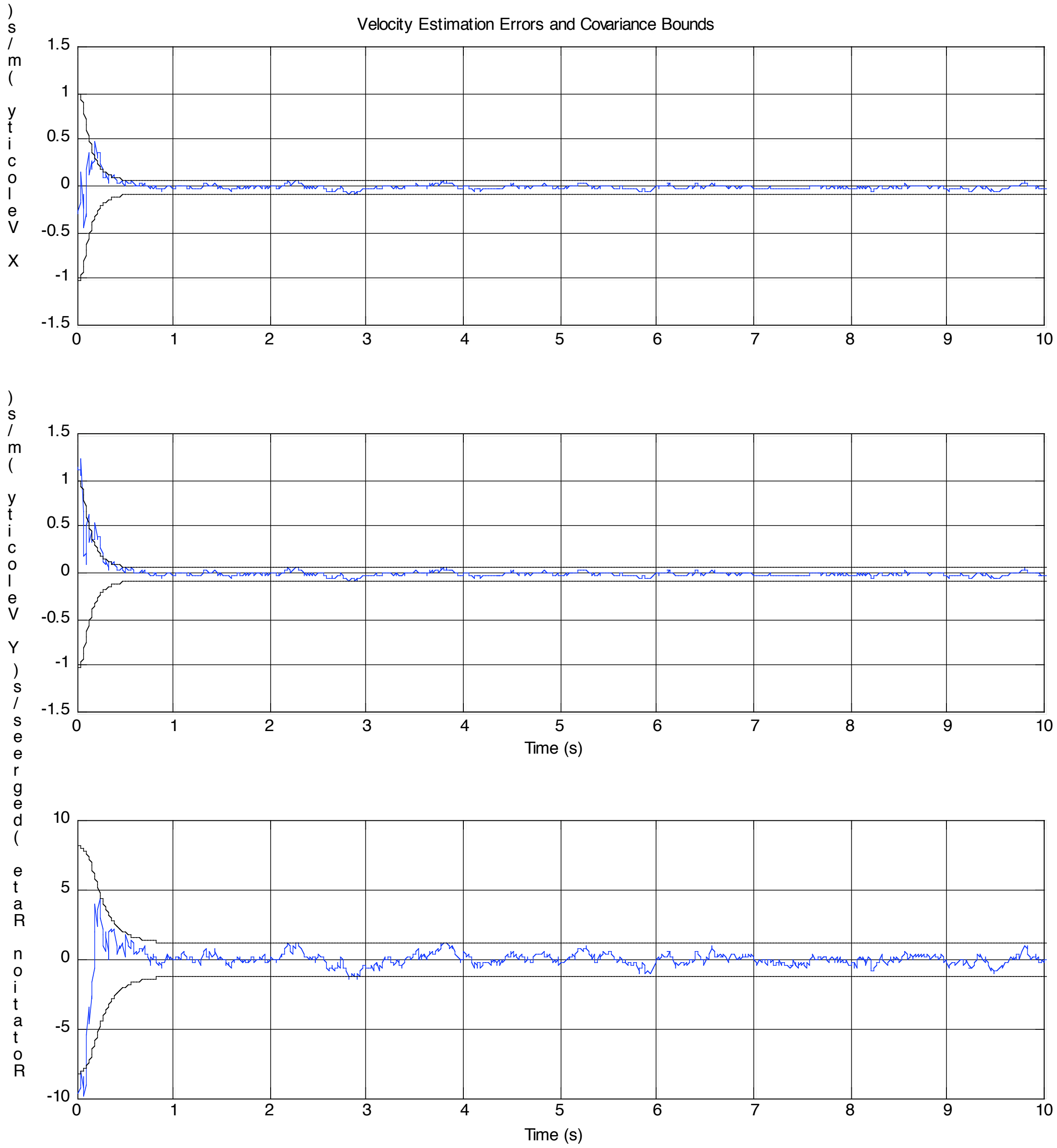


Figure 7.2 Velocity estimation errors and covariance bounds

**Problem 8.**

If you suspect that the friction has a Gaussian white noise effect, increase the velocity elements of the Q matrix to be at least as big as you think the friction might be. If you don't have any idea what the friction might look like, you would probably just assume it was Gaussian so that you could do the above.

If you knew exactly what the friction was, you could model it and incorporate it into the dynamics as part of the A matrix.

However, if you suspected a particular form of the friction, but did NOT know the parameters, you could “augment” the state matrix, which means incorporating the B matrix into your state vector. For instance, maybe you know that it took the form of:

F_friction = -B * velocity

But you did not know what B was. The Kalman filter has the ability to estimate what B is based on the data. The way you would do this would be to include B into your state
vector as a constant. The dynamics of B are nothing (because it is constant). You would need to incorporate this state into the measurement equation (so into H) also.
This way, not only would your filter track the position and velocity of the astronaut on the SAFER unit, you would also be able to do some system ID too!

# An Introduction to the Kalman Filter

**Greg Welch and Gary Bishop**

**Updated: Monday, April 5, 2004**

## Abstract

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation.

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The purpose of this paper is to provide a practical introduction to the discrete Kalman filter. This introduction includes a description and some discussion of the basic discrete Kalman filter, a derivation, description and some discussion of the extended Kalman filter, and a relatively simple (tangible) example with real numbers & results.

# 1 The Discrete Kalman Filter

In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [Kalman60]. Since that time, due in large part to advances in digital computing, the *Kalman filter* has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. A very "friendly" introduction to the general idea of the Kalman filter can be found in Chapter 1 of [Maybeck79], while a more complete introductory discussion can be found in [Sorenson70], which also contains some interesting historical narrative. More extensive references include [Gelb74; Grewal93; Maybeck79; Lewis86; Brown92; Jacobs93].

### *The Process to be Estimated*

The Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}, \tag{1.1}$$

with a measurement $z \in \Re^m$ that is

$$z_k = Hx_k + v_k. \tag{1.2}$$

The random variables $w_k$ and $v_k$ represent the process and measurement noise (respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0, Q), \tag{1.3}$$

$$p(v) \sim N(0, R). \tag{1.4}$$

In practice, the *process noise covariance* $Q$ and *measurement noise covariance* $R$ matrices might change with each time step or measurement, however here we assume they are constant.

The $n \times n$ matrix $A$ in the difference equation (1.1) relates the state at the previous time step $k-1$ to the state at the current step $k$, in the absence of either a driving function or process noise. Note that in practice $A$ might change with each time step, but here we assume it is constant. The $n \times l$ matrix $B$ relates the optional control input $u \in \Re^l$ to the state $x$. The $m \times n$ matrix $H$ in the measurement equation (1.2) relates the state to the measurement $z_k$. In practice $H$ might change with each time step or measurement, but here we assume it is constant.

### *The Computational Origins of the Filter*

We define $\hat{x}^-_k \in \Re^n$ (note the "super minus") to be our *a priori* state estimate at step $k$ given knowledge of the process prior to step $k$, and $\hat{x}_k \in \Re^n$ to be our *a posteriori* state estimate at step $k$ given measurement $z_k$. We can then define *a priori* and *a posteriori* estimate errors as

$$e^-_k \equiv x_k - \hat{x}^-_k, \text{ and}$$

$$e_k \equiv x_k - \hat{x}_k.$$

The *a priori* estimate error covariance is then

$$P_k^- = E[e_k^- e_k^{-T}], \tag{1.5}$$

and the *a posteriori* estimate error covariance is

$$P_k = E[e_k e_k^T]. \tag{1.6}$$

In deriving the equations for the Kalman filter, we begin with the goal of finding an equation that computes an *a posteriori* state estimate $\hat{x}_k$ as a linear combination of an *a priori* estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement $z_k$ and a measurement prediction $H\hat{x}_k^-$ as shown below in (1.7). Some justification for (1.7) is given in "The Probabilistic Origins of the Filter" found below.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{1.7}$$

The difference $(z_k - H\hat{x}_k^-)$ in (1.7) is called the measurement *innovation*, or the *residual*. The residual reflects the discrepancy between the predicted measurement $H\hat{x}_k^-$ and the actual measurement $z_k$. A residual of zero means that the two are in complete agreement.

The $n \times m$ matrix *K* in (1.7) is chosen to be the *gain* or *blending factor* that minimizes the *a posteriori* error covariance (1.6). This minimization can be accomplished by first substituting (1.7) into the above definition for $e_k$, substituting that into (1.6), performing the indicated expectations, taking the derivative of the trace of the result with respect to *K*, setting that result equal to zero, and then solving for *K*. For more details see [Maybeck79; Brown92; Jacobs93]. One form of the resulting *K* that minimizes (1.6) is given by[1]

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \frac{P_k^- H^T}{H P_k^- H^T + R} \end{aligned} \quad . \tag{1.8}$$

Looking at (1.8) we see that as the measurement error covariance $R$ approaches zero, the gain *K* weights the residual more heavily. Specifically,

$$\lim_{R_k \to 0} K_k = H^{-1}.$$

On the other hand, as the *a priori* estimate error covariance $P_k^-$ approaches zero, the gain *K* weights the residual less heavily. Specifically,

$$\lim_{P_k^- \to 0} K_k = 0.$$

[1] All of the Kalman filter equations can be algebraically manipulated into to several forms. Equation (1.8) represents the Kalman gain in one popular form.

Another way of thinking about the weighting by $K$ is that as the measurement error covariance $R$ approaches zero, the actual measurement $z_k$ is "trusted" more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. On the other hand, as the *a priori* estimate error covariance $P_k^-$ approaches zero the actual measurement $z_k$ is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

### *The Probabilistic Origins of the Filter*

The justification for (1.7) is rooted in the probability of the *a priori* estimate $\hat{x}_k^-$ conditioned on all prior measurements $z_k$ (Bayes' rule). For now let it suffice to point out that the Kalman filter maintains the first two moments of the state distribution,

$$E[x_k] = \hat{x}_k$$

$$E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] = P_k.$$

The *a posteriori* state estimate (1.7) reflects the mean (the first moment) of the state distribution— it is normally distributed if the conditions of (1.3) and (1.4) are met. The *a posteriori* estimate error covariance (1.6) reflects the variance of the state distribution (the second non-central moment). In other words,

$$\begin{aligned} p(x_k|z_k) &\sim N(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]) \\ &= N(\hat{x}_k, P_k). \end{aligned}$$

For more details on the probabilistic origins of the Kalman filter, see [Maybeck79; Brown92; Jacobs93].

### *The Discrete Kalman Filter Algorithm*

We will begin this section with a broad overview, covering the "high-level" operation of one form of the discrete Kalman filter (see the previous footnote). After presenting this high-level view, we will narrow the focus to the specific equations and their use in this version of the filter.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: *time update* equations and *measurement update* equations. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step. The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown below in Figure 1-1.

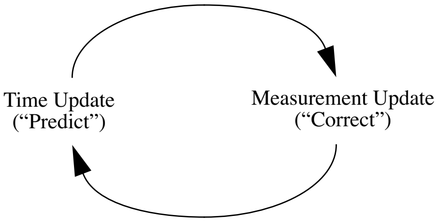


**Figure 1-1.** The ongoing discrete Kalman filter cycle. The *time update* projects the current state estimate ahead in time. The *measurement update* adjusts the projected estimate by an actual measurement at that time.

The specific equations for the time and measurement updates are presented below in Table 1-1 and Table 1-2.

**Table 1-1:** Discrete Kalman filter time update equations.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{1.9}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{1.10}$$

Again notice how the time update equations in Table 1-1 project the state and covariance estimates forward from time step $k-1$ to step $k$. $A$ and $B$ are from (1.1), while $Q$ is from (1.3). Initial conditions for the filter are discussed in the earlier references.

**Table 1-2:** Discrete Kalman filter measurement update equations.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{1.11}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{1.12}$$

$$P_k = (I - K_k H)P_k^- \tag{1.13}$$

The first task during the measurement update is to compute the Kalman gain, $K_k$. Notice that the equation given here as (1.11) is the same as (1.8). The next step is to actually measure the process to obtain $z_k$, and then to generate an *a posteriori* state estimate by incorporating the measurement as in (1.12). Again (1.12) is simply (1.7) repeated here for completeness. The final step is to obtain an *a posteriori* error covariance estimate via (1.13).

After each time and measurement update pair, the process is repeated with the previous *a posteriori* estimates used to project or predict the new *a priori* estimates. This recursive nature is one of the very appealing features of the Kalman filter—it makes practical implementations much more feasible than (for example) an implementation of a Wiener filter [Brown92] which is designed to operate on *all* of the data *directly* for each estimate. The Kalman filter instead recursively conditions the current estimate on all of the past measurements. Figure 1-2 below offers a complete picture of the operation of the filter, combining the high-level diagram of Figure 1-1 with the equations from Table 1-1 and Table 1-2.

### *Filter Parameters and Tuning*

In the actual implementation of the filter, the measurement noise covariance $R$ is usually measured prior to operation of the filter. Measuring the measurement error covariance $R$ is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we should generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise.

The determination of the process noise covariance $Q$ is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one "injects" enough uncertainty into the process via the selection of $Q$. Certainly in this case one would hope that the process measurements are reliable.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by *tuning* the filter parameters $Q$ and $R$. The tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter in a process generally referred to as *system identification*.

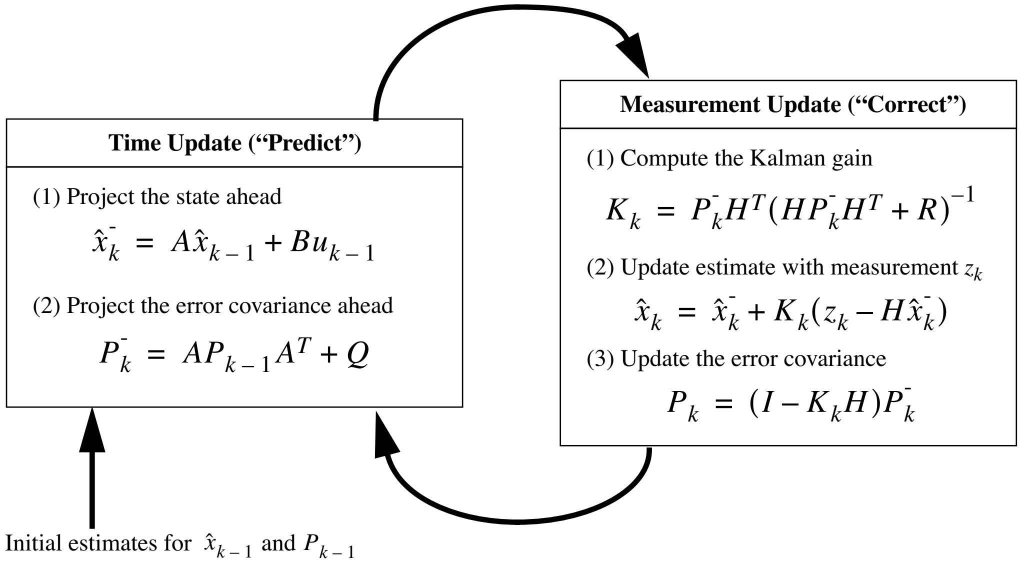

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement $z_k$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and $P_{k-1}$

**Figure 1-2.** A complete picture of the operation of the Kalman filter, combining the high-level diagram of Figure 1-1 with the equations from Table 1-1 and Table 1-2.

In closing we note that under conditions where $Q$ and $R$ .are in fact constant, both the estimation error covariance $P_k$ and the Kalman gain $K_k$ will stabilize quickly and then remain constant (see the filter update equations in Figure 1-2). If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of $P_k$ as described in [Grewal93].

It is frequently the case however that the measurement error (in particular) does not remain constant. For example, when sighting beacons in our optoelectronic tracker ceiling panels, the noise in measurements of nearby beacons will be smaller than that in far-away beacons. Also, the process noise $Q$ is sometimes changed dynamically during filter operation—becoming $Q_k$ —in order to adjust to different dynamics. For example, in the case of tracking the head of a user of a 3D virtual environment we might reduce the magnitude of $Q_k$ if the user seems to be moving slowly, and increase the magnitude if the dynamics start changing rapidly. In such cases $Q_k$ might be chosen to account for both uncertainty about the user's intentions and uncertainty in the model.

# 2 The Extended Kalman Filter (EKF)

### *The Process to be Estimated*

As described above in section 1, the Kalman filter addresses the general problem of trying to estimate the state $x \in \Re^n$ of a discrete-time controlled process that is governed by a *linear* stochastic difference equation. But what happens if the process to be estimated and (or) the measurement relationship to the process is non-linear? Some of the most interesting and successful applications of Kalman filtering have been such situations. A Kalman filter that linearizes about the current mean and covariance is referred to as an *extended Kalman filter* or EKF.

In something akin to a Taylor series, we can linearize the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face of non-linear relationships. To do so, we must begin by modifying some of the material presented in section 1. Let us assume that our process again has a state vector $x \in \Re^n$, but that the process is now governed by the *non-linear* stochastic difference equation

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \tag{2.1}$$

with a measurement $z \in \Re^m$ that is

$$z_k = h(x_k, v_k), \tag{2.2}$$

where the random variables $w_k$ and $v_k$ again represent the process and measurement noise as in (1.3) and (1.4). In this case the *non-linear* function $f$ in the difference equation (2.1) relates the state at the previous time step $k-1$ to the state at the current time step $k$. It includes as parameters any driving function $u_{k-1}$ and the zero-mean process noise $w_k$. The *non-linear* function $h$ in the measurement equation (2.2) relates the state $x_k$ to the measurement $z_k$.

In practice of course one does not know the individual values of the noise $w_k$ and $v_k$ at each time step. However, one can approximate the state and measurement vector without them as

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{2.3}$$

and

$$\tilde{z}_k = h(\tilde{x}_k, 0), \tag{2.4}$$

where $\hat{x}_k$ is some *a posteriori* estimate of the state (from a previous time step *k*).

It is important to note that a fundamental flaw of the EKF is that the distributions (or densities in the continuous case) of the various random variables are no longer normal after undergoing their respective nonlinear transformations. The EKF is simply an *ad hoc* state estimator that only approximates the optimality of Bayes' rule by linearization. Some interesting work has been done by Julier et al. in developing a variation to the EKF, using methods that preserve the normal distributions throughout the non-linear transformations [Julier96].

### *The Computational Origins of the Filter*

To estimate a process with non-linear difference and measurement relationships, we begin by writing new governing equations that linearize an estimate about (2.3) and (2.4),

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}, \tag{2.5}$$

$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k. \tag{2.6}$$

where

- $x_k$ and $z_k$ are the actual state and measurement vectors,
- $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors from (2.3) and (2.4),
- $\hat{x}_k$ is an *a posteriori* estimate of the state at step $k$,
- the random variables $w_k$ and $v_k$ represent the process and measurement noise as in (1.3) and (1.4).
- $A$ is the Jacobian matrix of partial derivatives of $f$ with respect to $x$, that is

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0),$$

- $W$ is the Jacobian matrix of partial derivatives of $f$ with respect to $w$,

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0),$$

- $H$ is the Jacobian matrix of partial derivatives of $h$ with respect to $x$,

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0),$$

- $V$ is the Jacobian matrix of partial derivatives of $h$ with respect to $v$,

$$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\tilde{x}_k, 0).$$

Note that for simplicity in the notation we do not use the time step subscript $k$ with the Jacobians $A$, $W$, $H$, and $V$, even though they are in fact different at each time step.

Now we define a new notation for the prediction error,

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k\,, \tag{2.7}$$

and the measurement residual,

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k\,. \tag{2.8}$$

Remember that in practice one does not have access to $x_k$ in (2.7), it is the *actual* state vector, i.e. the quantity one is trying to estimate. On the other hand, one *does* have access to $z_k$ in (2.8), it is the actual measurement that one is using to estimate $x_k$. Using (2.7) and (2.8) we can write governing equations for an *error process* as

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k\,, \tag{2.9}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k\,, \tag{2.10}$$

where $\varepsilon_k$ and $\eta_k$ represent new independent random variables having zero mean and covariance matrices $WQW^T$ and $VRV^T$, with $Q$ and $R$ as in (1.3) and (1.4) respectively.

Notice that the equations (2.9) and (2.10) are linear, and that they closely resemble the difference and measurement equations (1.1) and (1.2) from the discrete Kalman filter. This motivates us to use the actual measurement residual $\tilde{e}_{z_k}$ in (2.8) and a second (hypothetical) Kalman filter to estimate the prediction error $\tilde{e}_{x_k}$ given by (2.9). This estimate, call it $\hat{e}_k$, could then be used along with (2.7) to obtain the *a posteriori* state estimates for the original non-linear process as

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k\,. \tag{2.11}$$

The random variables of (2.9) and (2.10) have approximately the following probability distributions (see the previous footnote):

$$p(\tilde{e}_{x_k}) \sim N(0, E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T])$$

$$p(\varepsilon_k) \sim N(0, WQ_kW^T)$$

$$p(\eta_k) \sim N(0, VR_kV^T)$$

Given these approximations and letting the predicted value of $\hat{e}_k$ be zero, the Kalman filter equation used to estimate $\hat{e}_k$ is

$$\hat{e}_k = K_k\tilde{e}_{z_k}\,. \tag{2.12}$$

By substituting (2.12) back into (2.11) and making use of (2.8) we see that we do not actually need the second (hypothetical) Kalman filter:

$$\begin{aligned}\hat{x}_k &= \tilde{x}_k + K_k\tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k(z_k - \tilde{z}_k)\end{aligned} \tag{2.13}$$

Equation (2.13) can now be used for the measurement update in the extended Kalman filter, with $\tilde{x}_k$ and $\tilde{z}_k$ coming from (2.3) and (2.4), and the Kalman gain $K_k$ coming from (1.11) with the appropriate substitution for the measurement error covariance.

The complete set of EKF equations is shown below in Table 2-1 and Table 2-2. Note that we have substituted $\hat{x}_k^-$ for $\tilde{x}_k$ to remain consistent with the earlier "super minus" a priori notation, and that we now attach the subscript $k$ to the Jacobians $A$ , $W$ , $H$ , and $V$ , to reinforce the notion that they are different at (and therefore must be recomputed at) each time step.

**Table 2-1:** EKF time update equations.

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0) \qquad (2.14)$$

$$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \qquad (2.15)$$

As with the basic discrete Kalman filter, the time update equations in Table 2-1 project the state and covariance estimates from the previous time step $k-1$ to the current time step $k$ . Again $f$ in (2.14) comes from (2.3), $A_k$ and $W_k$ are the process Jacobians at step $k$, and $Q_k$ is the process noise covariance (1.3) at step $k$.

**Table 2-2:** EKF measurement update equations.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \qquad (2.16)$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \qquad (2.17)$$

$$P_k = (I - K_k H_k) P_k^- \qquad (2.18)$$

As with the basic discrete Kalman filter, the measurement update equations in Table 2-2 correct the state and covariance estimates with the measurement $z_k$ . Again $h$ in (2.17) comes from (2.4), $H_k$ and $V$ are the measurement Jacobians at step $k$, and $R_k$ is the measurement noise covariance (1.4) at step $k$. (Note we now subscript $R$ allowing it to change with each measurement.)

The basic operation of the EKF is the same as the linear discrete Kalman filter as shown in Figure 1-1. Figure 2-1 below offers a complete picture of the operation of the EKF, combining the high-level diagram of Figure 1-1 with the equations from Table 2-1 and Table 2-2.

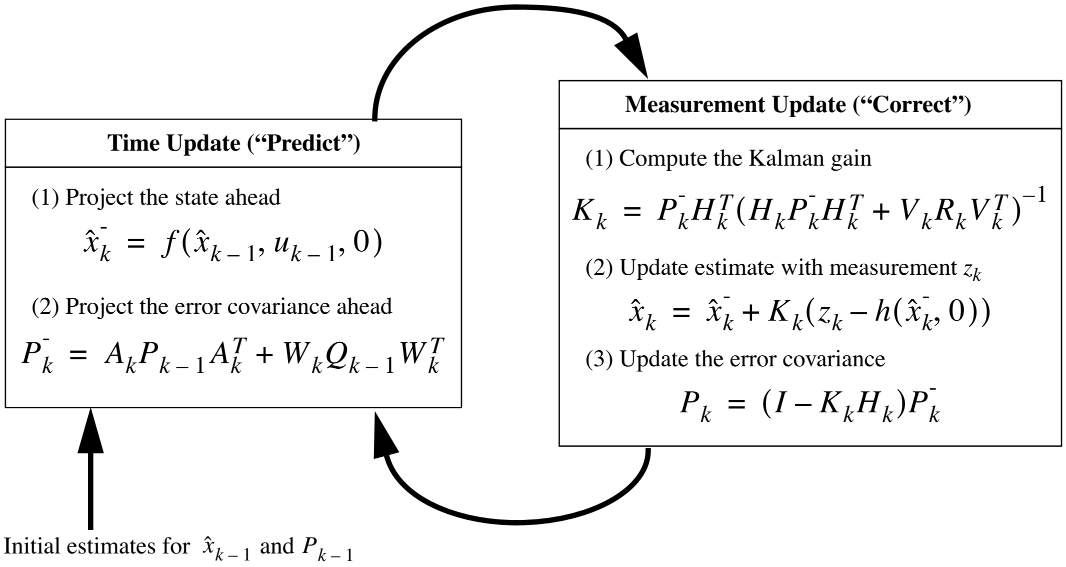

Time Update (“Predict”)
(1) Project the state ahead
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0)$
(2) Project the error covariance ahead
$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$
Measurement Update (“Correct”)
(1) Compute the Kalman gain
$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$
(2) Update estimate with measurement $z_k$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-, 0))$
(3) Update the error covariance
$P_k = (I - K_k H_k) P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and $P_{k-1}$

**Figure 2-1.** A complete picture of the operation of the *extended* Kalman filter, combining the high-level diagram of Figure 1-1 with the equations from Table 2-1 and Table 2-2.

An important feature of the EKF is that the Jacobian $H_k$ in the equation for the Kalman gain $K_k$ serves to correctly propagate or “magnify” only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between the measurement $z_k$ and the state via $h$, the Jacobian $H_k$ affects the Kalman gain so that it only magnifies the portion of the residual $z_k - h(\hat{x}_k^-, 0)$ that does affect the state. Of course if over *all* measurements there is *not* a one-to-one mapping between the measurement $z_k$ and the state via $h$, then as you might expect the filter will quickly diverge. In this case the process is *unobservable*.

## 3 A Kalman Filter in Action: Estimating a Random Constant

In the previous two sections we presented the basic form for the discrete Kalman filter, and the extended Kalman filter. To help in developing a better feel for the operation and capability of the filter, we present a very simple example here.

### *The Process Model*

In this simple example let us attempt to estimate a scalar random constant, a voltage for example. Let’s assume that we have the ability to take measurements of the constant, but that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise (e.g. our analog to digital converter is not very accurate). In this example, our process is governed by the linear difference equation

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_k \\ &= x_{k-1} + w_k \end{aligned},$$

with a measurement $z \in \Re^1$ that is

$$\begin{aligned} z_k &= Hx_k + v_k \\ &= x_k + v_k \end{aligned}.$$

The state does not change from step to step so $A = 1$. There is no control input so $u = 0$. Our noisy measurement is of the state directly so $H = 1$. (Notice that we dropped the subscript *k* in several places because the respective parameters remain constant in our simple model.)

### *The Filter Equations and Parameters*

Our time update equations are

$$\hat{x}_k^- = \hat{x}_{k-1},$$

$$P_k^- = P_{k-1} + Q,$$

and our measurement update equations are

$$\begin{aligned} K_k &= P_k^-(P_k^- + R)^{-1} \\ &= \frac{P_k^-}{P_k^- + R} \end{aligned}, \tag{3.1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-),$$

$$P_k = (1 - K_k)P_k^-.$$

Presuming a very small process variance, we let $Q = 1e-5$. (We could certainly let $Q = 0$ but assuming a small but non-zero value gives us more flexibility in “tuning” the filter as we will demonstrate below.) Let’s assume that from experience we know that the true value of the random constant has a standard normal probability distribution, so we will “seed” our filter with the guess that the constant is 0. In other words, before starting we let $\hat{x}_{k-1} = 0$.

Similarly we need to choose an initial value for $P_{k-1}$, call it $P_0$. If we were absolutely certain that our initial state estimate $\hat{x}_0 = 0$ was correct, we would let $P_0 = 0$. However given the uncertainty in our initial estimate $\hat{x}_0$, choosing $P_0 = 0$ would cause the filter to initially and always believe $\hat{x}_k = 0$. As it turns out, the alternative choice is not critical. We could choose almost any $P_0 \neq 0$ and the filter would *eventually* converge. We’ll start our filter with $P_0 = 1$.

### *The Simulations*

To begin with, we randomly chose a scalar constant $z = -0.37727$ (there is no "hat" on the $z$ because it represents the "truth"). We then simulated 50 distinct measurements $z_k$ that had error normally distributed around zero with a standard deviation of 0.1 (remember we presumed that the measurements are corrupted by a 0.1 volt RMS *white* measurement noise). We could have generated the individual measurements within the filter loop, but pre-generating the set of 50 measurements allowed me to run several simulations with the same exact measurements (i.e. same measurement noise) so that comparisons between simulations with different parameters would be more meaningful.

In the first simulation we fixed the measurement variance at $R = (0.1)^2 = 0.01$. Because this is the "true" measurement error variance, we would expect the "best" performance in terms of balancing responsiveness and estimate variance. This will become more evident in the second and third simulation. Figure 3-1 depicts the results of this first simulation. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

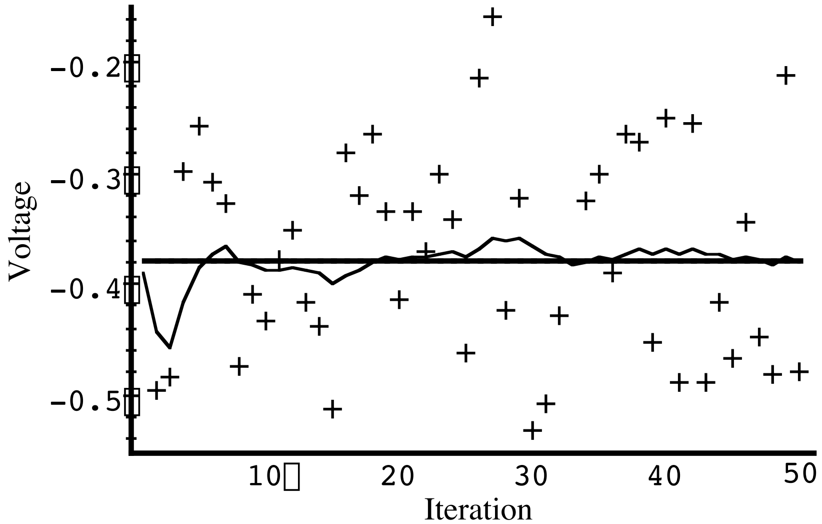


**Figure 3-1.** The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

When considering the choice for $P_0$ above, we mentioned that the choice was not critical as long as $P_0 \neq 0$ because the filter would eventually converge. Below in Figure 3-2 we have plotted the value of $P_k$ versus the iteration. By the 50th iteration, it has settled from the initial (rough) choice of 1 to approximately 0.0002 (Volts$^2$).

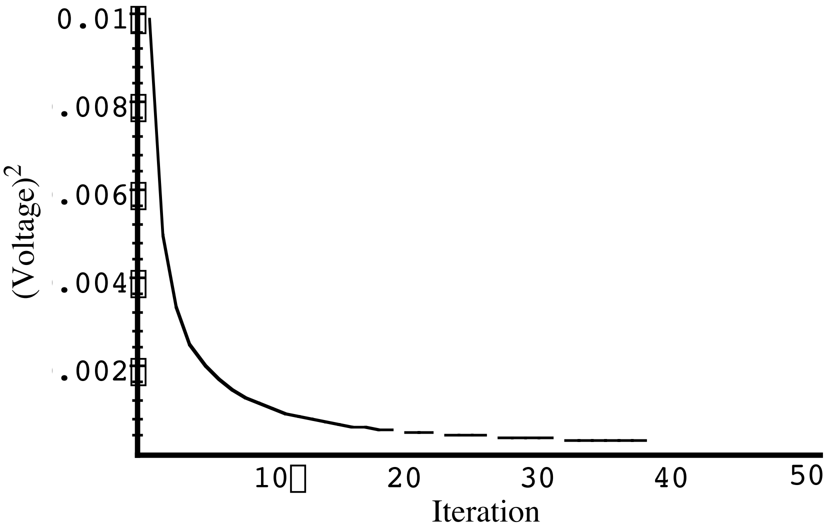


**Figure 3-2.** After 50 iterations, our initial (rough) error covariance $P_k^-$ choice of 1 has settled to about 0.0002 (Volts$^2$).

In section 1 under the topic "Filter Parameters and Tuning" we briefly discussed changing or "tuning" the parameters $Q$ and $R$ to obtain different filter performance. In Figure 3-3 and Figure 3-4 below we can see what happens when $R$ is increased or decreased by a factor of 100 respectively. In Figure 3-3 the filter was told that the measurement variance was 100 times greater (i.e. $R = 1$) so it was "slower" to believe the measurements.

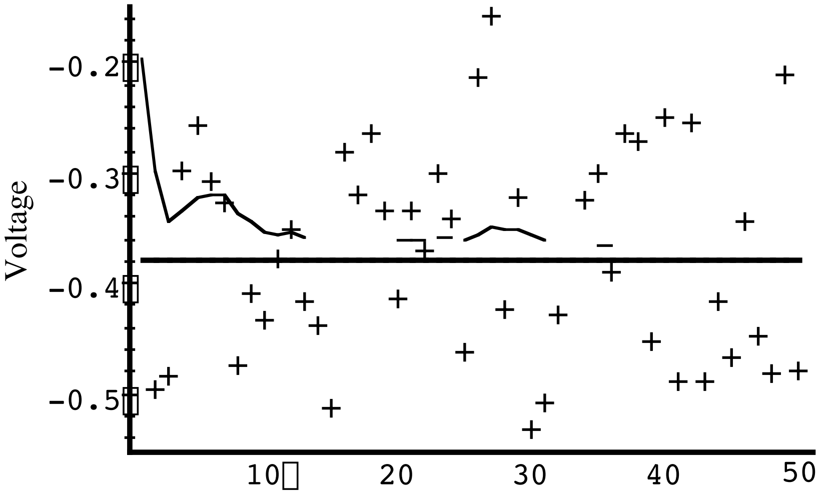


**Figure 3-3.** Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

In Figure 3-4 the filter was told that the measurement variance was 100 times smaller (i.e. $R = 0.0001$) so it was very "quick" to believe the noisy measurements.

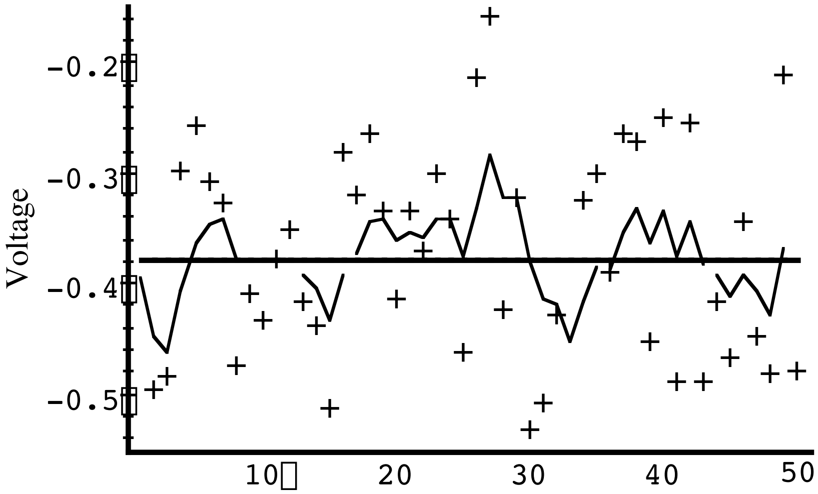


**Figure 3-4.** Third simulation: $R = 0.0001$ . The filter responds to measurements quickly, increasing the estimate variance.

While the estimation of a constant is relatively straight-forward, it clearly demonstrates the workings of the Kalman filter. In Figure 3-3 in particular the Kalman "filtering" is evident as the estimate appears considerably smoother than the noisy measurements.

# References

Brown92 Brown, R. G. and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, Second Edition*, John Wiley & Sons, Inc.

Gelb74 Gelb, A. 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.

Grewal93 Grewal, Mohinder S., and Angus P. Andrews (1993). Kalman Filtering Theory and Practice. Upper Saddle River, NJ USA, Prentice Hall.

Jacobs93 Jacobs, O. L. R. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.

Julier96 Julier, Simon and Jeffrey Uhlman. "A General Method of Approximating Nonlinear Transformations of Probability Distributions," Robotics Research Group, Department of Engineering Science, University of Oxford [cited 14 November 1995]. Available from http://www.robots.ox.ac.uk/~siju/work/publications/Unscented.zip.

Also see: "A New Approach for Filtering Nonlinear Systems" by S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, Proceedings of the 1995 American Control Conference, Seattle, Washington, Pages:1628-1632. Available from http://www.robots.ox.ac.uk/~siju/work/publications/ACC95_pr.zip.

Also see Simon Julier's home page at http://www.robots.ox.ac.uk/~siju/.

Kalman60 Kalman, R. E. 1960. "A New Approach to Linear Filtering and Prediction Problems," Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

Lewis86 Lewis, Richard. 1986. *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.

Maybeck79 Maybeck, Peter S. 1979.*Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.

Sorenson70 Sorenson, H. W. 1970. "Least-Squares estimation: from Gauss to Kalman," *IEEE Spectrum*, vol. 7, pp. 63-68, July 1970.