A KALMAN FILTERING TUTORIAL FOR UNDERGRADUATE STUDENTS

Matthew B. Rhudy[1], Roger A. Salguero[1] and Keaton Holappa[2]

[1]Division of Engineering, Pennsylvania State University, Reading, PA, 19610, USA
[2]Bosch Rexroth Corporation, Bethlehem, PA, 18017, USA

ABSTRACT

This paper presents a tutorial on Kalman filtering that is designed for instruction to undergraduate students. The idea behind this work is that undergraduate students do not have much of the statistical and theoretical background necessary to fully understand the existing research papers and textbooks on this topic. Instead, this work offers an introductory experience for students which takes a more practical usage perspective on the topic, rather than the statistical derivation. Students reading this paper should be able to understand how to apply Kalman filtering tools to mathematical problems without requiring a deep theoretical understanding of statistical theory.

KEYWORDS

Data Processing, Kalman Filtering, Tutorial

1. INTRODUCTION

Kalman filtering is a useful tool for a variety of different applications. However, this technique is not easily accessible to undergraduate students due to the high level details in existing publications on this topic. While it may not be practical to expect undergraduates to obtain a deep and thorough understanding of the stochastic theory behind Kalman filtering techniques, it is reasonable to expect undergraduates to be capable of making use of this computational tool for different applications.

While there are some excellent references detailing the derivation and theory behind the Kalman filter [1,2,3], this article aims to take a more teaching-based approach to presenting the Kalman filter from a practical usage perspective. The goal of this work is to have undergraduate students be able to use this guide in order to learn about and implement their own Kalman filter. One of the major differences between this work and the current state of the art Kalman filtering tutorial [3] is that the statistical theory is minimized, and focus is given to developing skills in implementing Kalman filters, rather than to understand the inner workings.

2. WHAT IS KALMAN FILTERING

So what is a Kalman filter? Let us start by breaking it down. The "Kalman" part comes from the primary developer of the filter, Rudolf Kalman [4]. So this is just a name that is given to filters of a certain type. Kalman filtering is also sometimes called "linear quadratic estimation." Now let us think about the "filter" part. All filters share a common goal: to let something pass through while something else does not. An example that many people can relate to is a coffee filter. This coffee filter will allow the liquid to pass through, while leaving the solid coffee grounds behind. You can also think about a low-pass filter, which lets low frequencies pass through while attenuating high frequencies. A Kalman filter also acts as a filter, but its operation is a bit more complex and harder to understand. A Kalman filter takes in information which is known to have some error, uncertainty, or noise. The goal of the filter is to take in this imperfect information, sort out the useful parts of interest, and to reduce the uncertainty or noise. Diagrams of these three filtering examples are offered in Figure 1.

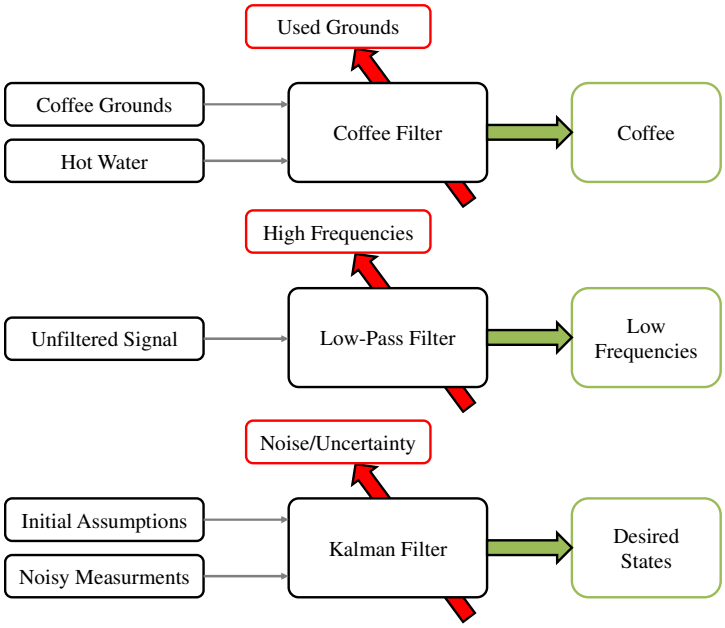

Figure 1. Three Example Diagrams of Types of Filters

3. IMPLEMENTATION OF A KALMAN FILTER

3.1. Filtering Problem Definition

The Kalman filter is designed to operate on systems in linear state space format, i.e.

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \quad (1)$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \quad (2)$$

where the variable definitions and dimensions are detailed in Table 1.

Table 1. Dimensions of Discrete Time System Variables

Variable	Description	Dimension
x	State Vector	$n_x \times 1$
y	Output Vector	$n_y \times 1$
u	Input Vector	$n_u \times 1$
w	Process Noise Vector	$n_x \times 1$
v	Measurement Noise Vector	$n_y \times 1$
F	System Matrix – State	$n_x \times n_x$
G	System Matrix – Input	$n_x \times n_u$
H	Observation Matrix	$n_y \times n_x$

Note that the number of states, outputs, and inputs are independent, and therefore the matrices **G** and **H** can be non-square, but **F** is always a square matrix. The state dynamics are described by (1), while the output equations are given by (2). Vectors are denoted in bold font, so (1) and (2) are vector equations.

The state vector, **x**, are the values that will be estimated by the filter. Using the filter analogy, the components of this vector are the things that you want to pass through the filter. Sometimes you may include more items in the state vector than you really care about if they are necessary in order to determine what you really want. For example if you want to determine the position of an object using information about the acceleration, you will likely need to determine the velocity as well. This is an important distinction between the state and output vectors of the system. While in other situations the "output" is what you are trying to get, for state estimation problems using Kalman filtering, the "state" is actually the desired result.

The output vector, **y**, is not what you are trying to get out of the filter, but rather what you are able to measure. You need to be able to express your measurements in terms of the states so that you can compare them with the measurements, i.e. you need to get apples to apples to know how much (or little) to correct. Not all measurements need to appear in the output vector for a particular formulation. Sometimes certain measurements are necessary for use in the state dynamics (1). The output vector should consist of values which can be both determined mathematically from the states as well as through some independent measurement system, i.e. the measurements are not used elsewhere in the filter.

The input vector, **u**, is probably the trickiest part of the Kalman filter definitions. This vector contains information that is necessary coming into the filter in order to define the system dynamics. These values can be sensor measurements, however in this case the uncertainty in these input values would need to be considered. In general, when defining your system equations (1) and (2), after determine the necessary states, any other terms which appear in the filter that do not need to be estimated as states can be considered as inputs. For example, in order to find the dynamics of a velocity state, you might include an acceleration measurement as an input.

The terms **w** and **v** which correspond to the process and measurement noise vectors for the system are interesting in that they do not typically appear directly in the equations of interest. Instead, these terms are used to model the uncertainty (or noise) in the equations themselves. This can manifest itself in a number of ways. The first is modelling error, which is uncertainty in the equation itself. How much can we trust this equation? Some equations, when derived from physics principles for instance, have negligible modelling error. However, some situations contain heuristically defined equations which may not be used with full confidence in their correctness. In this case, a certain amount of error can be considered to appear in the equation in the form of **w** or **v**. Another possible source of error is error in sensor measurements used in the equations. The **w** and **v** terms can be used to include the errors due to sensor measurements. In fact, this is the most common use of v in the output equations to account for the error in the measurement of the output. Note that this error is not in the equation itself, but accounts for how good the equation should be relative to the measurement. Note that **w** and **v** are not actually implemented in the calculations of (1) and (2) since they are assumed to be random errors with zero mean, but rather are just used to determine information about the process and measurement noise covariance matrices **Q** and **R**. If you are unfamiliar with the definition of a covariance matrix, please see Appendix A for more information.

What about the system matrices **F**, **G**, and **H**? These matrices will depend on the considered problem, and are used in order to represent the equations as a linear system of states and inputs. **F** will contain the coefficients of the state terms in the state dynamics (1), while **H** serves a similar function in the output equations (2). The **G** matrix contains coefficients of the input terms in the state dynamics (1). These matrices can in general vary with time, but cannot change with respect to the states or inputs. For many problems, these matrices are constant.

3.2. Kalman Filtering Algorithm

The Kalman filter uses a prediction followed by a correction in order to determine the states of the filter. This is sometimes called predictor-corrector, or prediction-update. The main idea is that using information about the dynamics of the state, the filter will project forward and predict what the next state will be. A simple example of this would be if I know where I was before (previous state), and how fast I was moving (state dynamics), I can guess where I am at now (current state). This can be thought of as a numerical integration technique such as Euler's method or Runge-Kutta [5]. The correction or update part then involves comparing a measurement with what we predict that measurement should be based on our predicted states.
The Kalman filtering technique is now discussed in equation format. Starting from some initial state estimate, $\hat{\mathbf{x}}_0$, and initial state error covariance matrix, $\mathbf{P}_0$, the predictor-corrector format is applied recursively at each time step, e.g. using a loop. First, the state vector is predicted from the state dynamic equation using

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{3}$$

where $\hat{\mathbf{x}}_{k|k-1}$ is the predicted state vector, $\hat{\mathbf{x}}_{k-1}$ is the previous estimated state vector, $\mathbf{u}$ is the input vector, and $\mathbf{F}$ and $\mathbf{G}$ are matrices defining the system dynamics. Note that the subscript $k|k$-1 is read as "k given k-1" and is a shorthand notation for the state at discrete time k given its previous state at discrete time k-1, i.e. this is the prediction of the state using the system model projected forward one step in time. Next, the state error covariance matrix must also be predicted using

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^{T} + \mathbf{Q}_{k-1} \tag{4}$$

where $\mathbf{P}_{k|k-1}$ represents the predicted state error covariance matrix, $\mathbf{P}_{k-1}$ is the previous estimated state error covariance matrix, and $\mathbf{Q}$ is the process noise covariance matrix. Again, $k|k$-1 is indicating that this is the expected covariance matrix at k based on the system model and the covariance at k-1. Once the predicted values are obtained, the Kalman gain matrix, $\mathbf{K}_k$, is calculated by

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{T}\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{T} + \mathbf{R}_k\right)^{-1} \tag{5}$$

where $\mathbf{H}$ is a matrix necessary to define the output equation and $\mathbf{R}$ is the measurement noise covariance. The state vector is then updated by scaling the "innovation," which is the difference between the measurement of the output, $\mathbf{z}_k$, and the predicted output, $\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}$ (sometimes called $\hat{\mathbf{y}}_{k|k-1}$), by the calculated Kalman gain matrix in order to correct the prediction by the appropriate amount, as in

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\right) \tag{6}$$

Similarly, the state error covariance is updated by

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1} \tag{7}$$

where $\mathbf{I}$ is an identity matrix.

3.3. Comments on Notation

A major notational difference that occurs between sources is the use of the discrete time index, k. There are essentially three different discrete time formats which can be found in the literature. Each one is equivalent, but results in different subscripts in (1) and (2). The different time

formatting is summarized in Table 2. Other different notational differences found in the various resources on Kalman filtering are summarized in Table 3, including example sources which use this notation.

Table 2. Different Discrete Time Notational Formatting

Time Format	State Dynamics (1)	Output Equations (2)
k and $k-1$ [1]	$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1}$	$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$
k+1 and k with output k [2,3]	$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k\mathbf{u}_k + \mathbf{w}_k$	$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$
k+1 and k with output k+1 [8]	$\mathbf{x}_{k+1} = \mathbf{F}_k\mathbf{x}_k + \mathbf{G}_k\mathbf{u}_k + \mathbf{w}_k$	$\mathbf{y}_{k+1} = \mathbf{H}_{k+1}\mathbf{x}_{k+1} + \mathbf{v}_{k+1}$

Table 3. Summary of Notational Differences in the Literature

Term	This Tutorial	Other Notation
State Vector	$\mathbf{x}$	z [6]
Output Vector	$\mathbf{y}$	z [2]
System State Matrix	$\mathbf{F}$	A [3,6], Φ [7,8]
Input matrix	$\mathbf{G}$	B [3]
Observation Matrix	$\mathbf{H}$	C [6]
State Error Covariance Matrix	$\mathbf{P}$	Σ [2]
Discrete time index	k	n [6]

3.4. Continuous vs. Discrete Time Kalman Filters

This article focuses on the discrete time Kalman filter. There is also a continuous time counterpart which is sometimes called the Kalman-Bucy filter [1]. Since the majority of Kalman filtering applications are implemented in digital computers, the implementation of Kalman filters in continuous time is not practical in many situations. Because of this, the continuous time cases are left to other sources [1,7]. Note that this does not mean that continuous system models cannot be approached with a discrete time Kalman filter. Consider a continuous system of the following form

$$\dot{\mathbf{x}}(t) = \mathbf{F}_c(t)\mathbf{x}(t) + \mathbf{G}_c(t)\mathbf{u}(t) + \mathbf{w}(t) \tag{8}$$

where $\mathbf{F}_c$ and $\mathbf{G}_c$ indicate the system matrices for a continuous system, and $\mathbf{w}_c$ is the continuous time process noise vector with covariance matrix $\mathbf{Q}_c$. The continuous model can be discretized, e.g. using a 1st order approximation, as in

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta t\dot{\mathbf{x}}_k \tag{9}$$

where Δt is the sampling time, or the duration of a single discrete time step, thus resulting in $t = k\Delta t$. Then, including the continuous time model gives

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta t\left[\mathbf{F}_c(t)\mathbf{x}(t) + \mathbf{G}_c(t)\mathbf{u}(t) + \mathbf{w}_c(t)\right] \tag{10}$$

which can be simplified to obtain the following discrete time system

$$\mathbf{x}_k = \left(\mathbf{I} + \Delta t\mathbf{F}_c(k\Delta t)\right)\mathbf{x}_{k-1} + \Delta t\mathbf{G}_c(k\Delta t)\mathbf{u}_k + \Delta t\mathbf{w}_c(k\Delta t) \tag{11}$$

Note that the following definitions can be made to match the general discrete time system as given in (1)

$$\begin{aligned}\mathbf{F}_{k-1} &= \mathbf{I} + \Delta t \mathbf{F}_c(k\Delta t) \\ \mathbf{G}_{k-1} &= \Delta t \mathbf{G}_c(k\Delta t) \\ \mathbf{w}_{k-1} &= \Delta t \mathbf{w}_c(k\Delta t)\end{aligned} \tag{12}$$

The covariance matrices for the process noise are therefore related by

$$\mathbf{Q}_{k-1} = (\Delta t)^2 \mathbf{Q}_c(t) \tag{13}$$

The output equations between continuous time and discrete time are equivalent, i.e. it does not matter if *k* or *t* is used for time in the equation, it is the same relationship.

3.5. Effect of Noise Covariance Assumptions

The selection of the assumed covariance matrices **Q**, **R**, and $\mathbf{P}_0$ can have a significant effect on the estimation performance of a Kalman filter. The selection of $\mathbf{P}_0$ is coupled with the assumed initial state, and affects the initial convergence of the filter. In many situations, the effect of $\mathbf{P}_0$ is not significant, and in fact it is often arbitrarily initialized to an identity matrix for simplicity. The effects of **Q** and **R** and much more significant and they affect the overall performance of the filter. A basic way to think of **Q** and **R** is that they are weighting factors between the prediction (state) equations and the measurement (output) equations. This ratio is shown within the Kalman gain equation (5). Considering a larger **Q** is equivalent to considering larger uncertainty in the state equations, which is equivalent to trusting the result of these equations less, which effectively means that the filter should correct more with the measurement update. Similarly, considering a larger **R** is equivalent to considering larger uncertainty in the measurement, which is equivalent to trusting the measurement less, which effectively means that the filter should correct less with the measurement update. A diagram detailing this phenomenon is shown in Figure 2.

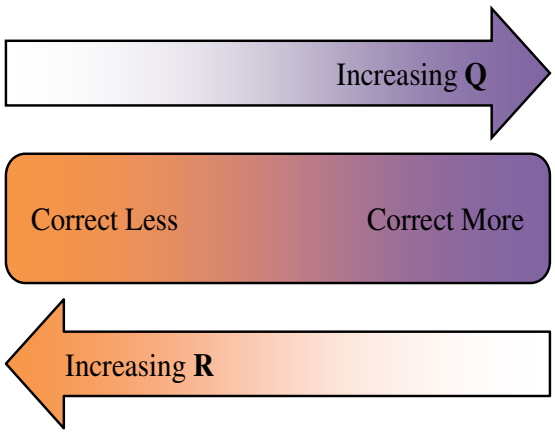

Figure 2. Diagram of Noise Covariance Assumptions Effect on Filter Operation

3.6. Has a linear transformation

The purpose of the **H** matrix is to essentially convert the states into outputs. For the linear Kalman filter this occurs by considering some linear combination of the states, i.e. **H** is a linear transformation. In some simple cases, **H** is just used to select certain states which are measured, when other states are not. For example, if the first, second, and fifth states of a 5-dimensional state vector are measureable, the **H** matrix would be defined as

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

such that

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k$$

$$\mathbf{y}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_5 \end{bmatrix} \tag{15}$$

The **H** matrix can also be used to consider scaling effects. For example, if the radius is a state, but the diameter can be measured, the output equation could be represented by

$$\begin{gathered} \mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k \\ \mathbf{y}_k = d_k \qquad \mathbf{H}_k = 2 \qquad \mathbf{x}_k = r_k \end{gathered} \tag{16}$$

Other uses of **H** could be to consider combinations of states. For example, if the lengths of 3 sides of a triangle are states, but the perimeter can be measured, the output equation could be represented by

$$\mathbf{y}_k = x_1 + x_2 + x_3, \quad \mathbf{H}_k = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{x}_k = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{17}$$

4. A LINEAR KALMAN FILTERING EXAMPLE

In order to illustrate the use of the Linear Kalman Filter, a simple example problem is offered. This example considers a simple object in freefall assuming there is no air resistance. The goal of the filter is to determine the position of the object based on uncertain information about the initial position of the object as well as measurements of the position provided by a laser rangefinder.
Using particle kinematics, we expect that the acceleration of the object will be equal to the acceleration due to gravity. Defining the height of the object in meters, *h*, we have:

$$\ddot{h}(t) = -g \tag{18}$$

where *g* is the acceleration due to gravity (g = 9.80665 m/s^2). Integrating this relationship over a small time interval, Δt, gives

$$\ddot{h}(t) = \frac{\dot{h}(t) - \dot{h}(t - \Delta t)}{\Delta t} = -g \tag{19}$$

which is a backward difference equation, which is useful for Kalman filtering applications due to the recursive structure of the filter, i.e. each time step in the Kalman filter always references the previous time step. Simplifying this expression gives

$$\dot{h}(t) = \dot{h}(t - \Delta t) - g\Delta t \tag{20}$$

Integrating again yields a commonly used kinematic equation relating successive positions of a particle with respect to time for constant acceleration

$$h(t) = h(t-\Delta t) + \dot{h}(t-\Delta t) - \frac{1}{2} g (\Delta t)^2 \tag{21}$$

Rather than consider these equations in terms of continuous time, t, it is beneficial to rewrite the equations in terms of a discrete time index, k, which is defined by $t = k\Delta t$. Additionally, discrete time values are traditionally written as subscripts rather than as a functional dependence, i.e.

$$\begin{aligned} h(t) &= h(k\Delta t) = h_k \\ h(t-\Delta t) &= h(k\Delta t - \Delta t) = h(\Delta t(k-1)) = h_{k-1} \end{aligned} \tag{22}$$

Rewriting the kinematic relationships for the example problem in terms of the discrete time variable, k, gives

$$\begin{aligned} \dot{h}_k &= \dot{h}_{k-1} - g\Delta t \\ h_k &= h_{k-1} + \dot{h}_{k-1}\Delta t - \frac{1}{2} g (\Delta t)^2 \end{aligned} \tag{23}$$

From these kinematic expressions, we see that we have two equations describing the motion of the object: one for velocity and one for position. Since we are interested in estimating the position, we know that the position must be included as a state. However, since we can see that the velocity also appears in the position equation, it is also necessary to obtain that information. One way to make sure that we have information about the velocity is to additionally include the velocity as a state. As a result, we now define the state vector for the Kalman filter as

$$\mathbf{x}_k = \begin{bmatrix} h_k \\ \dot{h}_k \end{bmatrix} \tag{24}$$

which results in the following expression

$$\mathbf{x}_k = \begin{bmatrix} h_{k-1} + \dot{h}_{k-1}\Delta t - \frac{1}{2} g (\Delta t)^2 \\ \dot{h}_{k-1} - g\Delta t \end{bmatrix} \tag{25}$$

With this definition, we can rewrite these equations in terms of the state vector, $\mathbf{x}$, as in

$$\mathbf{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} -\frac{1}{2}(\Delta t)^2 \\ -\Delta t \end{bmatrix} g \tag{26}$$

Now, we have the problem in necessary format for the Kalman filter

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{27}$$

where the system matrices $\mathbf{F}$ and $\mathbf{G}$ are given by

$$\mathbf{F}_{k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{G}_{k-1} = \begin{bmatrix} -\frac{1}{2}(\Delta t)^2 \\ -\Delta t \end{bmatrix} \tag{28}$$

and the input vector can be defined as

$$\mathbf{u}_{k-1} = g \tag{29}$$

The reason for selecting g as an input is because this information is necessary to define the state dynamics, but g is not defined as a state in the filter. This additional information is "input" into the equations at each time step. Note that for this particular problem the values of **F**, **G**, and **u** do not vary with respect to k. It is important to note in these equations that there is no process noise uncertainty term, **w**. For this particular set of equations, we are assuming that there are no errors in the equations themselves. For this problem, this is an assumption, as there could be disturbances from air resistance or other sources. However, assuming that these errors are small enough to ignore, we do not need to model the process noise for this problem. Because of this, the process noise covariance matrix, **Q**, can be set to zero.

Next, we need to consider the measurement part of the system. Let us consider a scenario where the position of the object can be measured using a laser rangefinder with 2 m standard deviation of error. Because the position is what can be measured, we need to define an output equation that gives the position as a function of the states of the filter. There is some uncertainty in the measurement, which is noted in the equations by the measurement noise vector, **v**:

$$\mathbf{y}_k = h_k + \mathbf{v}_k \tag{30}$$

Since the position can be written in terms of the state vector, this can be rewritten as

$$\mathbf{y}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{v}_k \tag{31}$$

Now, we have the output equations of the system defined in the proper form as in (2), where the system matrix, **H**, is given by

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{32}$$

The considered measurement system has a standard deviation of error of 2 m, which is a variance of 4 m^2. Because of this, and the fact that there is only one term in the output vector, the resulting measurement noise covariance matrix reduces to a scalar value, $\mathbf{R} = 4$ m^2.

In addition to the measurement noise, we also need to consider any uncertainty in the initial state assumption. The initial position is approximately known to be 105 m before the ball is dropped, while the actual initial position is 100 m. The initial guess was roughly determined, and should therefore have a relatively high corresponding component in the assumed initial covariance. For this example, we consider an error of 10 m^2 for the initial position. For the initial velocity, we assume that the object starts from rest. This assumption is fairly reasonable in this case, so a smaller uncertainty value of 0.01 m^2/s^2 is assumed. To aid the reader in the implementation of this example, the necessary parameters and definitions for this filtering application are summarized in Table 4.

Table 4. Example Problem Definitions

Term	Definition
State Vector	$\mathbf{x}_k = \begin{bmatrix} h_k \\ \dot{h}_k \end{bmatrix}$
Output Vector	$\mathbf{y}_k = h_k + \mathbf{v}_k$
Input Vector	$\mathbf{u}_{k-1} = g$
System State Matrix	$\mathbf{F}_{k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$
Input Matrix	$\mathbf{G}_{k-1} = \begin{bmatrix} -\frac{1}{2}(\Delta t)^2 \\ -\Delta t \end{bmatrix}$
Observation Matrix	$\mathbf{H}_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$
Process Noise Covariance Matrix	$\mathbf{Q}_{k-1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
Measurement Noise Covariance Matrix	$\mathbf{R}_k = 4$
True Initial State Vector	$\mathbf{x}_0 = \begin{bmatrix} 100 \\ 0 \end{bmatrix}$
Assumed Initial State Vector	$\hat{\mathbf{x}}_0 = \begin{bmatrix} 105 \\ 0 \end{bmatrix}$
Assumed Initial State Error Covariance Matrix	$\mathbf{P}_0 = \begin{bmatrix} 10 & 0 \\ 0 & 0.01 \end{bmatrix}$
Time Increment	$\Delta t = 0.001$

In order to complete this Kalman filtering example, we want to simulate values for the "true" states of the system as well as the noisy measurements. In a real application problem, this step is not done, as the "true" states are not known and the measurements are obtained from the measurement system.

The provided Kalman filtering example was executed over 1000 time steps, and the estimated and true states are offered in Figure 3. Additionally, the estimation errors are shown in Figure 4. The code used to generate these figures is provided in Appendix B. It is recommended, however, that the reader of this tutorial try to implement their own version of the code before consulting with the solution in Appendix B.

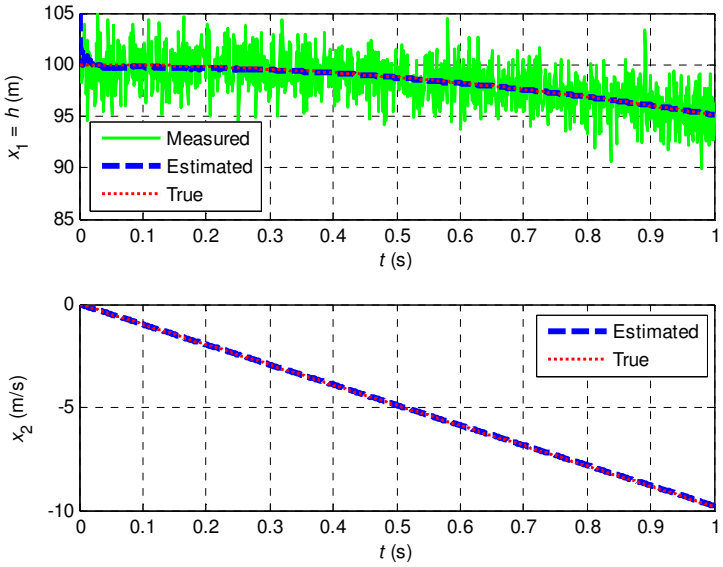

Figure 3. Kalman Filtering Example Estimated and True States

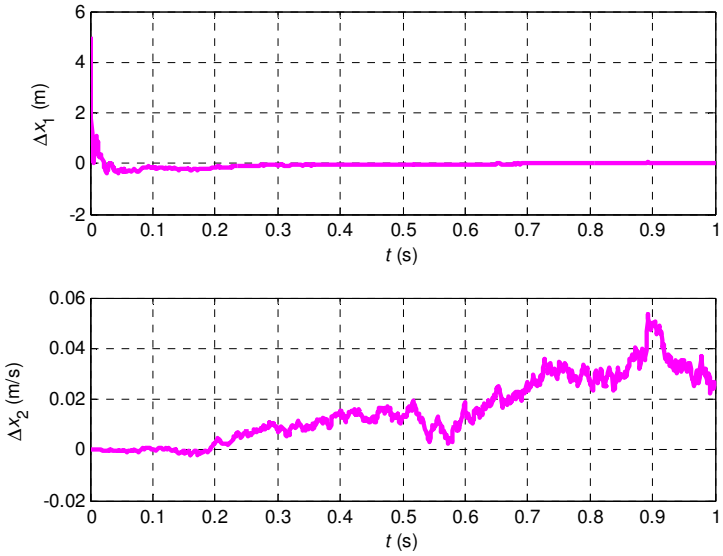

Figure 4. Kalman Filtering Example Estimation Errors

5. THE EXTENDED KALMAN FILTER

The Extended Kalman Filter (EKF) is just an extension of the Kalman filter to nonlinear systems. This means that the difference with the EKF is that the state and/or the output equations can contain nonlinear functions. So rather than considering systems of the form (1) and (2), the EKF can consider a more general nonlinear set of equations:

$$\mathbf{x}_k = \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) + \mathbf{w}_{k-1} \tag{33}$$

$$\mathbf{y}_k = \mathbf{h}\left(\mathbf{x}_k\right) + \mathbf{v}_k \tag{34}$$

where **f** is the vector-valued nonlinear state transition function and **h** is the vector-valued, nonlinear observation or output function. In order to handle the nonlinear functions, Jacobian

matrices are calculated in order to linearize the equations at each time step about the previous state, as in

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1}} \tag{35}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}} \tag{36}$$

Using these calculated Jacobian matrices, the EKF algorithm can be summarized as

$$\begin{aligned}
&\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}\right) \\
&\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \\
&\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \\
&\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left[\mathbf{z}_k - \mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}\right)\right] \\
&\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1}
\end{aligned} \tag{37}$$

Note that the EKF equations are very similar to the Kalman filter equations, but with a few exceptions. One difference is that the values of **F** and **H** are determined from the Jacobian matrix calculations at each time step. The other difference is that the nonlinear functions **f** and **h** are used directly in the prediction of the states and the outputs, rather than use the linearized version, i.e. the EKF uses $\mathbf{f}\left(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}\right)$ instead of $\mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1}$, and $\mathbf{h}\left(\hat{\mathbf{x}}_{k|k-1}\right)$ instead of $\mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}$. Besides these differences, the operation and usage of the EKF is comparable to that of the Kalman filter.

5.1. Nonlinear Observation/Output Equations

The purpose of the observation or output equations is to map the existing set of states to a set of "outputs" that are able to be measured within the considered system. As an example, consider a system with states describing the 2-dimensional Cartesian position of an object (x, y). A tracking system is used to follow the object and measures the distance to the object, r, and the angle that it makes with respect to the horizontal, θ. Note that this measurement system offers a polar representation of the 2-dimensional coordinates. With this system, the output equations would then be defined as

$$\begin{aligned}
r &= \sqrt{x^2 + y^2} \\
\theta &= \tan^{-1}\left(\frac{y}{x}\right)
\end{aligned} \tag{38}$$

For this system, the Jacobian matrix, **H**, would be calculated using

$$H = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{x}{\sqrt{x^2 + y^2}} & \frac{y}{\sqrt{x^2 + y^2}} \\ \frac{-y}{x^2 + y^2} & \frac{x}{x^2 + y^2} \end{bmatrix} \tag{39}$$

This expression could then be used at each time step in order to evaluate the value of **H** at the current predicted state.

5.2. Other Nonlinear Kalman Filtering Techniques

The Extended Kalman Filter (EKF) is not the only nonlinear Kalman filtering technique. Perhaps the most common alternative to the EKF is the Unscented Kalman Filter (UKF) which is sometimes referred to as a Sigma-Point Kalman Filter (SPKF). Rather than use Jacobian matrices to handle the nonlinear in the system, the UKF uses a statistical linearization technique called the unscented transformation [9]. The UKF has demonstrated some advantages in more accurate linearization, particularly in highly nonlinear problems [9,10], however in some situations it has been shown to produce similar results to that of the EKF [11]. The EKF tends to be more computationally efficient than the UKF, thus making it more desirable for real-time applications [11,12]. Additionally, the EKF is a bit easier to understand conceptually, which is beneficial for student use. The UKF, however, can provide superior linearization in certain situations, and can also be easier to implement since there is no need to calculate Jacobian matrices [10]. For details on the implementation of the UKF, see [13] or [14].

6. CONCLUSION

This work provided a tutorial for Kalman filtering that is geared toward undergraduate students with little or no experience on this topic. Detailed descriptions and an example problem were offered in order to help aid in the basic understanding of this complex topic.

ACKNOWLEDGEMENTS

This work was partially supported by the Lafayette College EXCEL Scholar Program for undergraduate research and the Rodney A. Erickson Discovery Grant Program for undergraduate research.

REFERENCES

[1] Simon, D., *Optimal State Estimation*, Wiley, New York, 2006.
[2] Anderson, B. D. O., and Moore, J. B., *Optimal Filtering*, Prentice-Hall, NJ, 1979.
[3] Welch, G., and Bishop, G, "An introduction to the Kalman filter," Technical Report TR 95-041, University of North Carolina, Department of Computer Science, 1995.
[4] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Trans. of the ASME – Journal of Basic Engineering*, March 1960, pp. 35-45.
[5] Kreyszig, E., *Advanced Engineering Mathematics*, 9th Ed., Wiley, NY, 2006.
[6] Reif, K., Günther, S., Yaz, E., and Unbehauen, R., "Stochastic Stability of the Discrete-Time Extended Kalman Filter," *IEEE Trans. on Automatic Control*, Vol. 44, No. 4, April, 1999.
[7] Jazwinski, A. H., *Stochastic Processes and Filtering Theory*, Academic, New York, 1970.
[8] Hargrave, P., "A tutorial introduction to Kalman filtering," *IEEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments*, Feb. 1989.
[9] Julier, S. and Uhlmann, J., "A New Extension of the Kalman Filter to Nonlinear Systems." *SPIE Proceedings Series*, 1997, Vol. 3068, pp. 182-193.
[10] Rhudy, M., and Gu, Y., "Understanding Nonlinear Kalman Filters, Part I: Selection between EKF and UKF," *Interactive Robotics Letters*, West Virginia University, June 2013. Link: http://www2.statler.wvu.edu/~irl/page13.html.
[11] Rhudy, M., Gu, Y., Gross, J., Gururajan, S., and Napolitano, M., "Sensitivity Analysis of Extended and Unscented Kalman Filters for Attitude Estimation," *AIAA Journal of Aerospace Information Systems*, Vol. 10, No. 3, March 2013, pp. 131-143. doi: 10.2514/1.54899.
[12] Gross, J., Gu, Y., Rhudy, M., Gururajan, S., and Napolitano, M., "Flight Test Evaluation of GPS/INS Sensor Fusion Algorithms for Attitude Estimation," *IEEE Transactions on Aerospace Electronic Systems*, Vol. 48, No. 3, July 2012, pp. 2128-2139.
[13] Rhudy, M., and Gu, Y., "Understanding Nonlinear Kalman Filters, Part II: An Implementation Guide," *Interactive Robotics Letters*, West Virginia University, June 2013. Link: http://www2.statler.wvu.edu/~irl/page13.html.
[14] Wan, E., and van der Merwe, R., "The Unscented Kalman Filter," Chap. 7 in *Kalman Filtering and Neural Networks*, Wiley, New York, March 2002, pp. 221–282.

APPENDIX A: Explanation of Covariance Matrices

In order to introduce the concept of a covariance matrix, we first need to establish a basic understanding of the concept of variance, which is simply defined as the square of standard deviation. Typical notation for variance is σ^2, where σ denotes the standard deviation. This discussion aims to minimize the necessity of statistical theory, however some basic idea of statistics is necessary. Let us first consider a random variable, x, that is Gaussian or normally distributed (note that this is an assumption used by the Kalman filter). The Gaussian or normal distribution is the "bell-shaped curve," that is sometimes referenced, and can be parameterized by two values: mean, μ, and standard deviation, σ (or variance, σ^2). The statistical notation for this would be $x \sim N(\mu, \sigma^2)$, where the N indicates the normal distribution. If a point is sampled from this distribution, there is a higher probability that the value will be near the mean, and a lower probability that it will be significantly different from the mean. These probabilities are determined from the "bell-shaped curve." The larger the standard deviation, the greater the spread of values for the curve. Some example curves are offered in Figure 5. Note that the official name for these curves is Probability Density Function (PDF).

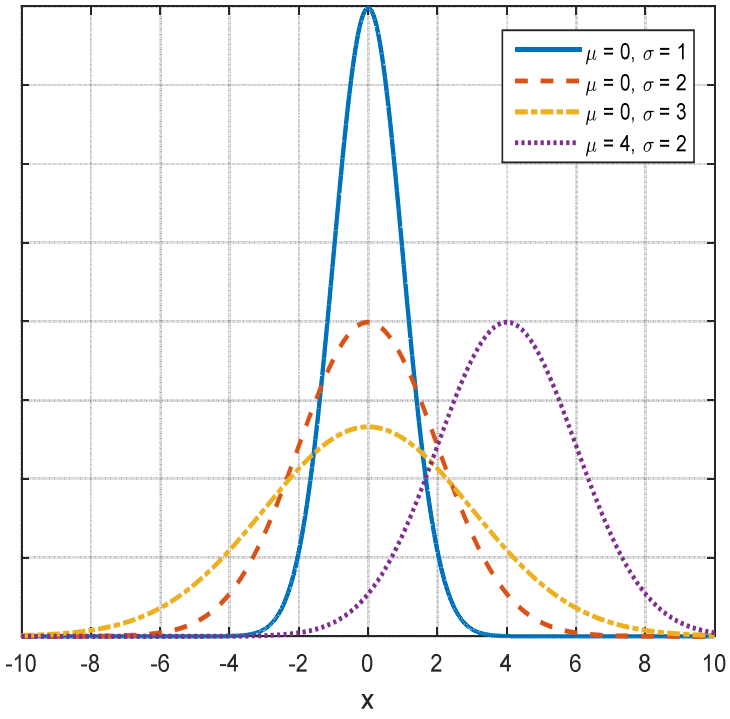

Figure 5. Example "Bell-Shaped Curves"

Now let us consider data points taken from a normal distribution with a mean $\mu = 0$ and standard deviation $\sigma = 2$. Taking 1000 samples from this distribution gives the result in Figure 6. Note that bars on drawn on the figure to show $\pm 1\sigma$ and $\pm 2\sigma$, which indicate 68% and 95% confidence intervals, respectively. I.e., we should expect that 68% of the data points should lie between the $\pm 1\sigma$ bars, and 95% of the data points should lie between the $\pm 2\sigma$ bars.

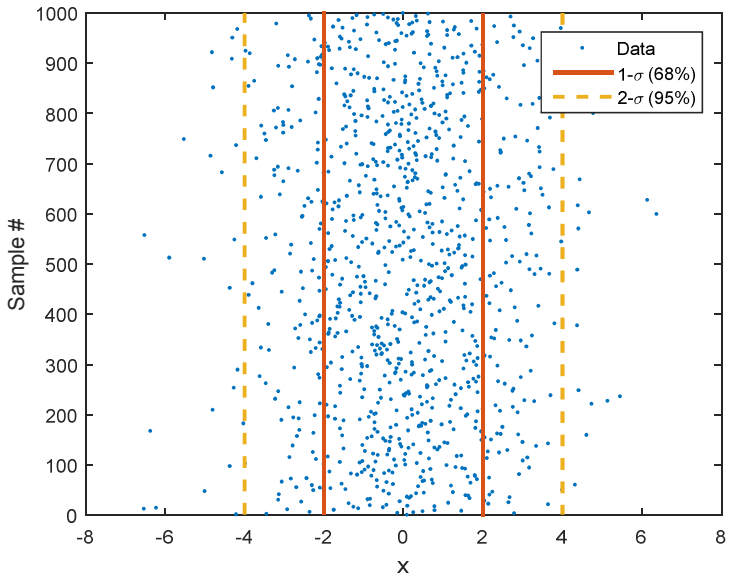

Figure 6. Sampling of 1000 Data Points from a Normal Distribution with $\mu = 0$ and $\sigma = 2$

Now we need to extend our discussion to multiple dimensions. I.e., how do we describe random variations in vectors as opposed to scalars? Here we will limit our discussion to two dimensions for simplicity, but the idea applies to higher dimensioned problems as well. When considering multiple dimensions, rather than just using a variance to describe a normal distribution, you now need to consider a covariance matrix. To illustrate this, let us consider a vector x with two components, x_1 and x_2. If we have a simple case where the distributions of the two components are independent, i.e. the variation in x_1 does not depend on x_2 and vice versa, the covariance matrix would be diagonal. Consider an example with means of 0 and covariance matrix

$$P = \begin{bmatrix} \sigma_{x_1}^2 & 0 \\ 0 & \sigma_{x_2}^2 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix} \tag{40}$$

Again sampling 1000 data points, we get the result shown in Figure 7. Note that our $\pm 1\sigma$ and $\pm 2\sigma$ boundaries are now described by ellipses rather than bars. It is possible, however, in multiple dimensions for the error in one component to depend on the other. In this case, there becomes a correlation between the two variables. Consider a second example that now considers cross-terms in the covariance matrix, as in

$$P = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1 x_2}^2 \\ \sigma_{x_1 x_2}^2 & \sigma_{x_2}^2 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 1 & 9 \end{bmatrix} \tag{41}$$

The resulting distribution is shown in Figure 8. Note the difference between Figure 7 and Figure **8**Figure 8 is the rotation of the distribution, thus describing the coupling in the two components. In the context of Kalman filtering, the important aspect of covariance matrices to understand is how the errors are related, e.g., does the error in the first measurement depend on the error in the second measurement, etc.

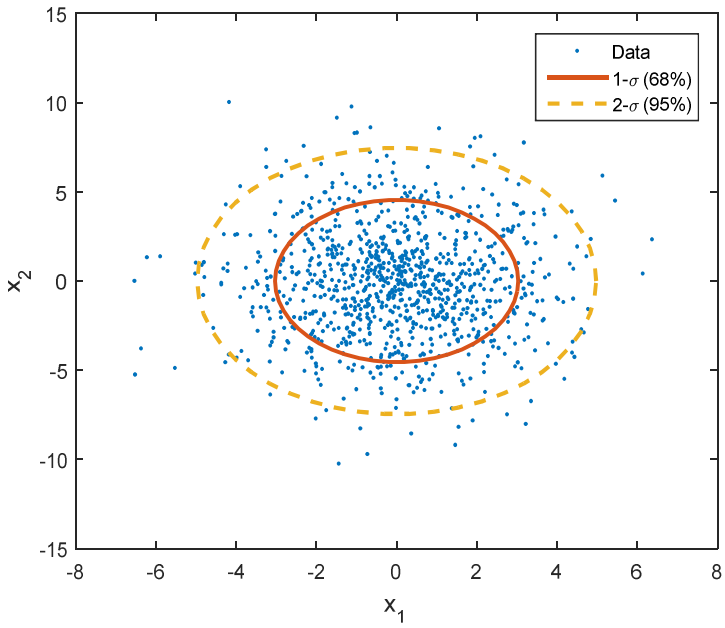

Figure 7. Two-Dimensional Example Distribution with Independent x_1 and x_2

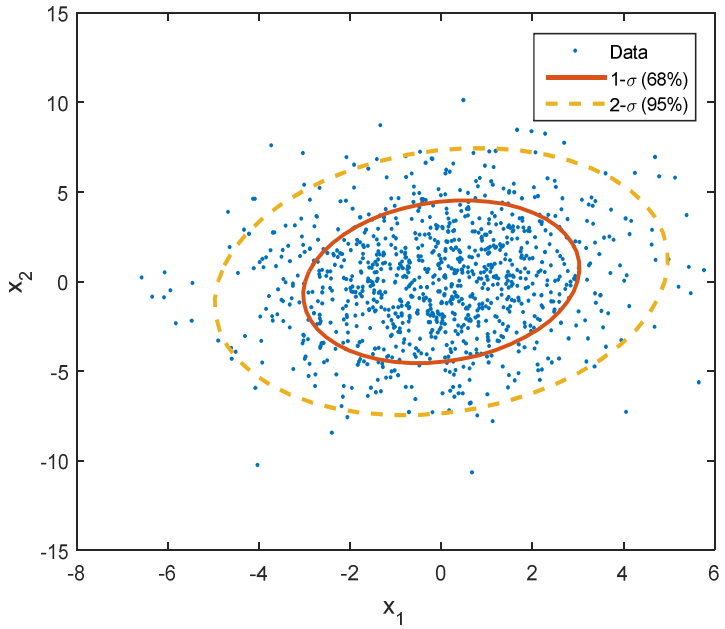

Figure 8. Two-Dimensional Example Distribution with Dependent x_1 and x_2

APPENDIX B: MATLAB Code for Example Problem

```
%% Simulate the true signal and measurement
% Define the system
N = 1000;                   % number of time steps
dt = 0.001;                 % Sampling time (s)
t = dt*(1:N);               % time vector (s)
F = [1, dt; 0, 1];          % system matrix - state
G = [-1/2*dt^2; -dt];       % system matrix - input
H = [1 0];                  % observation matrix
Q = [0, 0; 0, 0];           % process noise covariance
u = 9.80665;                % input = acceleration due to gravity (m/s^2)
I = eye(2);                 % identity matrix
% Define the initial position and velocity
y0 = 100;   % m
v0 = 0;     % m/s
% Initialize the state vector (true state)
xt = zeros(2, N);           % True state vector
xt(:, 1) = [y0; v0];        % True intial state
% Loop through and calculate the state
for k = 2:N
    % Propagate the states through the prediction equations
    xt(:, k) = F*xt(:, k-1) + G*u;
end
% Generate the noisy measurement from the true state
R = 4;                      % m^2/s^2
v = sqrt(R)*randn(1, N);    % measurement noise
z = H*xt + v;               % noisy measurement

%% Perform the Kalman filter estimation
% Initialize the state vector (estimated state)
x = zeros(2, N);            % Estimated state vector
x(:, 1) = [105; 0];         % Guess for initial state
% Initialize the covariance matrix
P = [10, 0; 0, 0.01];       % Covariance for initial state error
% Loop through and perform the Kalman filter equations recursively
for k = 2:N
    % Predict the state vector
    x(:, k) = F*x(:, k-1) + G*u;
    % Predict the covariance
    P = F*P*F' + Q;
    % Calculate the Kalman gain matrix
    K = P*H'/(H*P*H' + R);
    % Update the state vector
    x(:,k) = x(:,k) + K*(z(k) - H*x(:,k));
    % Update the covariance
    P = (I - K*H)*P;
end

%% Plot the results
% Plot the states
figure(1);
subplot(211);
plot(t, z, 'g-', t, x(1,:), 'b--', 'LineWidth', 2);
hold on; plot(t, xt(1,:), 'r:', 'LineWidth', 1.5)
xlabel('t (s)'); ylabel('x_1 = h (m)'); grid on;
legend('Measured','Estimated','True');
subplot(212);
plot(t, x(2,:), 'b--', 'LineWidth', 2);
hold on; plot(t, xt(2,:), 'r:', 'LineWidth', 1.5)
xlabel('t (s)'); ylabel('x_2 = v (m/s)'); grid on;
legend('Estimated','True');

% Plot the estimation errors
figure(2);
subplot(211);
plot(t, x(1,:)-xt(1,:), 'm', 'LineWidth', 2)
xlabel('t (s)'); ylabel('\Deltax_1 (m)'); grid on;
subplot(212);
plot(t, x(2,:)-xt(2,:), 'm', 'LineWidth', 2)
xlabel('t (s)'); ylabel('\Deltax_2 (m/s)'); grid on;
```

AUTHORS

Matthew Rhudy is currently an Assistant Professor in the Division of Engineering at the Pennsylvania State University, Berks Campus. Previously he was a Visiting Assistant Professor at Lafayette College in Easton, PA for 2 years. He received a Ph.D. in Aerospace Engineering from West Virginia University in 2013, a M.S. in Mechanical Engineering from the University of Pittsburgh in 2009, and a B.S. in Mechanical Engineering from the Pennsylvania State University in 2008.

Roger Salguero is an undergraduate mechanical engineering student at the Pennsylvania State University, Berks Campus. He is working on unmanned aircraft research through the Erickson Discovery Grant as well as the Frank Franco Undergraduate Research Award.

Keaton Holappa received a B.S. in Mechanical Engineering and a B.A. in Art from Lafayette College in 2016. He specializes in control theory, and is working on research related to the stability of micro-quadcopter swarms responding to musical inputs.