THE DISCRETE TIME KALMAN FILTER AND ITS APPLICATIONS

ALEC KNUTSEN, JONATHAN BUSH
DEPARTMENT OF MATHEMATICS
THE UNIVERSITY OF KANSAS
LAWRENCE, KANSAS USA
EMAIL: A991K109@KU.EDU; JLBUSH23@GMAIL.COM

Abstract. The discrete-time Kalman Filter is a recursively defined algorithm that can unbiasedly and optimally estimate the next state of a noisy discrete-time control system; the algorithm is named after one of its primary developers, Rudolf Kalman, who developed the algorithm in the 1960s. The Kalman Filter quickly emerged as a preeminent algorithm in control theory due to the optimality of the filter and its wide range of applicability and cross-disciplinary nature. For example, the Kalman Filter appears in position and navigation tracking, robotics, economics, medicine, and signal processing. Furthermore, one of the first implementations of the Kalman Filter concerned trajectory estimation of spacecrafts for NASA's Apollo Project. In this paper, we formulate the Kalman Filter that was first derived by Rudolf Kalman exactly fifty-six years ago. Then, we implement two examples of the Kalman Filter in physics settings to demonstrate the utility of the algorithm in applications. The filters in the examples are implemented using MATLAB.

1. Introduction

A filtering problem concerns forming the "best estimate" for some state in a system in which one only has some noisy measurements from the system. Noise is defined as some unexplained deviations in a measurement sample. For example, suppose we have a voltmeter that is set at a constant 0.5 V; however, we take some noisy measurements that vary around 0.5 V. In this filtering problem, we would like to determine a way to "filter" out the unexplained variations (noise) to bring the measurements closer to 0.5 V. In 1960, Rudolf E. Kalman published an article called "A new approach to linear filtering and prediction problems." This article introduced a recursive solution for a discrete time linear filtering problem. This solution is "the best" in the sense that it minimizes the variance of error (discussed in this paper), and it is recursive, allowing one to continually update as new measurements come in. After this development, Kalman Filtering has been used in a number of applications such as position tracking, robotics, and signal processing. The focus of this paper is to rigorously formulate the Kalman filter as a recursive, unbiased, linear, minimum variance estimator as in [6]; however, we tried to provide a more rigorous formulation that what was presented in these notes. In the first section, the setting for the Kalman Filter is described, as well as the assumptions that are made about the state-space model. In the next section, we derive the algorithm in six distinct steps. After the derivation, two implementations of the Kalman filter are described. All Matlab code is provided.

2. State-Space Model and Assumptions

In deriving the Kalman Filter, we assume we have the following model from [6]:

$$\boldsymbol{x_{k+1} = F_k x_k + G_k u_k + w_k} \tag{2.1}$$

$$\boldsymbol{z_k = H_k x_k + v_k} \tag{2.2}$$

List of Terms:

- Let k denote the discrete **time**
- x_k - The state of the system at time k where $x_k \in \mathbb{R}^{\mathrm{n}}$
- u_k - Input control at time k where $u_k \in \mathbb{R}^{\mathrm{m}}$
- z_k - An observation at time k where $z_k \in \mathbb{R}^{\mathrm{p}}$
- F_k - State transition matrix at time k with dimensions n x n
- G_k - Input transition matrix at time k with dimensions n x m
- H_k - Output transition matrix at time k with dimension p x n
- w_k - Process, system, or plant noise at time k where $w_k \in \mathbb{R}^n$
- v_k - measurement noise at time k where $v_k \in \mathbb{R}^p$

Assumptions:

- w_k is a **white-noise** vector $\forall\ k$
  1. The components of the vector w_k all have zero mean and finite variance $\forall\ k$
  2. The components of the vector w_k are independent $\forall\ k$
  3. Since w_k has zero mean $\forall\ k$, $cov(w_k, w_l) = E[w_k w_l^T]$. We assume w_k has a known covariance matrix $Q_k \implies cov(w_k, w_l) = E[w_k w_l^T] = \begin{cases} Q_k, & k = l \\ 0, & otherwise \end{cases}$
- v_k is a **white-noise** vector $\forall\ k$
  1. The components of the vector v_k all have zero mean and finite variance $\forall\ k$
  2. The components of the vector v_k are independent $\forall\ k$
  3. Since v_k has zero mean $\forall\ k$, $cov(v_k, v_l) = E[v_k v_l^T]$. We assume v_k has a known covariance matrix $R_k \implies cov(v_k, v_l) = E[v_k v_l^T] = \begin{cases} R_k, & k = l \\ 0, & otherwise \end{cases}$
- The noise w_k and v_l are uncorrelated $\implies cov(w_k, v_l) = E[w_k v_l^T] = 0\ \forall\ k, l$
- The initial system state, x_0 is uncorrelated to noise terms v_l, $w_k \implies cov(x_0, w_k) = E[x_0 w_k^T] = 0, cov(x_0, v_l) = E[x_0 v_l^T] = 0\ \forall\ k, l$
- The initial system state has a known mean $E[x_0]$ and error covariance $E[(\hat{x}_{0|0} - x_0)(\hat{x}_{0|0} - x_0)^T]$

Comments:

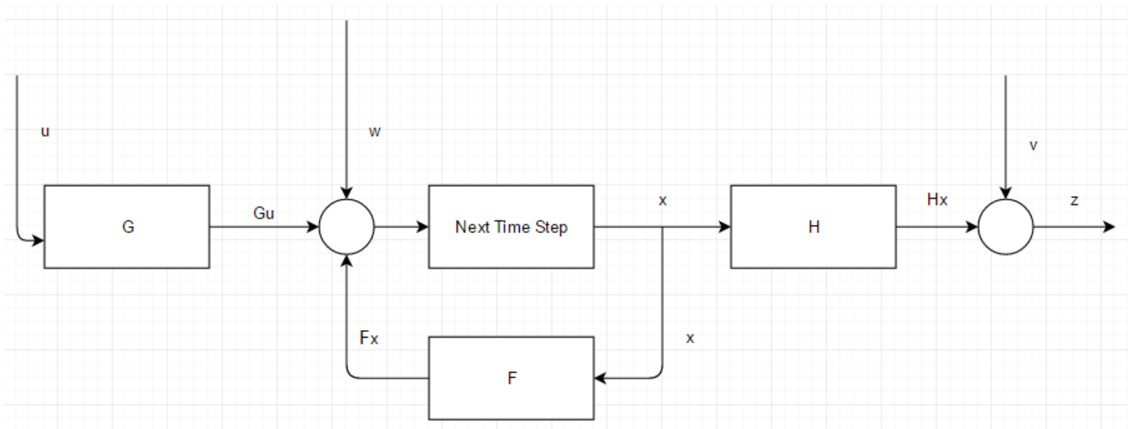

FIGURE 1. Picture of the State-Space Model described above

- When using the filter, the matrices F_k, G_k, and H_k are chosen based on the system one is modeling.
- Furthermore, one can also define an input control vector u_k based on the system dynamics.
- The model assumes one has gathered some set of measurements z_k from their system.
- The noise w_k, v_k need not be chosen, but instead one chooses the covariance matrices Q_k, R_k. Analysis of choosing these depends on the system and is beyond the scope of this paper.
- The initial state $\hat{x}_{0|0} = E[x_0]$ and error covariance matrix $P_{0|0} = E[(\hat{x}_{0|0} - x_0)(\hat{x}_{0|0} - x_0)^T]$ need to be defined as well. Analysis of choosing these initial values depends on the system and is beyond the scope of this paper.
- Finally, the states x_k are unknown and this is what one uses the Kalman Filter to estimate.

3. DERIVATION

The derivation of the Kalman Filter that follows is given in six steps. The first two steps develop two recursive, linear formulas for state estimators. Then, we prove that both these estimators are unbiased. Two more steps follow in which we derive linear, recursive formulas for the error covariance matrices of the two estimators. Finally, we derive a condition which makes our second estimator a minimum variance estimator.

3.1. 1st Recursive Formula for estimated states.

Let us denote $\hat{x}_{k+1|k}$ as an estimator for the state x_{k+1} given all observations $Z^k = z_1, z_2, ..., z_k$ as in [6]. Before, jumping into the derivation, we will need a couple definitions and theorems.

Definition 3.1 (Mean Squared Error)**.** The mean-squared error for the vector x and its estimator $\hat{x}$ is defined as:

$$E[||\hat{x} - x||^2] = E[(\hat{x}_1 - x_1)^2 + (\hat{x}_2 - x_2)^2 + ... + (\hat{x}_n - x_n)^2]$$

Definition 3.2 (Minimum Mean Squared Error Estimator (MMSE) from [4])**.** Let $z_1, z_2, ..., z_n$ be a discrete parameter process which represents measured values of a random variable Z that is related to a random variable X. The minimum mean squared error estimator is defined as the estimator which minimizes the mean squared error, and it is given by:

$$\hat{x} = E[X|Z_1 = z_1, Z_2 = z_2, ..., Z_n = z_n]$$

In the derivation of the Kalman Filter, we assume we have a discrete parameter process (set of measurements) $z_1, z_2, ..., z_k$ which are related to the states $x_1, x_2, ..., x_k$ through Equation 2.2. Thus, by the above definition, the minimum mean squared error estimator for the states given the observations $Z^k = z_1, z_2, ..., z_k$ is given by:

$$\hat{x}_{k+1|k} = E[x_{k+1}|Z^k]$$

Below, we will derive a recursive, linear formula for the MMSE.

$$
\begin{aligned}
\hat{x}_{k+1|k} &= E[x_{k+1}|Z^k] \\
&= E[F_k x_k + G_k u_k + w_k|Z^k] \ \text{ (Equation 2.1)} \\
&= E[F_k x_k|Z^k] + E[G_k u_k|Z^k] + E[w_k|Z^k] \ \text{ (Linearity of Expected Value)} \\
&= F_k E[x_k|Z^k] + G_k u_k E[1|Z^k] + E[w_k|Z^k] \ \text{ (Pull Out Constant Vectors, Matrices } F_k, G_k, u_k\text{)} \\
&= F_k E[x_k|Z^k] + G_k u_k + 0 \ \text{ (The noise } w_k \text{ is zero mean } \forall\ k\text{)} \\
&= F_k \hat{x}_{k|k} + G_k u_k
\end{aligned}
$$

Thus, we have a linear, recursive estimator (in terms of $\hat{x}_{k|k}$) that is also a MMSE. We will show that this estimator is unbiased later in the paper.

$$
\boldsymbol{\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k} \tag{3.1}
$$

3.2. **2nd Recursive Formula for estimated states.** Now, suppose we add a new observation z_{k+1}. We now update our state estimate to account for this new observation with the equation $\hat{x}_{k+1|k+1} = K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}z_{k+1}$ as done in [6]. This estimator is linear and recursive in terms of $\hat{x}_{k+1|k}$. In the next part of the paper, we will show that this estimator along with the previous estimator are both unbiased. Furthermore, we will show that this estimator is a MMSE.

$$
\boldsymbol{\hat{x}_{k+1|k+1} = K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}z_{k+1}} \tag{3.2}
$$

3.3. **Unbiasedness of State Estimator.**

Definition 3.3. We say an estimator $\hat{\theta}$ is an unbiased estimator of a parameter θ if $E[\hat{\theta}] = \theta$ or $E[\hat{\theta}] = E[\theta]$.

Theorem 3.4. *Let the estimator $\hat{x}_{k+1|k+1} = K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}z_{k+1}$ of the state vector x_{k+1} be defined for the state-space system described by equations 2.1 and 2.2. If $K'_{k+1} = I - K_{k+1}H_{k+1}$, then $\hat{x}_{k+1|k+1}$ is an unbiased estimator of the state vector x_{k+1}.*

Proof. Proof by induction:
Let this proof be governed by the state-space model described in equations 2.1 and 2.2.
Let P(n), n=-1,0,1,2,... be the statement that $\hat{x}_{n+1|n+1} = K'_n\hat{x}_{n+1|n} + K_{n+1}z_{n+1}$ is an unbiased estimator of $x_{n+1} \implies E[\hat{x}_{n+1|n+1}] = E[x_{n+1}]$.
Base case: P(-1)
By our initial conditions for our state-space model, $\hat{x}_{0|0} = E[x_0]$
$\implies E[\hat{x}_{0|0}] = E[x_0]$ Expected Value of Both Sides
$\implies \hat{x}_{n+1|n+1}$ is an unbiased estimator of x_{n+1} for $n = -1$.
Inductive Hypothesis:
Assume $\hat{x}_{k|k}$ is an unbiased estimator of x_k for some $k > -1$
$\implies E[\hat{x}_{k|k}] = E[x_k]$
Then...
$E[\hat{x}_{k+1|k+1}]$
$= E[K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}z_{k+1}]$ (Equation 3.2)
$= E[K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}(H_{k+1}x_{k+1} + v_{k+1})]$ (Equation 2.2)
$= E[K'_{k+1}\hat{x}_{k+1|k} + K_{k+1}H_{k+1}x_{k+1} + K_{k+1}v_{k+1}]$ (Expand)
$= E[K'_{k+1}\hat{x}_{k+1|k}] + E[K_{k+1}H_{k+1}x_{k+1}] + E[K_{k+1}v_{k+1}]$ (Linearity of Expected Value)
$= K'_{k+1}E[\hat{x}_{k+1|k}] + K_{k+1}H_{k+1}E[x_{k+1}] + K_{k+1}E[v_{k+1}]$ (Pull Out Constants Matrices from Expected Value)
$= K'_{k+1}E[\hat{x}_{k+1|k}] + K_{k+1}H_{k+1}E[x_{k+1}]$ (v_{k+1} has zero mean $\forall\ k$)
$= K'_{k+1}E[F_k\hat{x}_{k|k} + G_k u_k] + K_{k+1}H_{k+1}E[x_{k+1}]$ (Equation 3.1)
$= K'_{k+1}(E[F_k\hat{x}_{k|k}] + E[G_k u_k]) + K_{k+1}H_{k+1}E[x_{k+1}]$ (Linearity of Expected Value)
$= K'_{k+1}(F_k E[\hat{x}_{k|k}] + G_k u_k E[1]) + K_{k+1}H_{k+1}E[x_{k+1}]$ (Pull Out Constants from Expected Value)
$= K'_{k+1}(F_k E[x_k] + G_k u_k) + K_{k+1}H_{k+1}E[x_{k+1}]$ (Inductive Hypothesis)
$= K'_{k+1}E[x_{k+1}] + K_{k+1}H_{k+1}E[x_{k+1}]$ (See Below)

(1) $F_k E[x_k] + G_k u_k$
(2) $= F_k E[x_k] + G_k u_k + E[w_k]$ (w_k has zero mean $\forall\ k$)
(3) $= E[F_k x_k] + E[G_k u_k] + E[w_k]$ (F_k, G_k, u_k are constant vectors and matrices)
(4) $= E[F_k x_k + G_k u_k + w_k]$ (Linearity of Expected Value)
(5) $= E[x_{k+1}]$ (Equation 2.1)

$= (K'_{k+1} + K_{k+1}H_{k+1})E[x_{k+1}]$ (Simplify)
Thus, we have shown $E[\hat{x}_{k+1|k+1}] = (K'_{k+1} + K_{k+1}H_{k+1})E[x_{k+1}]$.
We need $E[\hat{x}_{k+1|k+1}] = E[x_{k+1}]$. We can see that this happens only when $K'_{k+1} + K_{k+1}H_{k+1} = I \implies K'_{k+1} = I - K_{k+1}H_{k+1}$.
Hence, we claim by induction that if $K'_{n+1} = I - K_{n+1}H_{n+1}$, then $\hat{x}_{n+1|n+1}$ is an **unbiased** estimator of the state vector x_{n+1} $\forall\ n = -1, 0, 1, ...$

□

By the previous theorem, our state estimate at time $t = k + 1$ given the observations $z_1, z_2, ..., z_{k+1}$ is unbiased when it is given by the equation:

(3.3) $$\boldsymbol{\hat{x}_{k+1|k+1} = (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} + K_{k+1}z_{k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - H_{k+1}\hat{x}_{k+1|k})}$$

Using this fact, we can now show that the estimator $\hat{x}_{k+1|k}$ is also unbiased.

Theorem 3.5. *Let the estimator $\hat{x}_{k+1|k}$ of the state vector x_{k+1} be defined for the state-space system described by equations 2.1 and 2.2. If $\hat{x}_{k|k}$ is an unbiased estimator of the state vector x_k, then $\hat{x}_{k+1|k}$ is an unbiased estimator of the state vector x_{k+1}*

Proof. .
$E[\hat{x}_{k+1|k}]$
$= E[F_k\hat{x}_{k|k} + G_k u_k]$ (Equation 2.1)
$= E[F_k\hat{x}_{k|k}] + E[G_k u_k]$ (Linearity of Expected Value)
$= F_k E[\hat{x}_{k|k}] + E[G_k u_k]$ (Pull out Constant Matrix F_k)
$= F_k E[x_k] + E[G_k u_k]$ ($\hat{x}_{k|k}$ is an Unbiased Estimator)
$= E[F_k x_k] + E[G_k u_k]$ (Pull in Csonstant Matrix F_k)
$= E[F_k x_k + G_k u_k]$ (Linearity of Expected Value)
$= E[F_k x_k + G_k u_k] + E[w_k]$ (w_k is zero mean)
$= E[F_k x_k + G_k u_k + w_k]$ (Linearity of Expected Value)
$= E[x_{k+1}]$ (Equation 2.1)
Thus, $E[\hat{x}_{k+1|k}] = E[x_{k+1}]$, so $\hat{x}_{k+1|k}$ is an unbiased estimator of x_{k+1}. □

Thus, we have shown our two intertwined estimators $\hat{x}_{k+1|k}$ and $\hat{x}_{k+1|k+1}$ are both unbiased estimators of the state x_{k+1}. Furthermore, we have shown $\hat{x}_{k+1|k}$ is MMSE. What remains to be shown is that $\hat{x}_{k+1|k+1}$ is a MMSE estimator. Next, we proceed by introducing the error covariance matrix. This will help us show that the estimator $\hat{x}_{k+1|k+1}$ is also a MMSE.

3.4. **1st Recursive Formula for Error Covariance.**

Definition 3.6. The error covariance for a parameter x with estimator $\hat{x}$ is given by in [6] as:

$$E[(x - \hat{x})(x - \hat{x})^T]$$

Definition 3.7. Let $X_n = (x_1, x_2, ..., x_n)$ be a vector of random variables and let $Y_n = (y_1, y_2, ..., y_n)$ be another vector of random variables. Then, the covariance of the two random vectors is defined by:

$$E([X_n - E[X_n])(Y_n - E[Y_n])^T]$$

Theorem 3.8. *The system states $x_0, x_1, ..., x_k$ are uncorrelated to the noise terms w_k,v_k $\implies$ $cov(x_n, w_k) = E[x_n w_k^T] = 0$ and $cov(x_n, v_k) = E[x_n v_k^T] = 0 \quad \forall \quad n \in [0, k]$*

Proof. Proof by Strong Induction:
We will prove the statement for the noise term w_k. Once this is proven the proof for the other noise term v_k is trivial. Let this proof be governed by the state-space model described in equations 2.1 and 2.2.

Let P(n), n=0,1,2,...,k be the statement that $cov(x_n, w_k) = E[x_n w_k^T] = 0$

Base case: P(0)
By the definition of covariance, $cov(x_0, w_k) = E[(x_0 - E[x_0])(w_k - E[w_k])^T]$
$cov(x_0, w_k) = E[(x_0 - E[x_0])w_k^T]$ ($E[w_k] = 0$, Kalman Filter Assumptions)
$cov(x_0, w_k) = E[x_0 w_k^T - E[x_0]w_k^T]$ (Multiply Through)
$cov(x_0, w_k) = E[x_0 w_k^T] - E[E[x_0]w_k^T]$ (Linearity of Expected Value)
$cov(x_0, w_k) = E[x_0 w_k^T] - E[x_0]E[w_k^T]$ (Pull out Constant Vector $E[x_0]$)
$cov(x_0, w_k) = 0 - E[x_0](0) = 0$ (Kalman Filter Assumptions, $E[x_0 w_k^T] = 0$, $E[w_k^T] = 0$)
Inductive Hypothesis: Assume $cov(x_n, w_k) = E[x_n w_k^T] = 0 \quad \forall \quad n \in [0, k-1]$
Then...
$cov(x_k, w_k)$
$= E[(x_k - E[x_k])(w_k - E[w_k])^T]$ (Definition of Covariance)
$= E[(x_k - E[x_k])(w_k)^T]$ (Kalman Filter Assumption, $E[w_k] = 0$)
$= E[x_k w_k^T - E[x_k]w_k^T]$ (Multiply Through)
$= E[x_k w_k^T] - E[E[x_k]w_k^T]$ (Linearity of Expected Value)
$= E[x_k w_k^T] - E[x_k]E[w_k^T]$ (Pull out constant $E[x_k]$)
$= E[x_k w_k^T]$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
$= E[(F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1})w_k^T]$ (Equation 2.1)
$= E[F_{k-1}x_{k-1}w_k^T + G_{k-1}u_{k-1}w_k^T + w_{k-1}w_k^T]$ (Expand)
$= E[F_{k-1}x_{k-1}w_k^T] + E[G_{k-1}u_{k-1}w_k^T] + E[w_{k-1}w_k^T]$ (Linearity of Expected Value)
$= F_{k-1}E[x_{k-1}w_k^T] + G_{k-1}u_{k-1}E[w_k^T] + E[w_{k-1}w_k^T]$ (Pull out Constant Matrices and Vectors $F_{k-1}, G_{k-1}, u_{k-1}$)
$= F_{k-1}E[x_{k-1}w_k^T] + G_{k-1}u_{k-1}(0) + (0)$ (Kalman Filter Assumptions, $E[w_k^T] = 0$ and $E[w_{k-1}w_k^T] = 0$)
$= F_{k-1}E[x_{k-1}w_k^T]$
$= F_{k-1}(0) = 0$ (Inductive Hypothesis)
Hence, $cov(x_k, w_k) = E[x_k w_k^T] = 0$
Thus, we claim by induction that the system state x_n is uncorrelated to noise term $w_k \implies cov(x_n, w_k) = E[x_n w_k^T] = 0 \; \forall \; n \in [0, k]$

□

Theorem 3.9. *The system state estimates $\hat{x}_{0|0}, \hat{x}_{1|1}, ..., \hat{x}_{k|k}$ are uncorrelated to the noise term $w_k \implies cov(\hat{x}_{n|n}, w_k) = E[\hat{x}_{n|n} w_k^T] = 0 \quad \forall \quad n \in [0, k]$.*

Proof. Proof by Strong Induction:
Let this proof be governed by the state-space model described in equations 2.1 and 2.2.
Let P(n), n=0,1,2,...,k be the statement that $cov(\hat{x}_{n|n}, w_k) = E[\hat{x}_{n|n} w_k^T] = 0$

Base case: P(0)
By the definition of covariance, $cov(\hat{x}_{0|0}, w_k) = E[(\hat{x}_{0|0} - E[\hat{x}_{0|0}])(w_k - E[w_k])^T]$
$cov(\hat{x}_{0|0}, w_k) = E[(\hat{x}_{0|0} - E[\hat{x}_{0|0}])w_k^T]$ ($E[w_k] = 0$, Kalman Filter Assumption)
$cov(\hat{x}_{0|0}, w_k) = E[\hat{x}_{0|0} w_k^T - E[\hat{x}_{0|0}]w_k^T]$ (Multiply Through)
$cov(\hat{x}_{0|0}, w_k) = E[\hat{x}_{0|0} w_k^T] - E[E[\hat{x}_{0|0}]w_k^T]$ (Linearity of Expected Value)
$cov(\hat{x}_{0|0}, w_k) = E[\hat{x}_{0|0} w_k^T] - E[\hat{x}_{0|0}]E[w_k^T]$ (Pull out Constant Vector $E[\hat{x}_{0|0}]$)
$cov(\hat{x}_{0|0}, w_k) = E[\hat{x}_{0|0} w_k^T]$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
$cov(\hat{x}_{0|0}, w_k) = E[E[x_0]w_k^T]$ (Initial Guess: $\hat{x}_{0|0} = E[x_0]$)
$cov(\hat{x}_{0|0}, w_k) = E[x_0]E[w_k^T]$ (Pull out Constant Vector $E[x_0]$)
$cov(\hat{x}_{0|0}, w_k) = 0$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
Inductive Hypothesis: Assume $cov(\hat{x}_{n|n}, w_k) = E[\hat{x}_{n|n} w_k^T] = 0 \quad \forall \quad n \in [0, k-1]$
Then...
$cov(\hat{x}_{k|k}, w_k)$
$= E[(\hat{x}_{k|k} - E[\hat{x}_{k|k}])(w_k - E[w_k])^T]$ (Definition of Covariance)
$= E[(\hat{x}_{k|k} - E[\hat{x}_{k|k}])(w_k)^T]$ (Kalman Filter Assumption, $E[w_k] = 0$)
$= E[\hat{x}_{k|k} w_k^T - E[\hat{x}_{k|k}]w_k^T]$ (Multiply Through)
$= E[\hat{x}_{k|k} w_k^T] - E[E[\hat{x}_{k|k}]w_k^T]$ (Linearity of Expected Value)

$= E[\hat{x}_{k|k}w_k^T] - E[\hat{x}_{k|k}]E[w_k^T]$ (Pull out Constant Vector $E[\hat{x}_{k|k}]$)
$= E[\hat{x}_{k|k}w_k^T]$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
$= E[((I - K_kH_k)\hat{x}_{k|k-1} + K_kz_k)w_k^T]$ (Equation 3.3)
$= E[(I - K_kH_k)\hat{x}_{k|k-1}w_k^T + K_kz_kw_k^T]$ (Multiply Through)
$= E[(I - K_kH_k)\hat{x}_{k|k-1}w_k^T] + E[K_kz_kw_k^T]$ (Linearity of Expected Value)
$= (I - K_kH_k)E[\hat{x}_{k|k-1}w_k^T] + K_kz_kE[w_k^T]$ (Pull out Constant Matrices $(I - K_kH_k)$,K_k and Known Vector z_k)
$= (I - K_kH_k)E[\hat{x}_{k|k-1}w_k^T]$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
$= (I - K_kH_k)E[(F_{k-1}\hat{x}_{k-1|k-1} + G_{k-1}u_{k-1})w_k^T]$ (Equation 3.1)
$= (I - K_kH_k)E[F_{k-1}\hat{x}_{k-1|k-1}w_k^T + G_{k-1}u_{k-1}w_k^T]$ (Multiply Through)
$= (I - K_kH_k)E[F_{k-1}\hat{x}_{k-1|k-1}w_k^T] + E[G_{k-1}u_{k-1}w_k^T]$ (Linearity of Expected Value)
$= (I - K_kH_k)F_{k-1}E[\hat{x}_{k-1|k-1}w_k^T] + G_{k-1}u_{k-1}E[w_k^T]$ (Pull out Constant Matrices and Vectors F_{k-1},G_{k-1}, and u_{k-1})
$= (I - K_kH_k)F_{k-1}E[\hat{x}_{k-1|k-1}w_k^T]$ (Kalman Filter Assumption, $E[w_k^T] = 0$)
$= (I - K_kH_k)F_{k-1}(0) = 0$ (Inductive Hypothesis)
Thus, we claim by induction that the system state estimate $\hat{x}_{n|n}$ is uncorrelated to noise term $w_k \implies cov(\hat{x}_{n|n}, w_k) = E[\hat{x}_{n|n}w_k^T] = 0 \; \forall \; n \in [0, k]$ □

Theorem 3.10. *The system state estimates* $\hat{x}_{1|0}, \hat{x}_{2|1}, ..., \hat{x}_{k-1|k}$ *are uncorrelated to the noise term* v_k, $\implies cov(\hat{x}_{n|n-1}, v_k) = E[\hat{x}_{n|n-1}v_k^T] = 0 \quad \forall \quad n \in [0, k]$.

Proof. The proof is similar to that above. First prove the base case, make an inductive assumption, and then prove $cov(\hat{x}_{k|k-1}, v_k) = E[\hat{x}_{k|k-1}v_k^T] = 0$. This proof will again involve Equations 3.1 and 3.3, but will be used in a different order than the last proof. □

Below we derive a formula for the error covariance matrix for the state x_{k+1} which is estimated by $\hat{x}_{k+1|k}$ as in [6].

$$
\begin{aligned}
P_{k+1|k} &= E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T] \\
&= E[((F_k x_k + G_k u_k + w_k) - (F_k \hat{x}_{k|k} + G_k u_k))((F_k x_k + G_k u_k + w_k) - (F_k \hat{x}_{k|k} + G_k u_k))^T] \\
&\quad \texttt{Equation 2.1 and 3.1} \\
&= E[(F_k(x_k - \hat{x}_{k|k}) + w_k)(F_k(x_k - \hat{x}_{k|k}) + w_k)^T] \\
&\quad \texttt{Simplify} \\
&= E[(F_k(x_k - \hat{x}_{k|k}) + w_k)((F_k(x_k - \hat{x}_{k|k}))^T + w_k^T)] \\
&\quad \texttt{Property:(A+B)}^T\texttt{ = A}^T\texttt{ + B}^T \\
&= E[(F_k(x_k - \hat{x}_{k|k}) + w_k)((x_k - \hat{x}_{k|k})^T F_k^T + w_k^T)] \\
&\quad \texttt{Property:(AB)}^T\texttt{ = B}^T\texttt{A}^T \\
&= E[F_k(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T F_k^T + F_k(x_k - \hat{x}_{k|k})w_k^T + w_k(x_k - \hat{x}_{k|k})^T F_k^T + w_k w_k^T] \\
&\quad \texttt{Expand} \\
&= E[F_k(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T F_k^T] + E[F_k(x_k - \hat{x}_{k|k})w_k^T] \\
&\quad + E[w_k(x_k - \hat{x}_{k|k})^T F_k^T] + E[w_k w_k^T] \\
&\quad \texttt{Linearity of Expected Value} \\
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + F_k E[(x_k - \hat{x}_{k|k})w_k^T] \\
&\quad + E[w_k(x_k - \hat{x}_{k|k})^T] F_k^T + E[w_k w_k^T] \\
&\quad \texttt{Pull out constant Matrices from Expected Value} \\
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + F_k E[(x_k - \hat{x}_{k|k})w_k^T] \\
&\quad + E[w_k(x_k^T - \hat{x}_{k|k}^T)] F_k^T + E[w_k w_k^T] \\
&\quad \texttt{Property:(A+B)}^T\texttt{ = A}^T\texttt{ + B}^T
\end{aligned}
$$

$$
\begin{aligned}
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + F_k E[x_k w_k^T - \hat{x}_{k|k} w_k^T] \\
&\quad + E[w_k x_k^T - w_k \hat{x}_{k|k}^T] F_k^T + E[w_k w_k^T] \\
&\texttt{Expand Middle Terms} \\
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + F_k(E[x_k w_k^T] - E[\hat{x}_{k|k} w_k^T]) \\
&\quad + (E[w_k x_k^T] - E[w_k \hat{x}_{k|k}^T]) F_k^T + E[w_k w_k^T] \\
&\texttt{Linearity of Expected Value} \\
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + E[w_k w_k^T]
\end{aligned}
$$

`Theorems 3.8, 3.9: The states` x_k `and state estimates` $\hat{x}_{k|k}$ `are uncorrelated with the noise` $w_k \implies E[x_k w_k^T] = 0, E[\hat{x}_{k|k} w_k^T] = 0, E[w_k x_k^T] = 0, E[w_k \hat{x}_{k|k}^T] = 0$

$$
\begin{aligned}
&= F_k E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T] F_k^T + Q_k \\
&= F_k P_{k|k} F_k^T + Q_k
\end{aligned}
$$

$$\boldsymbol{P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k} \tag{3.4}$$

Thus, we have a formula for the error covariance matrix for the estimator $\hat{x}_{k+1|k}$. Next, we derive a formula for our error covariance matrix for the estimator $\hat{x}_{k+1|k+1}$ as in [6].

3.5. **2nd Recursive Formula for Estimated Error Covariance.**

$$
\begin{aligned}
P_{k+1|k+1} &= E[(x_{k+1} - \hat{x}_{k+1|k+1})(x_{k+1} - \hat{x}_{k+1|k+1})^T] \\
&= E[(x_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} \\
&\quad - K_{k+1}z_{k+1})(x_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} - K_{k+1}z_{k+1})^T] \\
&\quad \texttt{Equation 3.3} \\
&= E[(x_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} - K_{k+1}(H_{k+1}x_{k+1} + v_{k+1}))(x_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} \\
&\quad - K_{k+1}(H_{k+1}x_{k+1} + v_{k+1}))^T] \\
&\quad \texttt{Equation 2.2} \\
&= E[(Ix_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} \\
&\quad - K_{k+1}(H_{k+1}x_{k+1} + v_{k+1}))(Ix_{k+1} - (I - K_{k+1}H_{k+1})\hat{x}_{k+1|k} - K_{k+1}(H_{k+1}x_{k+1} + v_{k+1}))^T] \\
&\quad \texttt{Note:} \quad Ix_{k+1} = x_{k+1} \\
&= E[((I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k}) - K_{k+1}v_{k+1})((I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k}) \\
&\quad - K_{k+1}v_{k+1})^T] \\
&\quad \texttt{Simplify} \\
&= E[((I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k}) - K_{k+1}v_{k+1})(((I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k}))^T \\
&\quad - (K_{k+1}v_{k+1})^T)] \\
&\quad \texttt{Property:(A+B)}^T \texttt{ = A}^T\texttt{+B}^T \\
&= E[((I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k}) - K_{k+1}v_{k+1})((x_{k+1} - \hat{x}_{k+1|k})^T(I - K_{k+1}H_{k+1})^T \\
&\quad - v_{k+1}^T K_{k+1}^T)] \\
&\quad \texttt{Property:(AB)}^T \texttt{ = B}^T A^T \\
&= E[(I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T(I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k})(v_{k+1}^T K_{k+1}^T) \\
&\quad - (K_{k+1}v_{k+1})(x_{k+1} - \hat{x}_{k+1|k})^T(I - K_{k+1}H_{k+1})^T + K_{k+1}v_{k+1}v_{k+1}^T K_{k+1}^T] \\
&\quad \texttt{Expand}
\end{aligned}
$$

$$
\begin{aligned}
&= E[(I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T(I - K_{k+1}H_{k+1})^T] \\
&\quad - E[(I - K_{k+1}H_{k+1})(x_{k+1} - \hat{x}_{k+1|k})(v_{k+1}^T K_{k+1}^T)] \\
&\quad - E[(K_{k+1}v_{k+1})(x_{k+1} - \hat{x}_{k+1|k})^T(I - K_{k+1}H_{k+1})^T] + E[K_{k+1}v_{k+1}v_{k+1}^T K_{k+1}^T]
\end{aligned}
$$

`Linearity of Expected Value`

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})v_{k+1}^T]K_{k+1}^T \\
&\quad - K_{k+1}E[v_{k+1}(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Pull Out Constant Matrices`

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})v_{k+1}^T]K_{k+1}^T \\
&\quad - K_{k+1}E[v_{k+1}(x_{k+1}^T - \hat{x}_{k+1|k}^T)](I - K_{k+1}H_{k+1})^T + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Property` $(A+B)^T = A^T + B^T$

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})E[x_{k+1}v_{k+1}^T - \hat{x}_{k+1|k}v_{k+1}^T]K_{k+1}^T \\
&\quad - K_{k+1}E[v_{k+1}x_{k+1}^T - v_{k+1}\hat{x}_{k+1|k}^T](I - K_{k+1}H_{k+1})^T + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Expand Two Middle Expected Values`

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})(E[x_{k+1}v_{k+1}^T] - E[\hat{x}_{k+1|k}v_{k+1}^T])K_{k+1}^T \\
&\quad - K_{k+1}(E[v_{k+1}x_{k+1}^T] - E[v_{k+1}\hat{x}_{k+1|k}^T])(I - K_{k+1}H_{k+1})^T + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Linearity of Expected Value`

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad - (I - K_{k+1}H_{k+1})(0 - 0)K_{k+1}^T \\
&\quad - K_{k+1}(0 - 0)(I - K_{k+1}H_{k+1})^T + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Theorems 3.8, 3.10:  The states` x_{k+1} `and state estimates` $\hat{x}_{k+1|k}$ `are uncorrelated with the noise` $v_{k+1} \implies E[x_{k+1}v_{k+1}^T] = 0, E[\hat{x}_{k+1|k}v_{k+1}^T] = 0, E[v_{k+1}x_{k+1}^T] = 0, E[v_{k+1}\hat{x}_{k+1|k}^T] = 0$

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad + K_{k+1}E[v_{k+1}v_{k+1}^T]K_{k+1}^T
\end{aligned}
$$

`Simplify`

$$
\begin{aligned}
&= (I - K_{k+1}H_{k+1})E[(x_{k+1} - \hat{x}_{k+1|k})(x_{k+1} - \hat{x}_{k+1|k})^T](I - K_{k+1}H_{k+1})^T \\
&\quad + K_{k+1}R_{k+1}K_{k+1}^T \\
&= (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}K_{k+1}^T
\end{aligned}
$$

Thus, our formula for the error covariance matrix of the estimator $\hat{x}_{k+1|k+1}$ is:

$$
\boldsymbol{P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}K_{k+1}^T} \tag{3.5}
$$

3.6. **Finding the Minimum Variance Estimator of the State.** Remember, we want to minimize the mean squared error $E[||\hat{x} - x||^2]$. Notice that

$$
trace(P_{k+1|k+1}) = E[||\hat{x}_{k+1|k+1} - x_{k+1}||^2] \text{ as given by [6]}
$$

Our goal becomes now to find the K_{k+1} which minimizes $trace(P_{k+1|k+1})$ and thus minimizes the mean squared error as they do in [6]. We first need to introduce a some terminology from [7].

Definition 3.11. Let f be a scalar and $A \in \mathbb{R}^{mxn}$ be a matrix. Then, the derivative of the scalar f with respect to the matrix A is $\frac{\partial f}{\partial A} = \begin{bmatrix} \partial f/\partial A_{11} & \partial f/\partial A_{12} & \dots & \partial f/\partial A_{1n} \\ \partial f/\partial A_{21} & \partial f/\partial A_{22} & \dots & \partial f/\partial A_{2n} \\ \vdots & \vdots & \ddots & \\ \partial f/\partial A_{m1} & \partial f/\partial A_{m2} & \dots & \partial f/\partial A_{mn} \end{bmatrix}$

Theorem 3.12. *If A is any matrix and B is a symmetric matrix, then* $\frac{\partial}{\partial A}(trace(ABA^T)) = 2AB$

Proof. Let A be a nxp matrix, and let B be a pxp symmetric matrix.
$f = trace(ABA^T)$
$= trace(AB^TA^T)$ (B is Symmetric)
$= trace(A(AB)^T)$ (Property $(AB^T) = B^TA^T$)
$= \sum_{i=1}^{n}[A(AB)^T]_{ii}$
$= \sum_{i=1}^{n}\sum_{j=1}^{p}[A_{ij}(AB)^T_{ji}]$
$= \sum_{i=1}^{n}\sum_{j=1}^{p}[A_{ij}(AB)_{ij}]$
$= \sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{p}[A_{ij}A_{ik}B_{kj}]$
Thus, we have $f = \sum_{i=1}^{n}\sum_{j=1}^{p}\sum_{k=1}^{p}[A_{ij}A_{ik}B_{kj}]$. Let the entries in the matrix $A^* = \frac{\partial f}{\partial A}$ be denoted by $(A^*)_{ij}$ for i, j fixed. Then,
$(A^*)_{ij} = \frac{\partial f}{\partial A_{ij}}A_{ij}\sum_{k=1}^{p}[A_{ik}B_{kj}] + A_{ij}\sum_{k=1}^{p}\frac{\partial f}{\partial A_{ij}}[A_{ik}B_{kj}]$
(Product Rule).
We can see that
$\frac{\partial f}{\partial A_{ij}}A_{ij}\sum_{k=1}^{p}[A_{ik}B_{kj}] = \sum_{k=1}^{p}[A_{ik}B_{kj}]$
and $A_{ij}\sum_{k=1}^{p}[\frac{\partial f}{\partial A_{ij}}(A_{ik}B_{kj})] = 0, k \neq j$
and $A_{ij}\sum_{k=1}^{p}[\frac{\partial f}{\partial A_{ij}}(A_{ik}B_{kj})] = A_{ij}\sum_{k=1}^{p}B_{kj}, k = j$
$= \sum_{k=1}^{p}A_{ik}B_{kj}, k = j$
Thus, $A^*_{ij} = 2\sum_{k=1}^{p}A_{ik}B_{kj} \implies$
$A^* = \frac{\partial f}{\partial A} = 2AB$

□

We now derive a condition that guarantees $\hat{x}_{k+1|k+1}$ is a MMSE as they do in [6].

$$\begin{aligned} f = trace(P_{k+1|k+1}) &= trace((I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}K_{k+1}^T) \\ &\quad \texttt{Equation 3.5} \\ &= trace((I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T) + trace(K_{k+1}R_{k+1}K_{k+1}^T) \\ &\quad \texttt{Linearity of Trace} \\ \frac{\partial}{\partial K_{k+1}} &= -2(I - K_{k+1}H_{k+1})P_{k+1|k}H_{k+1}^T + 2K_{k+1}R_{k+1} \\ &\quad \texttt{Chain Rule and Thm.3.12} \\ \frac{\partial}{\partial K_{k+1}} &= 0 \implies \\ 0 &= -2(I - K_{k+1}H_{k+1})P_{k+1|k}H_{k+1}^T + 2K_{k+1}R_{k+1} \implies \\ 0 &= -2IP_{k+1|k}H_{k+1}^T + 2K_{k+1}H_{k+1}P_{k+1|k}H_{k+1}^T + 2K_{k+1}R_{k+1} \implies \\ 0 &= -2IP_{k+1|k}H_{k+1}^T + 2K_{k+1}(H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1}) \implies \end{aligned}$$

f has an extremum when $K_{k+1} = P_{k+1|k}H_{k+1}^T(H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1})^{-1}$
Looking at the Hessian of f, we can verify the K_{k+1} indeed minimizes f.
Since f is equivalent to the mean-squared error, this value gives the minimum variance estimator. Thus, our equation for K_{k+1} which guarantees a MMSE:

$$\boldsymbol{K_{k+1} = P_{k+1|k}H_{k+1}^T(H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1})^{-1}} \tag{3.6}$$

Thus, we are done with our derivation. We will next summarize the equations we have.

4. Equations and Comments

Following the state-space model and assumptions given in section 2, we have:

- **Prediction Equations:**
  (1) $\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k$
    - This is an estimator for the state x_{k+1} based on the observations $z_1, z_2, ..., z_k$.
    - We have proven this estimator is unbiased and MMSE.
    - We can also see this estimator is linear in terms of $\hat{x}_{k|k}$.
    - Finally, this estimator is coupled with the other estimator $\hat{x}_{k+1|k+1}$

  (2) $P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k$
    - This is the error covariance matrix for the estimator $\hat{x}_{k+1|k}$.
    - It is coupled with the error covariance matrix for the estimator $\hat{x}_{k+1|k+1}$
- **Update Equations:**
  (1) $K_{k+1} = P_{k+1|k} H_{k+1}^T (H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1})^{-1}$
    - This term is known as the Kalman Gain.
    - It guarantees that the estimator $\hat{x}_{k+1|k+1}$ is a MMSE.

  (2) $\hat{x}_{k+1|k+1} = (I - K_{k+1} H_{k+1}) \hat{x}_{k+1|k} + K_{k+1} z_{k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - H_{k+1} \hat{x}_{k+1|k})$
    - This is an estimator for the state x_{k+1} based on the observations $z_1, z_2, ..., z_{k+1}$. Thus, it differs from the previous estimator in that we update the estimator with a new observation.
    - We have proven this estimator is unbiased and MMSE.
    - We can also see this estimator is linear in terms of $\hat{x}_{k+1|k}$.
    - Finally, this estimator is coupled with the other estimator $\hat{x}_{k+1|k}$

  (3) $P_{k+1|k+1} = (I - K_{k+1} H_{k+1}) P_{k+1|k} (I - K_{k+1} H_{k+1})^T + K_{k+1} R_{k+1} K_{k+1}^T$
    - This is the error covariance matrix for the estimator $\hat{x}_{k+1|k+1}$.
    - It is coupled with the error covariance matrix for the estimator $\hat{x}_{k+1|k}$.

5. Example 1: Simple Kalman Filter to filter out noise from readings of a noisy DC Voltmeter

In this example from [1], we assume we have a constant DC Voltage of 0.5 V, and we take noisy measurements $z_k, k = 1, 2, ..., 100$ with a noisy voltmeter. Below we list all terms in the model.

List of Terms:

- Let k denote the **time**
- x_k - State vectors
- $u_k = [0] \; \forall \; k$
- z_k - Represents noisy voltmeter readings at time k.
- $F_k = [1] \; \forall \; k$
- $G_k = [0] \; \forall \; k$
- $H_k = [1] \; \forall \; k$
- w_k - Process noise at time k. The noise is generated automatically using the MATLAB function *mvrnd* with the covariance matrix Q_k.
- v_k - Additive measurement noise at time k. The noise is generated automatically using the MATLAB function *mvrnd* with the covariance matrix R_k.
- $Q_k = [0.00001] \; \forall \; k$. This is an arbitrary choice. Analysis of this choice is beyond the scope of this paper.
- $R_k = [1] \; \forall \; k$. This is an arbitrary choice. Analysis of this choice is beyond the scope of this paper.
- Equation 2.1: $x_{k+1} = x_k + w_k$
- Equation 2.2: $z_k = x_k + v_k$
- This is a very simplified example. The next state differs from the previous state only with noise. Furthermore, the measurements only differ from the states due to a noise term.

We have **Initial Conditions**:

- $\hat{x}_{0|0} = 0.5$ - This is an arbitrary choice. Analysis of this choice is beyond the scope of this paper.
- $P_{0|0} = [1]$ - This is an arbitrary choice. Analysis of this choice is beyond the scope of this paper.

Notes:

- To generate simulated measurements: We simulate states x_k with noise according to our initial guess and Equation 2.1. Then, we proceed by simulating measurements z_k according to Equation 2.2.
- We introduce a term called a predicted measurement $\hat{z}_{k+1|k} = H_{k+1}\hat{x}_{k+1|k}$. This term is used to compare the simulated measurements to the predicted measurements generated by the Kalman Filter.

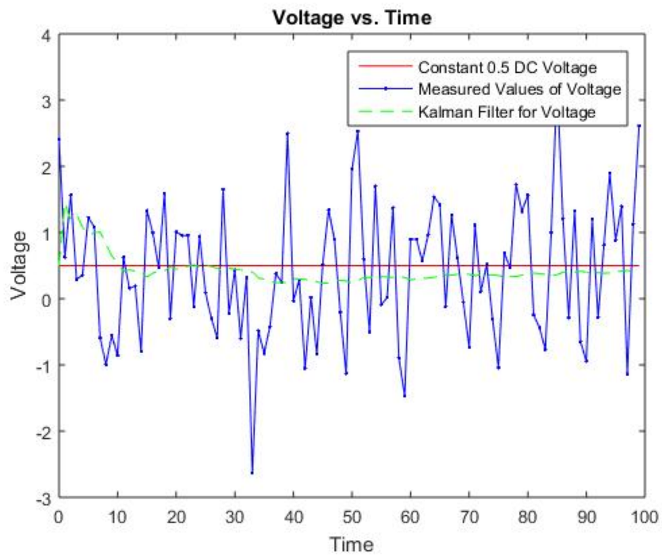

FIGURE 2. This plot corresponds to the first example of reading voltage from a noisy voltmeter. This graphs plots the simulated given measurements $z_1, ...z_{100}$, the predicted observations $\hat{z}_{1|0}, ..., \hat{z}_{101|100}$ from the Kalman Filter, and the constant 0.5 DC Voltage. Notice how the filter filters out the noise.

6. Example 2: Kalman Filter for projectile motion

In this example from [2], we will model projectile motion with drag b. Below we list all terms in the model and physics Equations:

- For our model, $x_0 = 0$, $y_0 = 0$,$v_{x,0} = 300$, $v_{y,0} = 600$, $a_x = 0$,$a_y = -9.8, \delta t = 0.1, b = 0.0001$
- Kinematic Equation for x position: $x_k = x_{k-1} + v_{x,k-1}\delta t$
- Kinematic Equation for y position: $y_k = y_{k-1} + v_{y,k-1}\delta t$
- Kinematic Equation for x velocity: $v_{x,k} = (1-b)v_{x,k-1}$
- Kinematic Equation for y velocity: $v_{y,k} = (1-b)v_{y,k-1} - a_y\delta t$

List of Terms:

- Let k denote the **time** where in the discrete case $k = 0, 0.1, 0.2, ..., 120$.
- x_k - Represents the state vector at time k where $x_k = \begin{bmatrix} x_k \\ y_k \\ v_{x,k} \\ v_{y,k} \end{bmatrix}$
- u_k - Represents the input vector at time k where $u_k = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -a_y\delta t \end{bmatrix}$
- z_k - Represents the noisy measurements of x position, y position $z_k = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$

- $F_k = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1-b & 0 \\ 0 & 0 & 0 & 1-b \end{bmatrix}$
- $G_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- $H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
- w_k - Process noise at time k. The noise is generated automatically using the MATLAB function $mvrnd$ with the covariance matrix Q_k.
- v_k - Additive measurement noise at time k. The noise is generated automatically using the MATLAB function $mvrnd$ with the covariance matrix R_k.
- $Q_k = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$. (Arbitrary)(Analysis of this choice is beyond the scope of this paper)
- $R_k = \begin{bmatrix} 500 & 0 \\ 0 & 500 \end{bmatrix}$. (Arbitrary)(Analysis of this choice is beyond the scope of this paper)

List of Terms:

- $P_{0|0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. (Arbitrary)(Analysis of this choice is beyond the scope of this paper)
- $E[x_{0|0}] = \begin{bmatrix} 0 & 0 & 300 & 600 \end{bmatrix}$. (Arbitrary)(Analysis of this choice is beyond the scope of this paper)

Notes

- Simulating measurements and true projectile motion: We simulate states x_k with noise according to our initial guess and Equation 2.1. Then, we proceed by simulating measurements z_k according to Equation 2.2. We also simulate ideal conditions (Equation 2.1, Equation 2.2 without the noise w_k and v_k)
- We introduce a term called a predicted measurement $\hat{z}_{k+1|k} = H_{k+1}\hat{x}_{k+1|k}$. This term is used to compare the simulated measurements to the predicted measurements generated by the Kalman Filter.

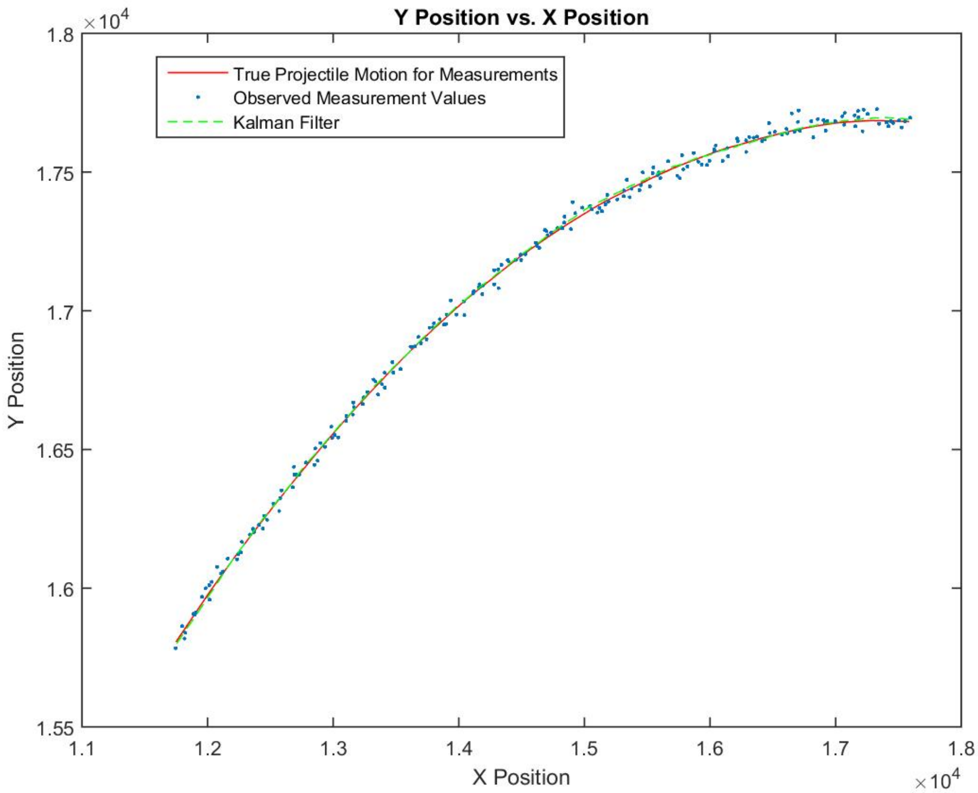

FIGURE 3. This plot corresponds to the projectile motion example. It plots the given observations, the predicted observations from the Kalman Filter, and the true projectile motion with no noise. Look how the Kalman Filter predicted observations are almost identical to the true projectile motion.

7. Conclusions

In this paper, we provided a more rigorous derivation of the discrete time Kalman Filter algorithm in full based on the derivation from [6]. We went through a step-by-step derivation to show that the algorithm produces a minimum variance, unbiased, recursive, linear estimate of the state of a noisy control system. We also provided two simple physics example in which the Kalman Filter algorithm was used, implementing these examples in MATLAB to demonstrate the utility of the Kalman Filter in practical applications. In this paper, we did not consider how to properly choose initial guesses and covariance terms, how to evaluate the performance of the filter, or any extension of the filter such as the extended Kalman Filter. This is something we look forward to doing in the future.

8. Matlab Code

```matlab

%Name: Alec Knutsen
%Date:12/08/15
%Description: Function that implements Kalman Filter Prediction Equations:
    %Predict State: xhat_k+1|k = (F_k)(xhat_k|k) + (G_k)(u_k)
    %Predict Error Covariance: P_k+1|k = (F_k)(P_k|k)(F_k)^T + Q_k

function [x_pred, p_pred] = predict(xhat,P,F_k,Q_k,G_k,u_k)


    x_pred = F_k*xhat + G_k*u_k;
    p_pred = F_k*P*F_k' + Q_k;

end
```

```matlab

%Name: Alec Knutsen
%Date:12/08/15
%Description: Function that implements Kalman Filter Update Equations, Kalman Gain, and Predicton of Observations:
        %Kalman Gain: K_k+1 = (P_k+1|k)(H_k+1)^T[(H_k+1)(P_k+1|k)(H_k+1|k)^T
        %                          +R_k+1]^-1
        %Predicted Observation: zhat_k+1|k = (H_k+1)(xhat_k+1|k)
        %Update State: xhat_k+1|k+1 = xhat_k+1|k + K_k+1(z_k+1 -
        %(H_k+1)(xhat_k+1|k))
        %Update Error Covariance: P_k+1|k+1 = (I - (K_k+1)(H_k+1))(P_k+1|k)(I - (K_k+1)(H_k+1))^T + (K_k+1)(R_k+1)(K_k+1)^T

function [x_update,p_update,K ,z_pred] = update(xhat, P,z_k, H_k, R_k)

    K = P*H_k'* inv(H_k*P*H_k' + R_k);

    z_pred = H_k*xhat;

    x_update = xhat + K*(z_k - H_k*xhat);

    d = size(K*H_k);
    needed_size = d(1);

    p_update = (eye(needed_size) - K*H_k)*P*(eye(needed_size) - K*H_k)' + K*R_k*K';



end
```

```
%Name: Alec Knutsen
%Date:12/08/15
%Description:This Program runs the Discrete Kalman Filter Algorithm for the
%voltage example described in the paper.
%
%
%State Space Model:
    %   Equation 1: x_k+1 = (F_k)(x_k)+ (G_k)(u_k) + w_k
    %   Equation 2: z_k = (H_k)(x_k) + v_k
    %
    % Parameters:
        % x_k - State vector at time k with deminsions nx1
        % u_k - Control input at time k with dimensions mx1
        % z_k - Observation at time k with dimensions px1
        % F_k - State transition matrix at time k with dimensions nxn
        % G_k - Input transition matrix at time k with dimensions nxm
        % H_k - Observation matrix at time k with dimensions pxn
        % w_k - Process noise at time k with dimensions nx1
        % v_k - Additive noise measurement at time k with dimension px1
        % Q_k - Covariance matrix (Q= E[(w_k)(w_k)^T]) at time k with
        % dimensions nxn
        %R_k - Covariance matrix (R_k= E[(v_k)(v_k)^T]) at time k with
        % dimensions pxp
        %P_k - Covariance error matrix(P_k = E[(xhat_k - x_k)(xhat_k-x_k)^T])

%We have the following Kalman Filter Equations:
    %Initial Conditions:
        %xhat_0 = E[x_0]
        %P_0|0 = E[(xhat_0 - x_0)(xhat_0-x_0)^T]
    %Prediction Equations:
        %Equation 3: xhat_k+1|k = (F_k)(xhat_k|k) + (G_k)(u_k)
        %Equation 4: P_k+1|k = (F_k)(P_k|k)(F_k)^T + Q_k
    %Update Equations:
        %Equation 5: xhat_k+1|k+1 = xhat_k+1|k + K_k+1(z_k+1 - (H_k+1))(xhat_k+1|k)
        %Equation 6: P_k+1|k+1 = (I - (K_k+1)(H_k+1))(P_k+1|k)(I - (K_k+1)(H_k+1))^T + (K_k+1)(R_k+1)(K_k+1)^T

% Extra Equations:
    % Kalman Gain: K_k+1 = (P_k+1|k)(H_k+1)^T[(H_k)(P_k+1|k)(H_k)^T +
    %                        R_k+1]^-1



%Beginning of Program:
%Note: Everything in this example is one dimensional. When, comments refer
%to vectors or matrices, everything is a scalar.

%User input: n - Size of state vectors
```

```
n=1;

%User input: m - Size of input vectors
m=1;

%User input: p - The size of observation vectors
p=1;

%time - Discrete time variable
time=1;

%User input: num_est - Total number of states you want to estimate. In this
%voltage example, it estimates 100 states
num_est=100;

%xhat - Cell array that will store each estimated state vector (xhat_k|k)
% Each  estimated state vector is size n x 1
xhat = cell(num_est,1);
%Initialize each estimated state vector to be zeros. These will be replaced
    with
%estimates
for  i =1:num_est
    xhat{i,1} = zeros(n,1);

end

%u - Cell array that will store each input vector (u_k)
%Each input vector is size mx1
u = cell(num_est,1);
%User input: See the paper for proper initialization
for  i =1:num_est
    u{i,1} = [0];

end


%F - Cell array that stores each F (state transistion) matrix
%Each F matrix is size nxn
%User input: See the paper for proper initialization
F = cell(num_est,1);
for  i =1:num_est
    F{i,1} = [1];

end


%G - Cell array that stores each G (input transition) matrix
%Each G matrix is size nxm
G = cell(num_est,1);
%User input: See the paper for proper initialization
```

```
for i =1:num_est
    G{i,1} = [0];

end

%H - Cell array that stores each H matrix
%Each H matrix is size pxn
H = cell(num_est,1);
%User input: See the paper for proper initialization
for i =1:num_est
    H{i,1} = [1];

end

%Q - Cell array that stores each covariance matrix for the noise w_k
%The size of each Q is nxn
Q = cell(num_est,1);
%User input: See the paper for proper initialization
for i =1:num_est
    Q{i,1} = [0.0001];


end

%R - Cell array that stores each covariance matrix for the noise v_k
%The size of each R is pxp
R = cell(num_est,1);
%User input: See the paper for proper initialization
for i =1:num_est
    R{i,1} = [1];

end

%W - Matrix that will store each noise vector
%The noise vectors are of size nx1
W = cell(num_est,1);

%Note the noise is automatically generated based on the covariance matrix
%using mvrnd
for i =1:num_est
    W{i,1} = (mvnrnd(zeros(1,n),Q{1,1}))';

end

%V - Matrix that will store each noise vector
%The v vectors are size px1
V = cell(num_est,1);
%Note the noise is automatically generated based on the covariance matrix
%using mvrnd
for i =1:num_est
```

```
    V{i,1} = (mvnrnd(zeros(1,p),R{1,1}))';

end

%P - Covariance matrix of errors (difference between estimated state and
    actual state)
%Each p matrix is size nxn
P = cell(num_est,1);
%We initially fill all covariance matrices to be a matrix of zeros
for  i =1:num_est
    P{i,1} = zeros(n,1);

end


%z_predicted stores each predicted observation (zhat_k+1|k)
%The size of each element of z_predicted is px1
z_predicted = cell(num_est,1);
for  i =1:num_est
    %We initially fill all z predicted vectors to be vectors of zeros
     z_predicted{i,1} = zeros(p,1);
end


%x - Cell array that will store each state vector (x_k). We will generate
    state
%vectors based on an initial value of data and using equation 1
% Each state vector is size n x 1
x = cell(num_est,1);

%z - Matrix that will store each observation vector (z_k) in its columns
%Each observation vector is px1
z = cell(num_est,1);

%User Input: P_0 - Initial Covariance Matrix.
%See the paper for proper initialization.
P_0 = [1];

%Store initial value in P_cell
p_update = P_0;

%We make an initial observation of 0.5 DC Voltage.
x_1 = [0.5];

%Store initial value for Kalman Filter
x{1,1} = x_1;
x_update = x_1;

%We generate the first observation vector
z{1,1} = H{1,1}*x{1,1} + V{1,1};
```

```

%We generate state vectors based on the initial observation and equation 1
%We generate observation vectors based on equation 2
for  i = 2:num_est
   x{i,1} = F{i-1,1}*x{i-1,1} + G{i-1,1}*u{i-1,1} + W{i-1,1};
   z{i,1} = H{i,1}*x{i,1} + V{i,1};

end


while(time<=num_est);


    %Store updated estimates for state(xhat_k+1|k+1) and error covariance (Phat_k+1|k+1)
    xhat{time,1} = x_update;
    P{time,1} = p_update;

    % Implements prediction equations 3 and 4
    [x_pred, p_pred] = predict(x_update,p_update, F{time,1}, Q{time,1},G{time,1},u{time,1});


    % Implements update equations 5 and 6
    [x_update,p_update, K,z_predictions] = update(x_pred, p_pred, z{time,1},H{time,1}, R{time,1});

    %Store z_predicted (zhat_k+1|k)
    z_predicted{time,1}= z_predictions;

    time=time+1;

end;

%Store time for plotting purposes
time_vec = zeros(1,num_est);


%This vector will hold a constant DC Voltage of 0.5V
ideal_voltage = zeros(1,num_est);

%This vector will hold the actual voltage measurements (z_k)
measurements_with_noise = zeros(1,num_est);

%This vector will hold the predicted observations based on The Kalman Filter (zhat_k+1)
predicted_observations = zeros(1,num_est);

%This stores the elements of the cell array elements into the the above
%vectors
```

```
for i=0:num_est-1

    %Stores time
    time_vec(1,i+1)=i;

    %Stores ideal voltage
    ideal_voltage(1,i+1)=0.5;

    %Stores noisy voltage measurements
    d = z{i+1,1};
    measurements_with_noise(1,i+1)=d(1,1);

    %Stores Kalman Filter predicted measurements
    g=z_predicted{i+1,1};
    predicted_observations(1,i+1)=g(1,1);


end


%Plot of the constant 0.5 V DC Voltage, Kalman Filter predicted voltage, and
%the noisy voltage measurements
figure()
plot(time_vec,ideal_voltage,'r'); hold on;
plot(time_vec,measurements_with_noise,'.-b'); hold on;
plot(time_vec,predicted_observations,'--g');
legend('Constant 0.5 DC Voltage','Measured Values of Voltage','Kalman Filter
    for Voltage')
title('Voltage vs. Time')
xlabel('Time')
ylabel('Voltage');
```

```

%Name: Alec Knutsen
%Date:12/08/15
%Description:This Program runs the Discrete Kalman Filter Algorithm for the
%projectile motion example described in the paper.
%
%
%State Space Model:
    %   Equation 1: x_k+1 = (F_k)(x_k)(G_k)(u_k) + w_k
    %   Equation 2: z_k = (H_k)(x_k) + v_k
    %
    % Parameters:
        % x_k - State vector at time k with deminsions nx1
        % u_k - Control input at time k with dimensions mx1
        % z_k - Observation at time k with dimensions px1
        % F_k - State transition matrix at time k with dimensions nxn
        % G_k - Input transition matrix at time k with dimensions nxm
```

```
        % H_k - Observation matrix at time k with dimensions pxn
        % w_k - Process noise at time k with dimensions nx1
        % v_k - Additive noise measurement at time k with dimension px1
        % Q_k - Covariance matrix (Q= E[(w_k)(w_k)^T]) at time k with
        % dimensions nxn
        %R_k - Covariance matrix (R_k= E[(v_k)(v_k)^T]) at time k with
        % dimensions pxp
        %P_k - Covariance error matrix(P_k = E[(xhat_k - x_k)(xhat_k-x_k)^T])

%We have the following Kalman Filter Equations:
    %Initial Conditions:
        %xhat_0 = E[x_0]
        %P_0|0 = E[(xhat_0 - x_0)(xhat_0-x_0)^T]
    %Prediction Equations:
        %Equation 3: xhat_k+1|k = (F_k)(xhat_k|k) + (G_k)(u_k)
        %Equation 4: P_k+1|k = (F_k)(P_k|k)(F_k)^T + Q_k
    %Update Equations:
        %Equation 5: xhat_k+1|k+1 = xhat_k+1|k + K_k+1(z_k+1 - (H_k+1))(xhat_k+1|k)
        %Equation 6: P_k+1|k+1 = (I - (K_k+1)(H_k+1))(P_k+1|k)(I - (K_k+1)(H_k+1))^T +
        %                + (K_k+1)(R_k+1)(K_k+1)^T

% Extra Equations:
    % Kalman Gain: K_k+1 = (P_k+1|k)(H_k+1)^T[(H_k)(P_k+1|k)(H_k)^T +
    %                        R_k+1]^-1


%Beginning of Program:

%User input: n - Size of state vectors
n=4;

%User input: m - Size of input vectors
m=4;

%User input: p - Size of observation vectors
p=2;

%time - Discrete time variable
time=1;

%User input: num_est - Total number of states you want to estimate
%Projectile Example: This estimates from 0-120 seconds at 0.1 second time
%intervals
num_est=1200;

%xhat - Cell array that will store each updated state vector (xhat_k|k)
% Each updated state vector is size n x 1
xhat = cell(num_est,1);
```

```
%Initialize each estimated state vector to be zeros. These will be replaced with
%estimates
for  i =1:num_est
    xhat{i,1} = zeros(n,1);

end

%u - Cell array that will store each input vector (u_k)
%Each input vector is size mx1
u = cell(num_est,1);
%User input: See the paper for proper initialization
for  i =1:num_est
    u{i,1} = [0;   0;   0;  -0.98];

end


%F - Cell array that stores each F (state transistion) matrix
%Each F matrix is size nxn
%User input: See the paper for proper initialization
F = cell(num_est,1);
for  i =1:num_est
    F{i,1} = [1 0 0.1 0; 0 1 0 0.1;0 0 1-(0.0001) 0;0 0 0 1-(0.0001)];

end


%G - Cell array that stores each G (input transition) matrix
%Each G matrix is size nxm
G = cell(num_est,1);
%User input: See the paper for proper initialization
for  i =1:num_est
    G{i,1} = [1 0 0 0; 0 1 0 0; 0 0 1 0 ; 0 0 0 1];

end

%H - Cell array that stores each H matrix
%Each H matrix is size pxn
H = cell(num_est,1);
%User input: See the paper for proper initialization
for  i =1:num_est
    H{i,1} = [1 0 0 0;0 1 0 0;];

end

%Q - Cell array that stores each covariance matrix for the noise w_k
%The size of each Q is nxn
Q = cell(num_est,1);
%User input: See the paper for proper initialization.
```

```
for  i =1:num_est
    Q{i,1} = [0.1 0 0 0;0 0.1 0 0;0 0 0.1 0;0 0 0 0.1];


end

%R - Cell array that stores each covariance matrix for the noise v_k
%The size of each R is pxp
R = cell(num_est,1);
%User input: See the paper for proper initialization
for  i =1:num_est
    R{i,1} = [500 0;0 500];

end

%W - Cell array that will store each noise vector
%The noise vectors are of size nx1
W = cell(num_est,1);

%Note the noise is automatically generated based on the covariance matrix
%using mvrnd
for  i =1:num_est
    W{i,1} = (mvnrnd(zeros(1,n),Q{1,1}))';

end

%V - Cell array that will store each noise vector
%The v vectors are size px1
V = cell(num_est,1);
%Note the noise is automatically generated based on the covariance matrix
%using mvrnd
for  i =1:num_est
    V{i,1} = (mvnrnd(zeros(1,p),R{1,1}))';

end

%P - Covariance matrix of errors (difference between estimated state and
    actual state)
%Each p matrix is size nxn
P = cell(num_est,1);
%We initially fill all covariance matrices to be a matrix of zeros
for  i =1:num_est
    P{i,1} = zeros(n,1);

end

%z_predicted - Stores each predicted observation (zhat_k+1|k)
%The size of each element predicted obervation is px1
z_predicted = cell(num_est,1);
for  i =1:num_est
```

```
    %We initially fill all z predicted vectors to be vectors of zeros
    z_predicted{i,1} = zeros(p,1);
end

%x - Cell array that will store a generated sequence of state vectors (x_k) given
%an initial state vector.
% Each state vector is size n x 1
x = cell(num_est,1);

%z - Matrix that will store each observation vector (z_k) in its columns
%Each observation vector is px1
z = cell(num_est,1);

%ideal_observations - Matrix that will store each z_k with noise w_k,v_k
%removed
%Each ideal observation vector is px1
ideal_observations = cell(num_est,1);

%User Input: P_0 - Initial Covariance Matrix.  In this example, use the
%identity matrix.
P_0 = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];

%Store initial value in P_cell
p_update = P_0;

%Initiialize initial state vector. Assume we know the initial conditions
%for projectile motion.
x_1 = [0;0;300;600];
x{1,1} = x_1;
x_update = x{1,1};

%Generate the first observation vector
z{1,1} = H{1,1}*x{1,1} + V{1,1};

%Generate first ideal motion value (no noise)
ideal_observations{1,1} = H{1,1}*x{1,1};

%Generate a sequence of state vectors (x_k) based on the initial conditions.
%Also, generate a sequence of observation vectors (z_k)
%Finally, generate a sequence of observations with no noise w_k,v_k
for  i = 2:num_est
   x{i,1} = F{i-1,1}*x{i-1,1} + G{i-1,1}*u{i-1,1} + W{i-1,1};
   z{i,1} = H{i,1}*x{i,1} + V{i,1};

   x_no_noise = F{i-1,1}*x{i-1,1} + G{i-1,1}*u{i-1,1};
   ideal_observations{i,1} = H{i,1}*x_no_noise;

end

```

```
while(time<=num_est);


    %Store updated estimates for state(xhat_k+1|k+1) and error covariance (
        Phat_k+1|k+1)
    xhat{time,1} = x_update;
    P{time,1} = p_update;

    % Implements prediction equations 3 and 4
    [x_pred, p_pred] = predict(x_update,p_update, F{time,1}, Q{time,1},G{time
        ,1},u{time,1});

    % Implements update equations 5 and 6
    [x_update,p_update, K,z_predictions] = update(x_pred, p_pred, z{time,1},H{
        time,1}, R{time,1});

    %Store z_predicted (zhat_k+1|k)
    z_predicted{time,1}= z_predictions;

    time=time+1;




end;

%END OF PROGRAM ANALYSIS AFTER.

%Store time for plotting purposes
time_vec = zeros(1,num_est);

%This matrix will hold the ideal projectile motion observations (z_k with no
    noise w_k, v_k)
ideal_motion_mesurements = zeros(2,num_est);

%This matrix will hold the actual measurements (z_k)
measurements_with_noise = zeros(2,num_est);

%This matrix will hold the predicted observations (zhat_k+1)
predicted_observations = zeros(2,num_est);


%This stores the elements of the cell array elements into the the above
%vectors
for i=0:num_est-1

    %Stores time
    time_vec(1,i+1)=i*0.1;

    %Store ideal x,y position (z_k with no noise w_k, v_k)
```

```
    b = ideal_observations{i+1,1};
    ideal_motion_mesurements(1,i+1)=b(1,1);
    ideal_motion_mesurements(2,i+1)=b(2,1);

    %Store x,y position of observations with noise(z_k)
    d = z{i+1,1};
    measurements_with_noise(1,i+1)=d(1,1);
    measurements_with_noise(2,i+1)=d(2,1);

    %Store x,y position of predicted observations from Kalman Filter (zhat_k
        +1|k)
    g=z_predicted{i+1,1};
    predicted_observations(1,i+1)=g(1,1);
    predicted_observations(2,i+1)=g(2,1);


end


%Plot of the ideal projectile motion, the noisy measurements motion, and
%the Kalman Filter predicted motion for the time between 4 and 6 seconds
%and 0.1 second time intervals
figure()
plot(ideal_motion_mesurements(1,400:600),ideal_motion_mesurements(2,400:600),'
    r'); hold on;
plot(measurements_with_noise(1,400:600),measurements_with_noise(2,400:600),'.'
    ); hold on;
plot(predicted_observations(1,400:600),predicted_observations(2,400:600),'--g'
    );
legend('True Projectile Motion for Measurements','Observed Measurement Values'
    ,'Kalman Filter')
title('Y Position vs. X Position')
xlabel('X Position')
ylabel('Y Position')
```

References

[1] Greg Czerniak. "Kalman Filters for Undergrads 1". Online resource at http://greg.czerniak.info/guides/kalman1/. 2015.

[2] Jefferey Humpherys, Preston Redd, and Jeremy West. "A Fresh Look at the Kalman Filter". In: *SIAM REVIEW* 54.4 (2012).

[3] R. E. KALMAN. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME Journal of Basic Engineering* 82 (1960), pp. 35–45.

[4] Alan V. Oppenheim and George C. Verghese. "Estimation with Minimum Mean Square Error". Online resource found through http://ocw.mit.edu. 2010.

[5] Dr. Pasik-Duncan. "Stochastic Adaptive Control". Lecture Notes from a class at the University of Kansas. 2015.

[6] Ian Reid and Hilary Term. "Estimation II". Lecture notes found online at http://www.robots.ox.ac.uk/ ian/Teaching/Estimation/LectureNotes2.pdf. 2001.

[7] Johannes Traa. "Matrix Calculus - Notes on the derivative of a trace". Online resource. 2010. URL: `http://cal.cs.illinois.edu/~johannes/research/matrix%20calculus.pdf`.