HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

Digital Signal Processing - Semester 201

Class: CC01

Lecturer: PHAM HOANG ANH

KALMAN FILTER

Group member	*Student ID*
DOAN ANH TIEN	1852789
NGUYEN DUY TINH	1852797
HO HOANG THIEN LONG	1852161

December 5, 2020

Table of contents

1. Principle and its applications

1.1. The basic idea of the algorithm

Kalman filters are used to estimate states based on linear dynamical systems in state space format. The **process model** defines the evolution of the state from time k−1 to time k as:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

in which:
A *is the state transition matrix applied to the previous state vector* x_{k-1}
B *is the control-input matrix applied to the control vector* u_{k-1}
w_{k-1} *is the process noise vector that is assumed to be zero-mean Gaussian with the covariance* Q

The process model is paired with the **measurement model** that describes the relationship between the state and the measurement at the current time step k as:

$$z_k = Cx_k + v_k$$

in which:
C *is the measurement matrix*
v_k *is the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance* R

The Kalman filter algorithm consists of two stages: prediction and update. The algorithm is summarized as follows[1]:

Prediction	Predicted state estimate	$\hat{x}^-_k = A\hat{x}^+_{k-1} + Bu_{k-1}$
	Predicted error covariance	$P^-_k = AP^+_{k-1}A^T + Q$

Update	Measurement residual	$\tilde{y}_k = z_{k-1} - C\hat{x}^-_k$
	Kalman gain	$K_k = P^-_kC^T(R + CP^-_kC^T)^{-1}$
	Updated state estimate	$\hat{x}^+_k = \hat{x}^-_k + K_k\tilde{y}$
	Updated error covariance	$P^+_k = (I - K_kC)P^-_k$

[1] *In the equations, the hat operator means an estimate of a variable. That is, x^ is an estimate of x. The superscripts − and + denote predicted (prior) and updated (posterior) estimates, respectively.*

1.2. Applications

The Kalman filter has since been developed as a tool that combines current uncertain information with turbulent environmental information into a new, more reliable form of information for predicting, with the strength of running very fast and highly stable.

Such applications has been applied with the Kalman filter:

- Tracking objects (e.g., missiles, faces, heads, hands)
- Fitting Bezier patches to (noisy, moving, ...) point data
- Economics
- Navigation
- Many computer vision applications
    - Stabilizing depth measurements
    - Feature tracking
    - Cluster tracking
    - Fusing data from radar, laser scanner and stereo-cameras for depth and velocity measurements

2. Describe how to apply Kalman filter in some practical applications

2.1. The rocket problem

Situation

Suppose that there is an astronaut flying out into space, he needs to adjust the temperature of the chamber so it doesn't damage his spacecraft. At this point, he needs to know the temperature inside the chamber and the problem is that he can't put the sensor in there. So instead, he sits out of the chamber to measure the temperature outside of the chamber and uses this parameter along with calculations to calculate the internal temperature.

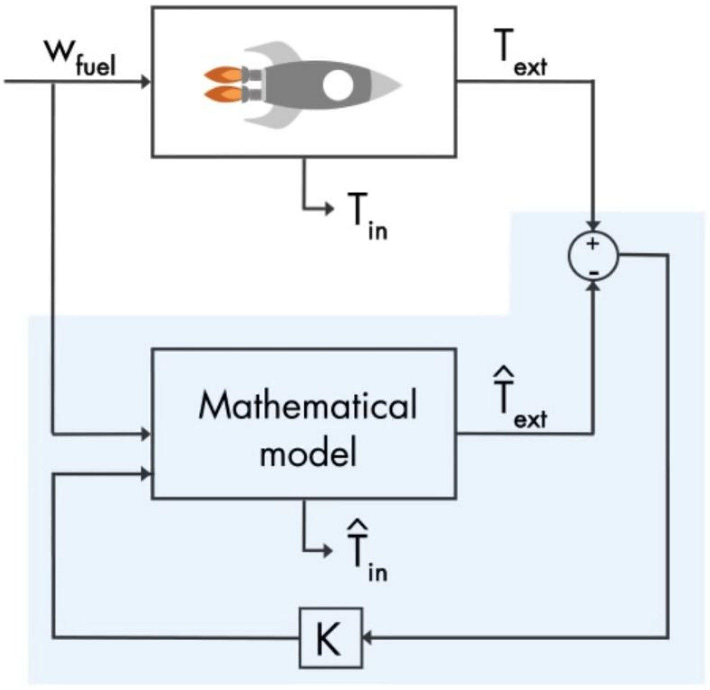


Figure 1. The system to measure the temperature inside booster

As from the graph, there are variables:

T_{ext} : *measured temperature outside the engine booster*
T_{in} : *measured temperature inside the engine booster*
$\hat{T}_{ext}$: *estimated temperature outside the engine booster*
$\hat{T}_{in}$: *estimated temperature inside the engine booster*
K : *controller gain (to minimize the observed error in the process)*
W_{fuel} : *supplied fuel flow*

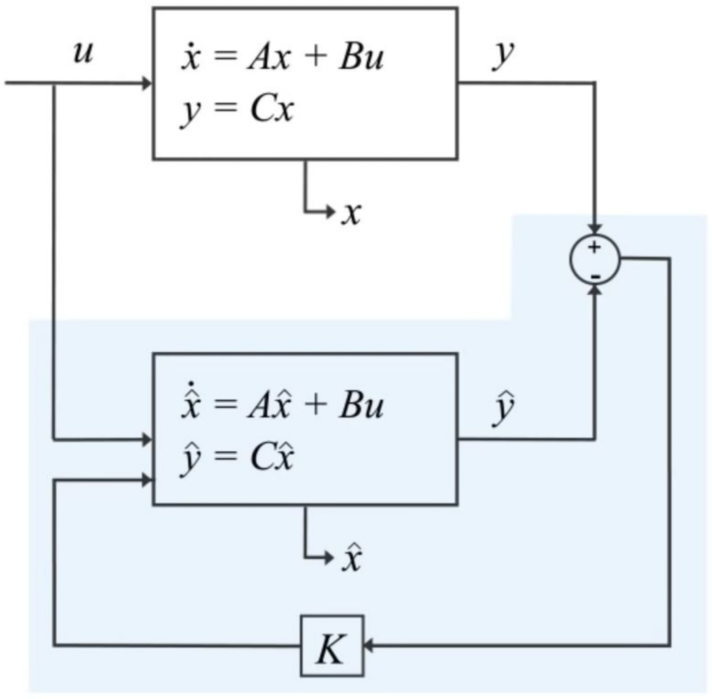


Figure 2. The rocket dynamics and rocket model system

The figure above generalizes the system and shows the input as **u**, the output as **y**, and any states we want to estimate as **x**. And the goal is to drive $\widehat{x}$ to x and define the difference between these values as an error.

Mathematical

$$e_{obs} = x - \widehat{x}\,; \qquad y - \widehat{y} = C(x - \widehat{x})$$

$$x' = Ax + Bu;\ (1) \qquad \widehat{x}\,' = A\widehat{x} + Bu + K(y - \widehat{y})\ (2)$$

$$(1)(2) \Rightarrow x' - \widehat{x}\,' = A(x - \widehat{x}) - K(y - \widehat{y})$$

$$\Rightarrow x' - \widehat{x}\,' = A(x - \widehat{x}) - KC(x - \widehat{x})$$

$$e_{obs}' = (A - KC)\, e_{obs} \ \Rightarrow \frac{de_{obs}}{dt} = (A - KC)e_{obs} \Rightarrow \frac{de_{obs}}{e_{obs}} = (A - KC)dt$$

$$\Rightarrow \int_0^{e_{obs}} \frac{de_{obs}}{e_{obs}} = \int_0^t (A - KC)dt \Rightarrow ln\left|e_{obs}\right| = (A - KC)t \Rightarrow e_{obs} = e^{(A-KC)t} \quad (1)$$

$$From\ (1) \Rightarrow If (A - KC) < 0\ then:$$

$$e_{obs} \to 0\ when\ t \to \infty \Leftrightarrow \widehat{x} \to x\ (what\ we\ need\ initially)$$

2.2. Self-driving car locates itself using GPS

Situation

Another example for implementing the Kalman Filter is to apply it on automated transportation. Imagine we are joining a competition to win the big prize as we have asked to design a self-driving car that needs to drive 1 km on 100 different terrains. In each trial, the car must stop as close as possible to the finish line.

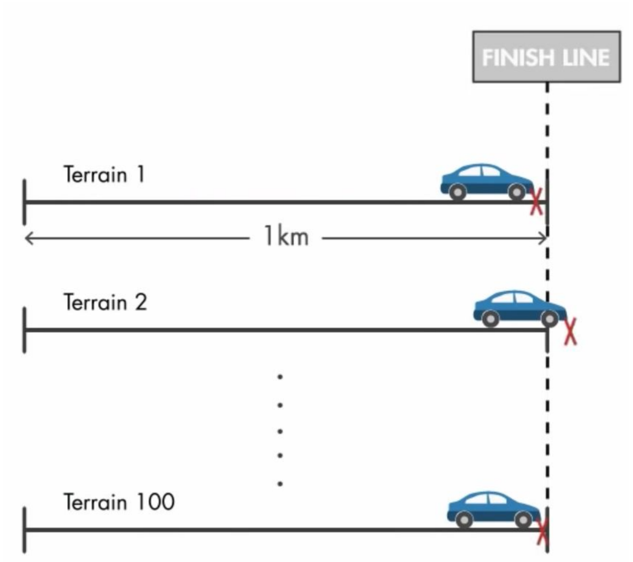


Figure 3. The final position marked in each trials

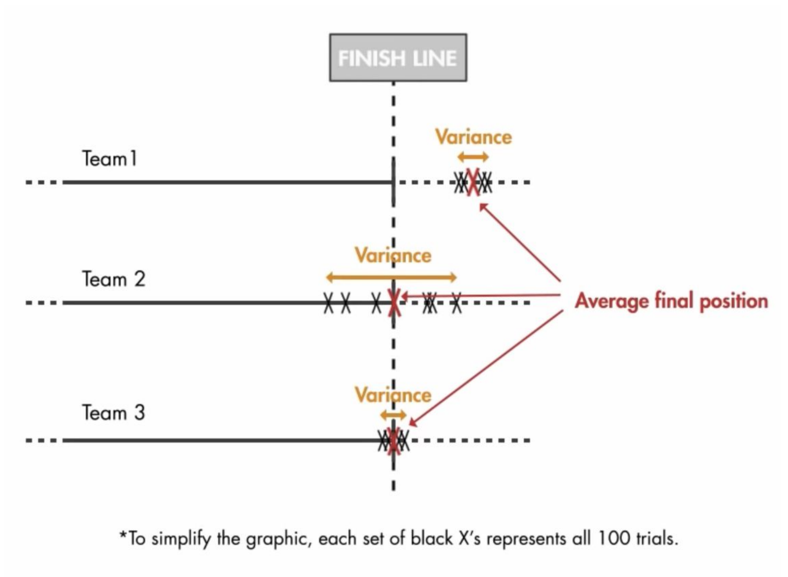

FINISH LINE
Team1
Variance
Team 2
Variance
Average final position
Team 3
Variance
*To simplify the graphic, each set of black X's represents all 100 trials.

Figure 4. Computed average final position computed of each team

At the end of the competition, the average final position is computed for each team, and the owner of the car with **the smallest** error variance and the final position **closest** to 1 km will be the winner.

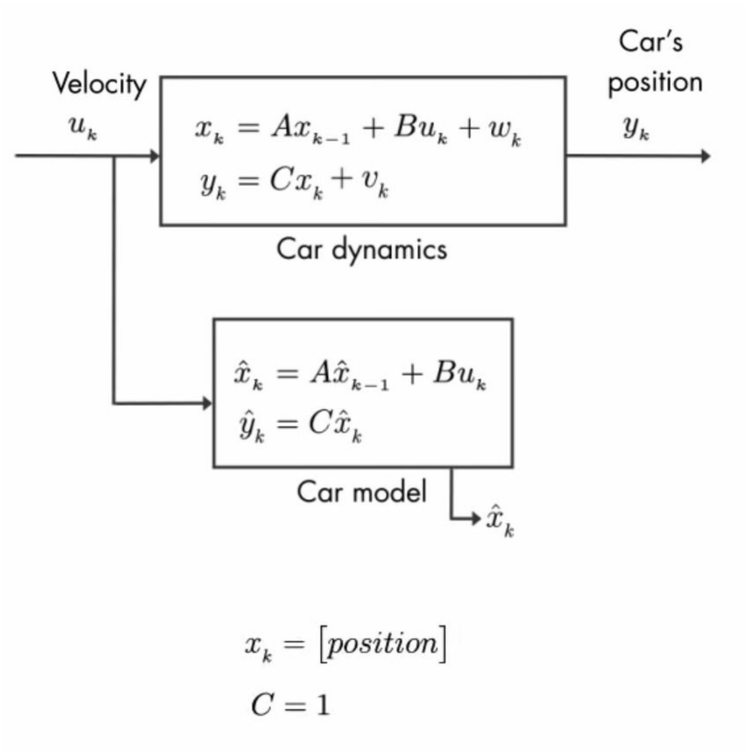


Figure 5. Car dynamics and car model system

Let say we could win the competition by using Kalman filter, which computes an optimal unbiased estimate of the car's position with minimum variance (Figure 6)

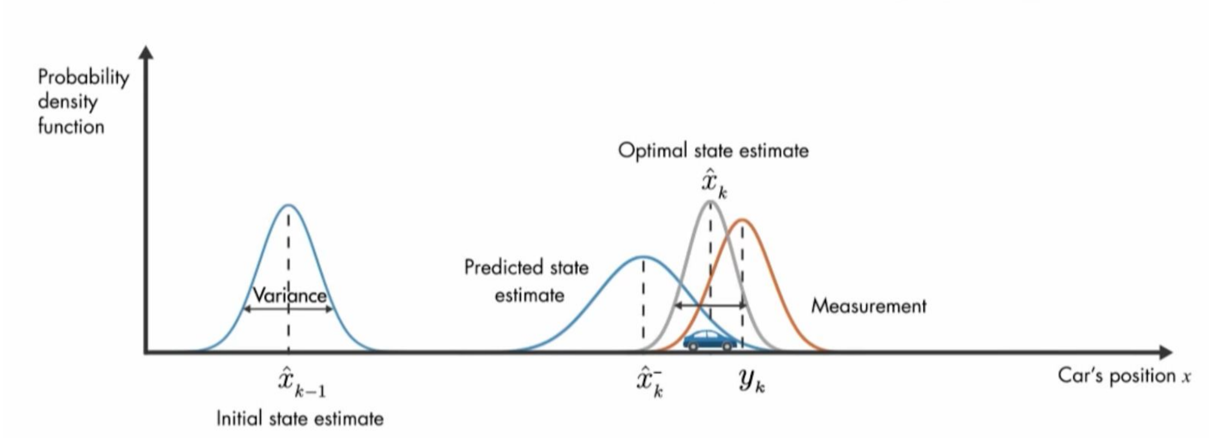


Figure 6. Car's position and the probability density function

This optimal estimate is found by multiplying the prediction and measurement probability function together (the blue and orange line around the gray one, respectively). The mentioned multiplication relates to the Kalman filter equation shown below:

$$\widehat{x}_{k+1} = A\widehat{x}_{k-1} + Bu_{k-1} + K_k(y_k - C(A\widehat{x}_{k-1} + Bu_{k-1}))$$

Now we consider back to the two compulsory stages in Kalman filter which are to predict and update progress.

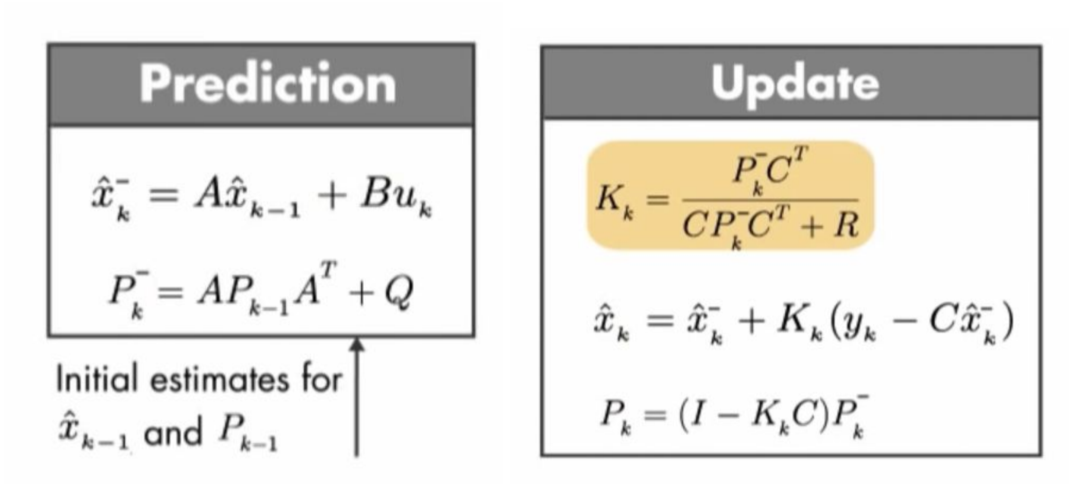

Prediction
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
$P_k^- = AP_{k-1}A^T + Q$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}
Update
$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$
$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$
$P_k = (I - K_k C)P_k^-$

Figure 7. Car's position and the probability density function

At the very start of the algorithm, $\widehat{x}_{k-1}$ and P_{k-1} are used as initial values to calculate the prior state estimate $\widehat{x}^-_k$ and error covariance P^-_k. The outputs will then being used for the update step to find out the posterior estimates of the states and error covariance.

The **Kalman gain** (the highlighted one) is calculated such that it minimizes the posterior error covariance P_k. In order to understand more about how it works, we will look at two extreme cases:

- **Case 1: Measurement covariance R -> 0**

  To calculate the Kalman gain, we take its limit as R goes to zero:

$$\lim_{R\to 0} K_k = \lim_{R\to 0} \frac{P_k^- C^T}{CP_K^- C^T + R} = \lim_{R\to 0} \frac{P_k^- C^T}{CP_K^- C^T + 0} = C^{-1}$$

  Since C = 1 from the beginning, therefore $C^{-1} = 1$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) = \hat{x}_k^- + C^{-1}(y_k - C\hat{x}_k^-)$$

$$\hat{x}_k = \hat{x}_k^- + C^{-1}y_k - C^{-1}C\hat{x}_k^- = C^{-1}y_k = y_k$$

  From the result, we can conclude that the calculation comes from the measurement only (y_k)

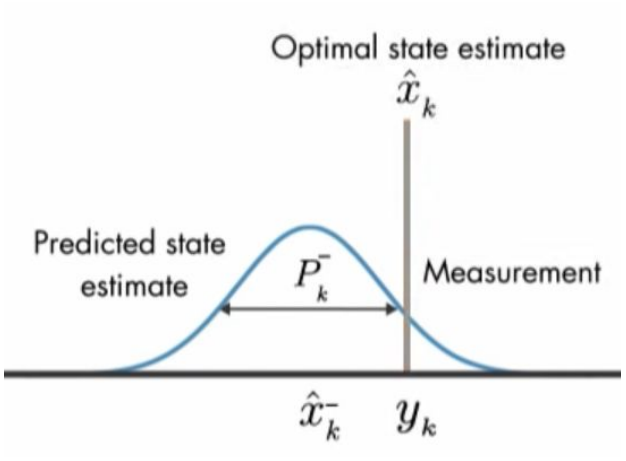


Figure 8. The measurement as vertical line equals to optimal state estimate

The variance in the measurement is zero since R goes to 0, which means the measurement can be shown with an impulse function (vertical line). In addition, it also equals the posterior estimate after being updated.

- **Case 2: prior error covariance $P^{-}{}_{k}$ -> 0**

  In the same way, we take its limit as $P^{-}{}_{k}$ goes to zero:

$$\lim_{P_k^- \to 0} K_k = \lim_{P_k^- \to 0} \frac{P_k^- C^T}{C P_K^- C^T + R} = \lim_{P_k^- \to 0} \frac{0}{0+R} = 0$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) = \hat{x}_k^- + 0(y_k - C\hat{x}_k^-)$$

$$\hat{x}_k = \hat{x}_k^-$$

From the result, we can conclude that the calculation comes from the prior state estimate ($\widehat{x}^{-}{}_{k}$)

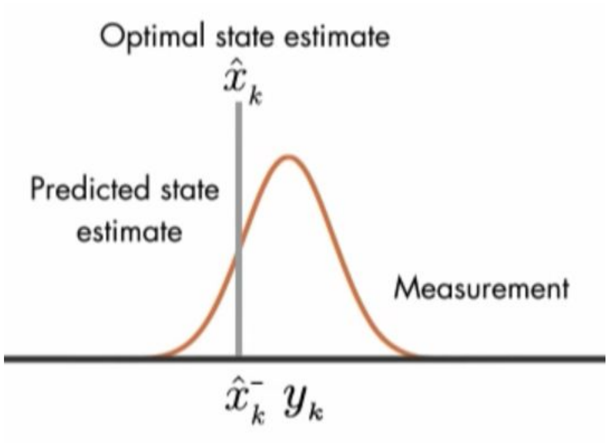


Figure 9. The prior state estimate as vertical line equals to optimal state estimate

With the same fashion of case 1, the variance in the prior state estimate is zero since $P^{-}{}_{k}$ goes to 0, which means the prior one can be shown with an impulse function (vertical line) and equals to posterior state estimate

Once we have calculated the updated equations, the posterior estimates will be used to predict a new prior estimates and the algorithm repeats itself (as shown below)

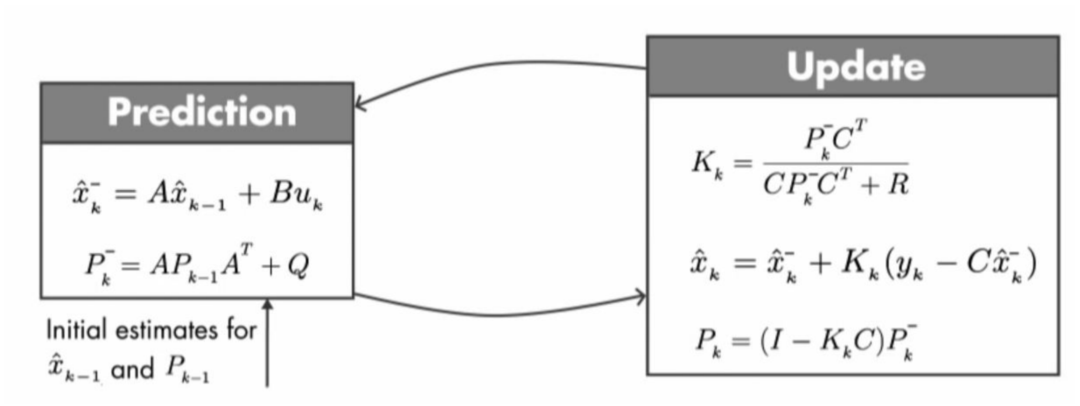

Prediction
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
$P_k^- = AP_{k-1}A^T + Q$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}
Update
$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$
$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$
$P_k = (I - K_k C)P_k^-$

Figure 10. Recursion of Kalman filter

3. Demonstration or Simulation in Scilab

3.1. Situation

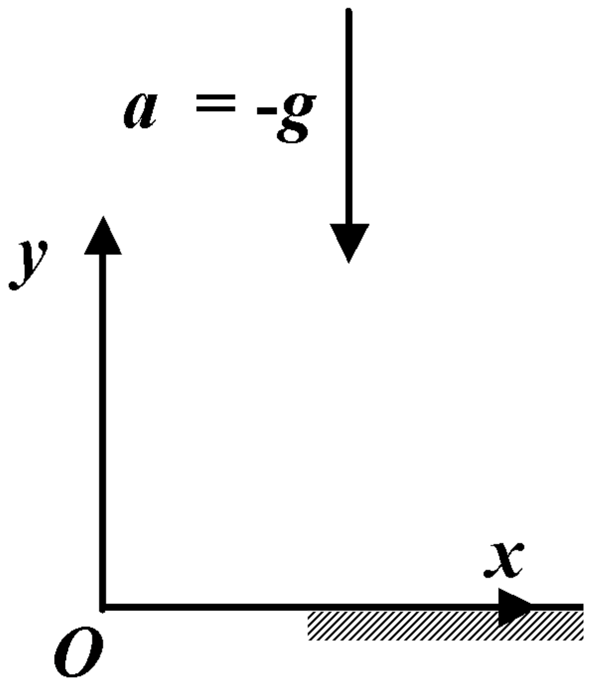


Figure 11. Freefall problem diagram

Initially, consider 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:

(I) uncertain information about the initial position.
(II) 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 is h, we have:

$$h(t)'' = \frac{h(t)' - h(t-\Delta t)'}{\Delta t} = -g$$

$$\Rightarrow h(t)' = \Delta h = h(t-\Delta t)' - g\Delta t$$

$$\Rightarrow \int_{h(t-\Delta t)}^{h(t)} dh = \int_{0}^{\Delta t} (h(t-\Delta t) - g\Delta t)dt$$

$$\Rightarrow h(t) = h(t-\Delta t) + h(t-\Delta t)'\Delta t + \tfrac{1}{2}g\Delta t^2 \ (1)$$

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 = kΔt:

$$h(t) = h(k\Delta t) = h_k \quad (2)$$
$$h(t-\Delta t) = h(k\Delta t - \Delta t) = h((k-1)\Delta t) = h_{k-1} \quad (3)$$

From these kinematic expressions, we see that we have two equations describing the motion of the object: **velocity** (I) and **position** (II)

$$(1)(2)(3) \qquad \Rightarrow h_k' = h_{k-1}' - g\Delta t \ \text{(II)}$$
$$\Rightarrow h_k = h_{k-1} + h_{k-1}'\Delta t - \tfrac{1}{2}g\Delta t^2 \ \text{(I)}$$

$$x_k = \begin{bmatrix} h_k \\ h_k' \end{bmatrix} = \begin{bmatrix} h_{k-1} + h_{k-1}'\Delta t - \frac{1}{2}g\Delta t^2 \\ h_{k-1}' - g\Delta t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} -\frac{1}{2}\Delta t^2 \\ -\Delta t \end{bmatrix} g$$

$$with\ F_{k-1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix},\ G_{k-1} = \begin{bmatrix} -\frac{1}{2}\Delta t^2 \\ -\Delta t \end{bmatrix} \quad \Rightarrow x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1}$$

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:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

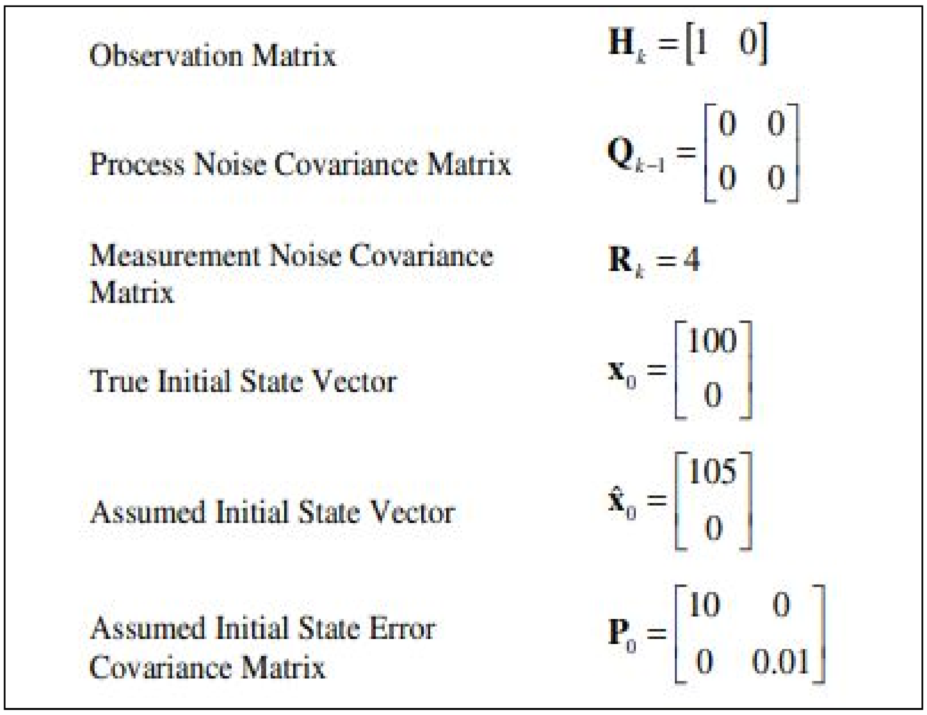
Next, we need to consider the measurement part of the system. Let us consider a scenario where the position of the object can be measured using a laser rangefinder with **2m standard deviation** of error. Because the position is what can be measured, we need to define an output equation that gives the 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:

$$y_k = h_k + v_k = \begin{bmatrix} 1 & 0 \end{bmatrix} x_k + v_k = H_k x_k + v_k, \; with \; H_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The considered measurement system has a **standard deviation of error of 2 m**, which is **a variance of 4 m2**. 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:

$$R = 4\ m^2$$

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 **100m**. 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 m2** 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 m2/s2** is assumed.



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	$\mathbf{P}_0 = \begin{bmatrix} 10 & 0 \\ 0 & 0.01 \end{bmatrix}$

Figure 12. Summarize the assumed functional matrices

3.2. Source code simulated on SciLab

```
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 initial 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)*rand(1,N,"normal"); // 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);
set(gca(),"auto_clear","off"); plot(t, xt(1,:), 'r:', 'LineWidth', 1.5)
xlabel('t (s)'); ylabel('x_1 = h (m)'); set(gca(),"grid",[1 1]);
legend('Measured','Estimated','True');
subplot(212);
plot(t, x(2,:), 'b--', 'LineWidth', 2);
set(gca(),"auto_clear","off"); plot(t, xt(2,:), 'r:', 'LineWidth', 1.5)
xlabel('t (s)'); ylabel('x_2 = v (m/s)'); set(gca(),"grid",[1 1]);
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)'); set(gca(),"grid",[1 1]);
subplot(212);
plot(t, x(2,:)-xt(2,:), 'm', 'LineWidth', 2)
xlabel('t (s)'); ylabel('\Deltax_2 (m/s)'); set(gca(),"grid",[1 1]);
```

Result:

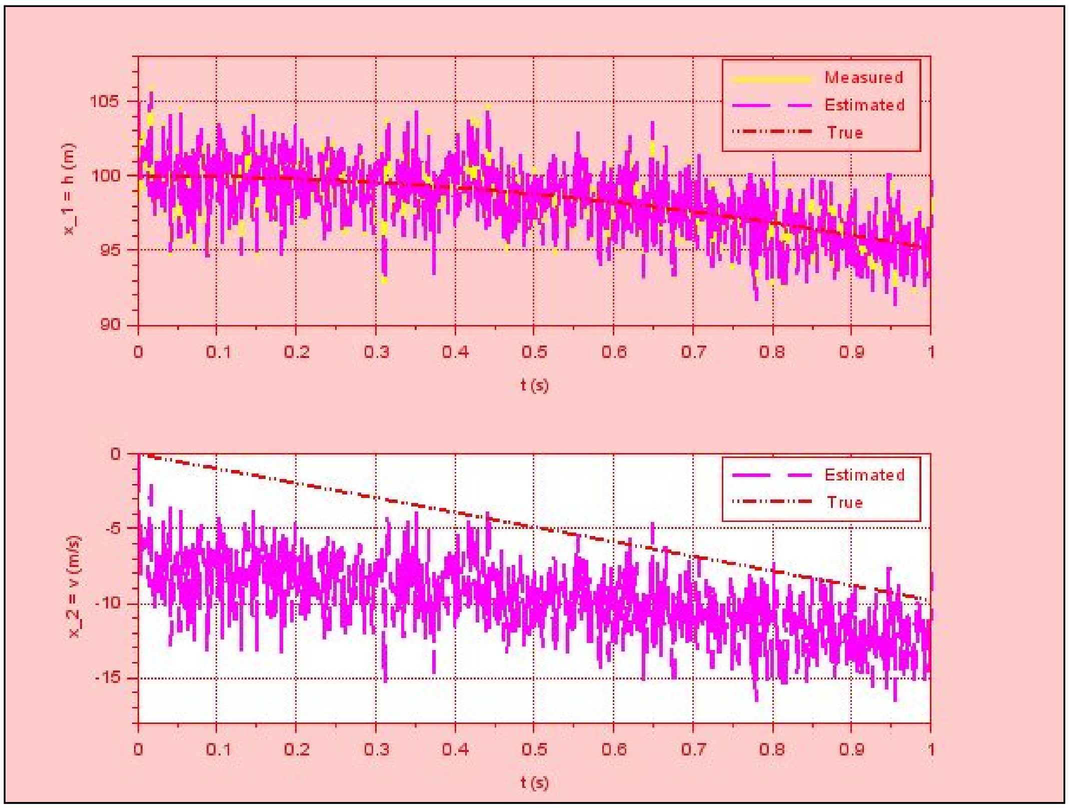


Figure 13. Plot of states on SciLab

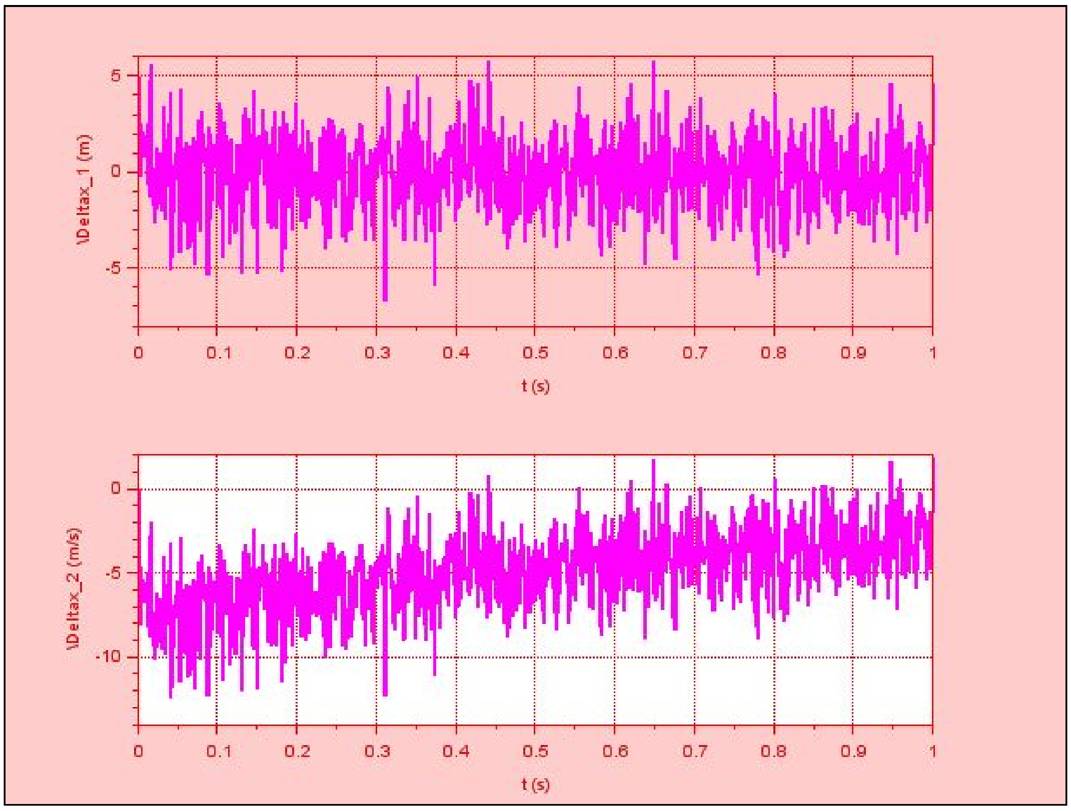


Figure 14. Plot of estimation errors on SciLab

References

Kim, Youngjoo, and Hyochoong Bang. *Introduction to Kalman Filter and Its Applications*. Felix Govaers, 2018. *IntechOpen*, https://www.intechopen.com/books/introduction-and-implementations-of-the-kalman-filter/introduction-to-kalman-filter-and-its-applications.

Le, Tien Quang. “Kalman Filter và bài toán chuỗi thời gian.” *ThetaLog*, 25 October 2018, https://thetalog.com/machine-learning/kalman-filter/.

MATLAB. “Understanding Kalman Filters.” *Youtube*, MATLAB, 31 January 2017, https://www.youtube.com/playlist?list=PLn8PRpmsu08pzi6EMiYnR-076Mh-q3tWr.