On Linear-Kalman-Filter And Its Applications In Finance

Debanjan Manna (190255)
Dept. of Aerospace Eng.[*]
debmanna@iitk.ac.in

Shubham Indoliya (190830)
Dept. of Economic Sci.[*]
shubhind@iitk.ac.in

Utkarsh Verma (190932)
Dept. of Civil Eng.[*]
utkrsh@iitk.ac.in

[*] IIT Kanpur, India

Abstract

Will explain how kalman filter works The paper explains LKF equations in detail and also discusses LKF formulation for a general LTI system; which has been followed by a direct application stock price prediction.

1 Introduction

In 1960, Rudolf E. Kalman introduced the Kalman filter, which is an algorithm designed to estimate nonobservable state variables using observable variables that may contain measurement errors. The Kalman Filter can identify the hidden state of a dynamic linear system, it can also function when the system is subjected to additive noise.

Kalman Filter can be applied to any field which involves "estimation" and "prediction" of states .

Kalman filter is an easy to implement recursive algorithm. It is appropriate for use when dealing with multivariable systems, timevarying systems, and nonstationary random processes.

The linear nature of the Kalman Filter permits the estimation of variables based on linear relationships between them, while its optimality guarantees that the estimated variables are as accurate as possible relative to their true values.

`Linear Kalman Filter`(henceforth LKF) is inherently applicable to systems whose dynamics is linear (or can be reasonably approximated to have a linear dynamics within a certain interval of time over which the analysis is made).

Researchers have also developed various extensions to the algorithm to apply kalman Filter on nonlinear systems. These extensions include the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). However, we have only discussed Linear Kalman Filter in our project.

2 One-dimensional Kalman Filter

Let us consider a `static system` (static wrt some state variables; for the dummy example that we are going to construct let us assume it to be one);

Example: Empty of an engine, Speed of light in any particular medium etc.

To obtain the true value of a state variable (whose dynamics is constant) we can take multiple measurements (over the period of time when the dynamics is constant) and average them.

• Consider that we are tasked to find the empty weight of an engine:

Let x denote the true weight of the engine then,

$$\hat{x}_{n,n} = \frac{\Sigma_{i=1}^{n} z_i}{n} \tag{1}$$

Where
z_n is the measured value of x at time step n
$x_{n,n}$ is `an estimation of x`, made using the n measured weights at time step n

By using (1) (for time step n and n-1) one can show that:

$$\hat{x}_{n,n} = \hat{x}_{n-1,n-1} + \frac{1}{n}(z_n - \hat{x}_{n-1,n-1}) \tag{2}$$

Where
$x_{n-1,n-1}$ is `an estimate of x`, made using the (n-1) measured weights at time step n-1

For a static system (ie a system with constant dynamics):

$$\hat{x}_{n,n-1} = \hat{x}_{n-1,n-1} \tag{3}$$

Where $\hat{x}_{n,n-1}$ is the `predicted value of x`, made using the (n-1) measured values for time step n-1.

Now using (22) and (3) we have

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha_n(z_n - \hat{x}_{n,n-1}) \tag{4}$$

where $\alpha_n = \frac{1}{n}$ (the factor α_n depends on the choice of the "estimation technique" we select; which in this example is considered to be "average of the observed values") & the expression $(z_n - \hat{x}_{n,n-1})$ is called the measurement residual or `innovation` (as it contains new information).
Observe that (4) requires an initial value $(\hat{x}_{1,0})$ to start the algorithm.

We will call the (4) to be the `State update equation` for a static system.

A working example of (4)

• Estimating the empty weight of GE's CF6-80 turbofan engine (true weight is 3975Kg):

Clearly the empty weight of the engine doesn't change with time and hence it constitute a static system; will apply (4) to estimate the weight.

An implementation in python is there in the appendix A.2
`Intialization:` $\hat{x}_{1,0} = 0$
Here z_n: Measured value, $\hat{x}_{n,n-1}$: Predicted value and $\hat{x}_{n,n}$: Estimated value

Iterations, n	1	2	3	4	5	6	7	8	9	10
z_n	3970	3969	3990	3981	3983	3972	3969	3980	3976	3979
$\hat{x}_{n,n-1}$	0	3970.0	3969.5	3976.3	3977.5	3978.6	3977.5	3976.3	3976.8	3976.7
$\hat{x}_{n,n}$	3970.0	3969.5	3976.3	3977.5	3978.6	3977.5	3976.3	3976.8	3976.7	3976.9

Refer figure 1.

We can rewrite (4) as follows:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n-1}) \tag{5}$$

We have not considered any error in measurement that we discussed yet. In Kalman filtering, the estimation is considered a stochastic process, which means that the estimation variance $(p_{n,n})$ must also be projected to the succeeding state.

Our observations can now be represented by z_n and r_n, where z_n is the observed value and r_n is the variance associated with it.

The previous example that we took is a simple case of filter to establish the state update equation.

Now we establish more equations for Kalman Filter in One Dimension

For the constant dynamic example that we took next state variance can be predicted as

$$p_{n+1,n} = p_{n,n} \tag{6}$$

The state update equation that we have seen can be represented as

$$\hat{x}_{n,n} = \ w_1 z_n + \ w_2 \hat{x}_{n,n-1} \tag{7}$$

$$w_1 + w_2 = 1 \tag{8}$$

$$\hat{x}_{n,n} = \ w_1 z_n + \ (1 - w_1)\hat{x}_{n,n-1} \tag{9}$$

For this

$$p_{n,n} = \ w_1^2 r_n + \ (1 - w_1)^2 \hat{x}_{n,n-1} \tag{10}$$

Here, $p_{n,n}$ represents the variance of the optimal combined estimate $\hat{x}_{n,n}$, while $p_{n,n-1}$ represents the variance of the prior estimate $\hat{x}_{n,n-1}$.

We want to minimise uncertainty in current state so we want to minimise $p_{n,n}$

Minimising $p_{n,n}$ wrt w_1 gives us

$$w_1 = \frac{p_{n,n-1}}{p_{n,n-1} + 1} \tag{11}$$

Now our state update equation can be written as

$$\hat{x}_{n,n} = w_1 z_n + (1 - w_1)\hat{x}_{n,n-1} = \hat{x}_{n,n-1} + \frac{p_{n,n-1}}{p_{n,n-1} + r_n}(z_n - \hat{x}_{n,n-1}) \tag{12}$$

This leads to

$$k_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{13}$$

This is the kalman gain equation for the kalman filter in one dimension.

Now, our objective is to determine the variance of the current state estimate.

$$p_{n,n} = \ w_1^2 r_n + \ (1 - w_1)^2 \hat{x}_{n,n-1} \tag{14}$$

as $w_1 = k_n$

$$1 - w_1 = 1 - k_n = \frac{r_n}{p_{n,n-1} + r_n} \tag{15}$$

Substituting $1 - w_1$ and w_1 we can obtain

$$p_{n,n} = \ (1 - k_n)p_{n,n-1} \tag{16}$$

This is Covariance Update Equation; it modifies the estimated variance of the current state by incorporating new information

Developing an algorithm from these equations

Step 0 - Initialisation

We have to initialise two parameters

• Initial system state ($x_{0,0}$)

• Initial system variance ($p_{0,0}$)

Step 1 - Measurement

Here we take observations for a state

i) current state observation (Z_n)

ii)variance related to current observation (r_n).

Step 2 -System update

The process of system update is in charge of determining the estimated state of the system at present.

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \ K_n(z_n - \hat{x}_{n,n-1}) \tag{17}$$

$$k_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{18}$$

$$p_{n,n} = (1 - k_n)p_{n,n-1} \tag{19}$$

Using these equations we determine kalman Gain and estimate system state and related variance.

Step 3 - Prediction

The prediction process extrapolates the current estimated state and its variance to the upcoming system state.

The equations for state extrapolation and covariance extrapolation rely on the dynamics of the system. During the initial iteration of the filter, the initialization is regarded as the prior state estimate and variance.

For a constant dynamic example that we considered earlier

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} \tag{20}$$

$$p_{n+1,n} = p_{n,n} \tag{21}$$

Real-world systems are subject to uncertainties in their dynamic models, which are referred to as process noise. We have not considered noise yet.

Let q represent the process noise variance.

In the case of noise the covariance extrapolation equation modifies as

$$p_{n+1,n} = p_{n,n} + q_n \tag{22}$$

3 Multi-dimensional Kalman Filter

3.1 What is a LTI system ?

The linear kalman filter that we are going to discuss is applicable only to `Linear Time Invariant` (LTI) system.
A system is "`linear`" if the following two conditions hold simultaneously:

1. `Law of Additivity`: if $\overline{y(t)} = y_1(t) + y_2(t)$ then law of additivity holds else it does not hold. Refer figure 2.
2. `Law of Homogeneity`: if $\overline{y(t)} = ky(t)$ then law of homogeneity holds else it does not hold. Here $k \in \mathbb{R}$. Refer figure 3

A system is "`time invariant`" if the system output variables' ($\boldsymbol{y}(t)$) time dependency is because of the input variables' ($\boldsymbol{x}(t)$) time dependency ie $\boldsymbol{y}(t) = f(\boldsymbol{x}(t))$ ($\boldsymbol{y}(t) \neq f(\boldsymbol{x}(t), t)$)

3.2 Basic equations involved

Linear Kalman Filter (LKF) involves five basic equation:

1. Two prediction equations:
   (a) State Extrapolation Equation: estimation of future state using the knowledge of the present state estimation
   (b) Covariance Extrapolation Equation : measures uncertainty in our prediction (and also considers the uncertainty of the syst
2. Two update equations:
   (a) State Update Equation: estimating the current state based on the known past estimation and present measurement
   (b) Covariance Update Equation: the measure of uncertainty in our estimation
3. Kalman Gain Equation: required for computation of the update equations. The Kalman Gain is a "weighting" parameter for the measurement and the past estimations. It defines the weight of the past estimation and the weight of the measurement in estimating the current state

3.2.1 State extrapolation equation

$$\hat{\boldsymbol{x}}_{\boldsymbol{n+1,n}} = \boldsymbol{F}\hat{\boldsymbol{x}}_{\boldsymbol{n,n}} + \boldsymbol{G}\boldsymbol{u}_{\boldsymbol{n}} + \boldsymbol{w}_{\boldsymbol{n}} \tag{23}$$

where
$\hat{\boldsymbol{x}}_{n+1,n}$ is `predicted state vector` for time step n+1
$\hat{\boldsymbol{x}}_{n,n}$ is `estimated state vector` at time step n
$\boldsymbol{F}$ is `State transition matrix`
$\boldsymbol{G}$ is `control matrix` or `input transition matrix`
$\boldsymbol{u}_{\boldsymbol{n}}$ is `input variable` at time step n
$\boldsymbol{w}_{\boldsymbol{n}}$ is `process noise vector` at time step n.

NOTE: Even though (23) contains $\boldsymbol{w}_n$; the working equation won't contain $\boldsymbol{w}_n$. The `Process noise` is taken care by $\boldsymbol{Q}_n$ of (28)

Any system which is governed by an n^{th} order linear differential equation can be modelled as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{24a}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \tag{24b}$$

If $\boldsymbol{u}(t)$ is (atleast) piecewise continuous on a given interval of time say $[t_0, t_0 + \Delta t]$ (which guarantees that $\boldsymbol{u}(t)$ integrable on the interval $[t_0, t_0 + \Delta t]$) (24a) can be written as:

$$\boldsymbol{x}(t_0 + \Delta t) = [\exp\{\boldsymbol{A}\Delta t\}]\boldsymbol{x}(t_0) + [\int_{t_0}^{t_0+\Delta t} \exp\big\{\boldsymbol{A}(\overline{t_0 + \Delta t} - \tau)\big\}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau] \tag{25}$$

The derivation of (25) from (24a) has been mentioned in A.1 of section A.

Further if $\boldsymbol{u}(t) = \boldsymbol{u}(t_0)$ which is constant over the time interval $[t_0, t_0 + \Delta t]$ then one can simplify (25) to obtain:

$$\boldsymbol{x}(t_0 + \Delta t) = [\exp\{\boldsymbol{A}\Delta t\}]\boldsymbol{x}(t_0) + [\int_{t_0}^{t_0+\Delta t} \exp\big\{\boldsymbol{A}(\overline{t_0 + \Delta t} - \tau)\big\}\boldsymbol{B}d\tau]\boldsymbol{u}(t_0) \tag{26}$$

Observe that on comparing equation (26) with (23) we can consider:

$$\boldsymbol{F} \equiv \exp\{\boldsymbol{A}\Delta t\} \tag{27a}$$
$$\boldsymbol{G} \equiv \int_{t_0}^{t_0+\Delta t} \exp\big\{\boldsymbol{A}(\overline{t_0 + \Delta t} - \tau)\big\}\boldsymbol{B}d\tau \tag{27b}$$

3.2.2 Covariance extrapolation equation

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^{\top} + \boldsymbol{Q}_n \tag{28}$$

where
$\boldsymbol{P}_{n+1,n}$ is `Covariance matrix` (taking care of the uncertainty) of the predicted state for time step n+1
$\boldsymbol{P}_{n,n}$ is `Covariance matrix` (considers the uncertainty) of the estimated state at time step n
$\boldsymbol{F}$ is `State transition matrix`
$\boldsymbol{Q}_n$ is `Process noise covariance matrix`

By definition

$$\boldsymbol{P}_{\boldsymbol{n,n}} = E[(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}_{n,n}})(\hat{\boldsymbol{x}}_{n,n} - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}_{n,n}})^{\top}] \tag{29a}$$
$$\boldsymbol{P}_{\boldsymbol{n+1,n}} = E[(\hat{\boldsymbol{x}}_{n+1,n} - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}_{n+1,n}})(\hat{\boldsymbol{x}}_{n+1,n} - \boldsymbol{\mu}_{\hat{\boldsymbol{x}}_{n+1,n}})^{\top}] \tag{29b}$$
$$\boldsymbol{Q}_n = E[\boldsymbol{w}_n\boldsymbol{w}_n^{\top}] \tag{29c}$$

• Measurement Equation

Before proceeding to the next set of equation we should formulate how the measured data is affected by the measurement process.

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{30}$$

where
$\boldsymbol{Z}_n$ is `measured data` at time step = n
$\boldsymbol{H}$ is an observation matrix
$\boldsymbol{x}_n$ is true system state (hidden state)
$\boldsymbol{v}_n$ is Measurement noise vector at time step = n
For an unbiased measurement process $\mu_{\boldsymbol{v}_n} = 0$. Therefore the `measurement noise covariance matrix` R_n is:

$$R_n = E[\boldsymbol{v_n v_n}^\top] \tag{31}$$

3.2.3 State update equation

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{32}$$

where
$\hat{\boldsymbol{x}}_{n,n}$ is `estimated state vector` at time step n
$\hat{\boldsymbol{x}}_{n,n-1}$ is `predicted state vector` for time step n
$\boldsymbol{K}_n$ is `kalman gain matrix`
$\boldsymbol{Z}_n$ is `measured data` at time step = n
$\boldsymbol{H}$ is `observation matrix`

Like the 1D case, the $(\boldsymbol{z}_n - \boldsymbol{H}\hat{x}_{n,n-1})$ is called "`innovation`" as it takes care of the new information.

3.2.4 Covariance update equation

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^\top + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^\top \tag{33}$$

where
$\boldsymbol{P}_{n,n}$ is `Covariance (uncertainty) matrix` of the estimated state at time step n
$\boldsymbol{P}_{n,n-1}$ is `Covariance (uncertainty) matrix` of the predicted state for time step n
$\boldsymbol{K}_n$ is `Kalman gain matrix`
$\boldsymbol{H}$ is `observation matrix`
$\boldsymbol{R}_n$ is `measurement noise covariance matrix`

3.2.5 Kalman gain equation

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^\top + \boldsymbol{R}_n)^{-1} \tag{34}$$

where
$\boldsymbol{K}_n$ is `Kalman gain`
$\boldsymbol{P}_{n,n-1}$ is `Covariance (uncertainty) matrix` of the predicted state for time step n
$\boldsymbol{H}$ is `observation matrix`
$\boldsymbol{R}_n$ is `measurement noise covariance matrix`

Finally we have constructed a diagram mentioning the five working equations of the LKF 4

4 An application of Linear Kalman Filter

4.1 Prediction of Stock Price

Analysis of stock prices has always remained an open problem. Finance industry is always in search of better but computationally less intensive models. We will be using LKF to model the stock prices (of course assuming the underlying dynamics to be Linear time invariant)[2].

4.1.1 Dynamics of stock Price

We will observe the stock price at discrete time (after every time interval of T).

$$x_{n+1} = x_n + T\dot{x}_n + \frac{1}{2}T^2 a_n \tag{35a}$$

$$\dot{x}_{n+1} = \frac{x_{n+1} - x_n}{T} \tag{35b}$$

$$a_n = \frac{\dot{x}_{n+1} - \dot{x}_n}{T} \tag{35c}$$

$$z_n = x_n + v_n \tag{35d}$$

where
x_n : stock price at time step = n
$\dot{x_n}$: stock price's rate of change of the stock at time step = n
a_n : acceleration of the stock price at time step = n
T : sampling period
z_n : measurement made at time step = n
v_n : measurement noise

Further we will consider $E[a_n] = 0$ to account for the high volatility of the stock price. Let us write the state space model of the

$$w_n = a_n \tag{36a}$$

$$\boldsymbol{x}_n = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \tag{36b}$$

$$\boldsymbol{F} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{36c}$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} \tag{36d}$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{36e}$$

This system of equations can be rewritten as

$$\boldsymbol{x}_{n+1} = \boldsymbol{F}\boldsymbol{x}_n + \boldsymbol{\Gamma} w_n \tag{37a}$$

$$z_n = \boldsymbol{H}\boldsymbol{x}_n + v_n \tag{37b}$$

here
v_n : measurement noise
w_n : process noise

Assuming the measurement method and the process to be unbiased (which is often the case) we have $E[w_n] = 0$ & $E[v_n] = 0$ for any time step n. Therefore `measurement noise covariance matrix`, $\boldsymbol{R}_n = E[v_n {v_n}^\top]$ & `process noise covariance matrix`, $\boldsymbol{Q}_n = E[w_n {w_n}^\top]$

Initialization

$$\hat{\boldsymbol{x}}_{0,0} = E[\boldsymbol{x}_0] = \boldsymbol{\mu_0} \tag{38a}$$

$$\boldsymbol{P}_{0,0} = E[(\boldsymbol{x_0} - \boldsymbol{\mu_0})^T(\boldsymbol{x}_0 - \boldsymbol{\mu_0})] = \boldsymbol{P}_0 \tag{38b}$$

Prediction

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} \tag{39a}$$

$$\boldsymbol{P_{n+1,n}} = \boldsymbol{F}\boldsymbol{P_{n,n}}\boldsymbol{F}^\top + \boldsymbol{\Gamma}\boldsymbol{Q}\boldsymbol{\Gamma}^\top \tag{39b}$$

Kalman Gain

$$\boldsymbol{K}_{n+1} = \boldsymbol{P}_{n+1,n}\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{P}_{n+1,n}\boldsymbol{H}^\top + \boldsymbol{R}_n)^{-1} \tag{40a}$$

Update

$$\hat{\boldsymbol{x}}_{n+1,n+1} = \hat{\boldsymbol{x}}_{n+1,n+1} + \boldsymbol{K}_{n+1}(z_{n+1} - \boldsymbol{H}\hat{x}_{n+1,n}) \tag{41a}$$

$$\boldsymbol{P}_{n+1,n+1} = (\boldsymbol{I} - \boldsymbol{K}_{n+1})\boldsymbol{H}\boldsymbol{P}_{n+1,n} \tag{41b}$$

A Appendix

A.1 Derivation 1:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{IA.42}$$

(IA.42) is defined over a time interval $[t_0, t_0 + \Delta t]$. Note that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are independent of time and $\boldsymbol{A}$ is a square matrix but $\boldsymbol{B}$ is not necessarily a square matrix.

Our target is to obtain an expression for $\boldsymbol{x}$(t) on the time interval $[t_0, t_0 + \Delta t]$ by integrating (IA.42)

Let us define `matrix exponential` of a square matrix $\boldsymbol{A}$ as:

$$e^{\boldsymbol{A}t} = \boldsymbol{I} + \frac{(\boldsymbol{A}t)}{1!} + \frac{(\boldsymbol{A}t)^2}{2!} + \frac{(\boldsymbol{A}t)^3}{3!} + ... + \frac{(\boldsymbol{A}t)^k}{k!} + ... \tag{IA.43}$$

Now we will rearrange (IA.42) and pre multiply with $e^{-\boldsymbol{A}t}$ to obtain

$$e^{-\boldsymbol{A}t}\dot{\boldsymbol{x}}(t) - e^{-\boldsymbol{A}t}\boldsymbol{A}\boldsymbol{x}(t) = \frac{de^{-\boldsymbol{A}t}\boldsymbol{x}(t)}{dt} = e^{-\boldsymbol{A}t}\boldsymbol{B}\boldsymbol{u}(t)$$

On integrating the above expression from $t = t_0$ to $t = t_0 + \Delta t$ and using $[e^{-\boldsymbol{A}t}]^{-1} = e^{\boldsymbol{A}t}$ we have

$$\int_{t_0}^{t_0+\Delta t} \frac{d(e^{-\boldsymbol{A}\tau}\boldsymbol{x}(\tau))}{d\tau} = e^{-\boldsymbol{A}(t_0+\Delta t)}\boldsymbol{x}(t_0 + \Delta t) - e^{-\boldsymbol{A}t_0}\boldsymbol{x}(t_0) = \int_{t_0}^{t_0+\Delta t} e^{-\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{u}(\tau)$$

$$\implies \boldsymbol{x}(t_0 + \Delta t) = e^{\boldsymbol{A}\Delta t}\boldsymbol{x}(t_0) + \int_{t_0}^{t_0+\Delta t} e^{\boldsymbol{A}(\overline{t_0+\Delta t}-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau \tag{IA.44}$$

Q.E.D.

A.2 Implementation 1:

Link to the `.py` code

A.3 Implementation 2:

Link to the `.py` code

References

[1] (www.kalmanfilter.net), A. B. (n.d.). Online Kalman Filter Tutorial. Kalman Filter Tutorial. https://www.kalmanfilter.net/

[2] Xu Yan and Zhang Guosheng. 2015. Application of Kalman Filter in the Prediction of Stock Price. Proceedings of the 5th International Symposium on Knowledge Acquisition and Modeling (2015). DOI:https://doi.org/10.2991/kam-15.2015.53

[3] 2011. Kalman filter - Wikipedia. Kalman filter - Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Kalman_filter

[4] Tom M. Apostle. 1975. Calculus (2nd edition ed.). Wiley.

[5] R. Mortensen. 1972. Filtering for stochastic processes with applications to guidance. IEEE Transactions on Automatic Control 17, 1 (February 1972), 184–185. DOI:https://doi.org/10.1109/tac.1972.1099917

Figures & Graphs

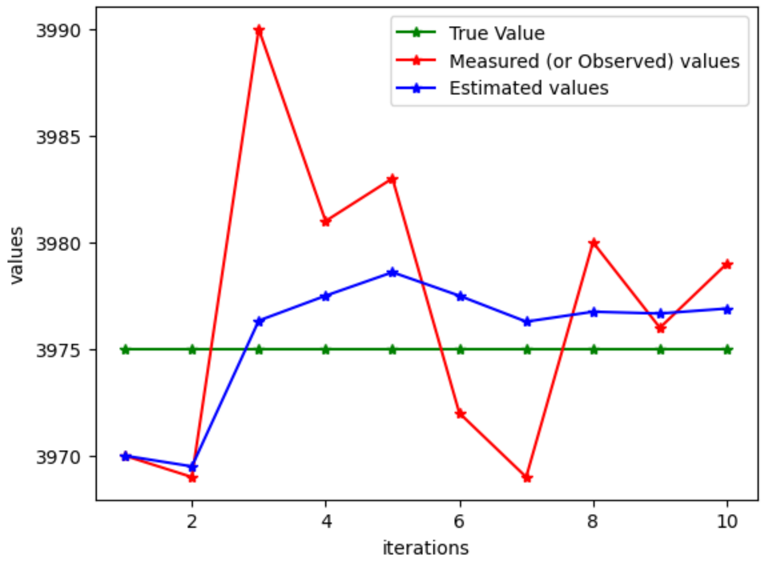

Figure 1: Estimation of the Engine weight (values are in kg)

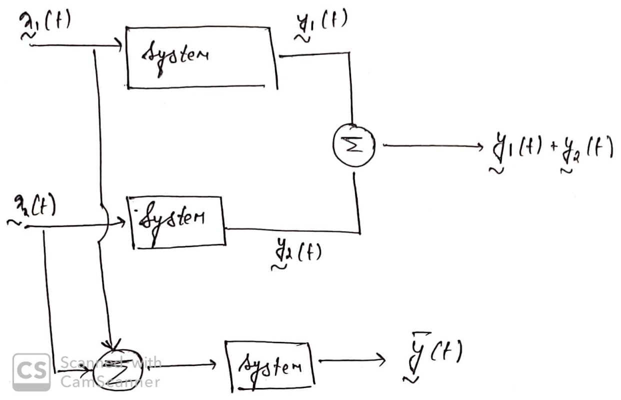

Figure 2: Law of Additivity

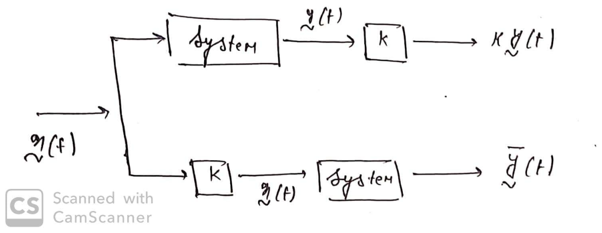

Figure 3: Law of Homogeneity

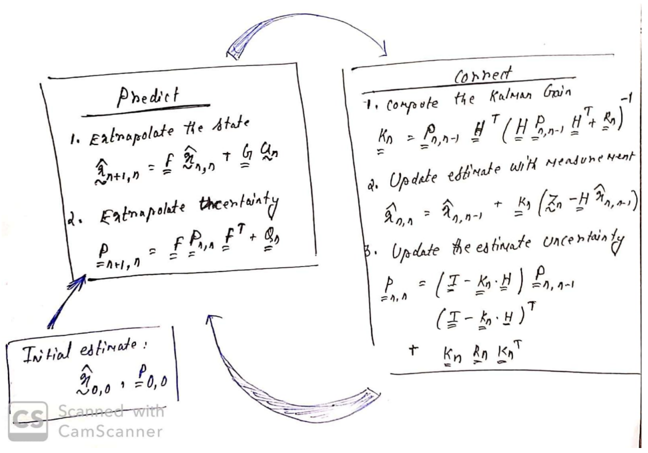
Predict
1. Extrapolate the state
$\hat{x}_{n+1,n} = F \hat{x}_{n,n} + G u_n$
2. Extrapolate uncertainty
$P_{n+1,n} = F P_{n,n} F^T + Q_n$
Correct
1. Compute the Kalman Gain
$K_n = P_{n,n-1} H^T (H P_{n,n-1} H^T + R_n)^{-1}$
2. Update estimate with measurement
$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n (z_n - H \hat{x}_{n,n-1})$
3. Update the estimate uncertainty
$P_{n,n} = (I - K_n \cdot H) P_{n,n-1} (I - K_n \cdot H)^T + K_n R_n K_n^T$
Initial estimate:
$\hat{x}_{0,0}$, $P_{0,0}$

Figure 4: Working of Linear Kalman Filter