KALMAN FILTERING

Module: Advanced Signal Processing

B00122875

Vimal Jaswal

TU Dublin - Blanchardstown Campus

Contents

Introduction

This report describes the Kalman filter, its origination and mathematical equations associated with it. I will describe its history and background and various application where kalman filter can be practically employed. I will use MATLAB for practical demonstration of its application. Then I will analyse the result obtained and the performance of filter.

Background of Kalman Filter

Kalman filter was first introduced in a Research paper named "A New Approach to Linear Filtering and Prediction Problems" by Rudolf E. Kalman in 1960. "New Results in Linear Filtering and Prediction Theory"-1961 (R. E. KALMAN ,R. S. BUCY) was another paper he published along with his co-researcher. R.E. KALMAN was born in 1930 and died in 2016. Kalman filter was invented in his research for space navigation problem. Kalman filter method utilises a series of measurements perceived over time, containing statistical noise and other inaccuracies and produces estimates of unknown variables that tend to be more precise rather than on basis of single measurement. It was first time Implemented on the Apollo Project in 1961 - space navigation mission to moon.[1]

It contains set of mathematical equations. It is based on Recursive data Processing Algorithm which means there is need to store all previous measurements and reprocess all data each time. Kaman filter uses Optimal estimation method i.e., minimum mean square error in results between actual parameters and predicted parameters.

Kalman filter is very Versatile and involves four techniques.

- Estimation
- Filtering
- Prediction
- Fusion

Following are some advantages of Kalman filter:

- Optimal estimator technique for reducing errors in measurement.
- Kalman filter Provides a precise measurement for noises present in the system.
- Kalman filter method handles missing data in measurements.
- Kalman filtering also handles the large uncertainty of the initialization phase.
- Kalman filter is useful in sensor fusion as it fuses information from multiple-sensors.

Kalman Filter Concept

Let us consider a Hydrologic model that predicts water level in a hydro-dam every hour. We know that model is not perfect as there are other factors such as noise and error associated to it. We can use this model to measure the water level but cannot rely completely on this for perfect result. On the other hand, we can employ someone to manually take readings from the measuring scale. The person readings could also include human mistakes and thus cannot be fully trusted for actual results for perfection. Therefore, we can combine both case results of water level from model and person's measurements to get a "fused" (average) and better estimation of correct water level. This technique gives us better result than a single model can provide. In this manner we can say that the Kalman filter is based on fusion process which is combination of prediction process and the measurement process. Both processes can be combined and operated in a recursive manner to achieve optimal Kalman filtering process.

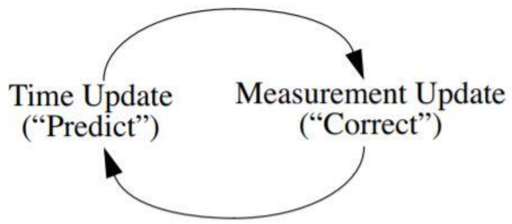
Applications of Kalman Filter

Kalman Filter is a sensor fusion and data fusion algorithm widely used for various applications such as:

- State Estimation and multiple Sensors Fusion
- Wireless sensor network (WSN), Localization, Tracking
- Guiding Systems, Navigation, GPS
- Autopilots
- Radar Tracking Systems
- Orbit tracking, trajectory tracking
- Simultaneous Localization and Mapping (SLAM)
- State estimators for Unmanned Aerial Vehicles (UAV)
- Computer Graphics - 3D Modelling feature estimation

Discrete Kalman filter (Predictor- Corrector)

The Kalman filter estimates a phenomenon or process by feedback control: the filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. This results in Time update equations and measurement update equations formed during the process. This is termed as discrete Kalman filtering method and forms predictor-corrector structure.[2]



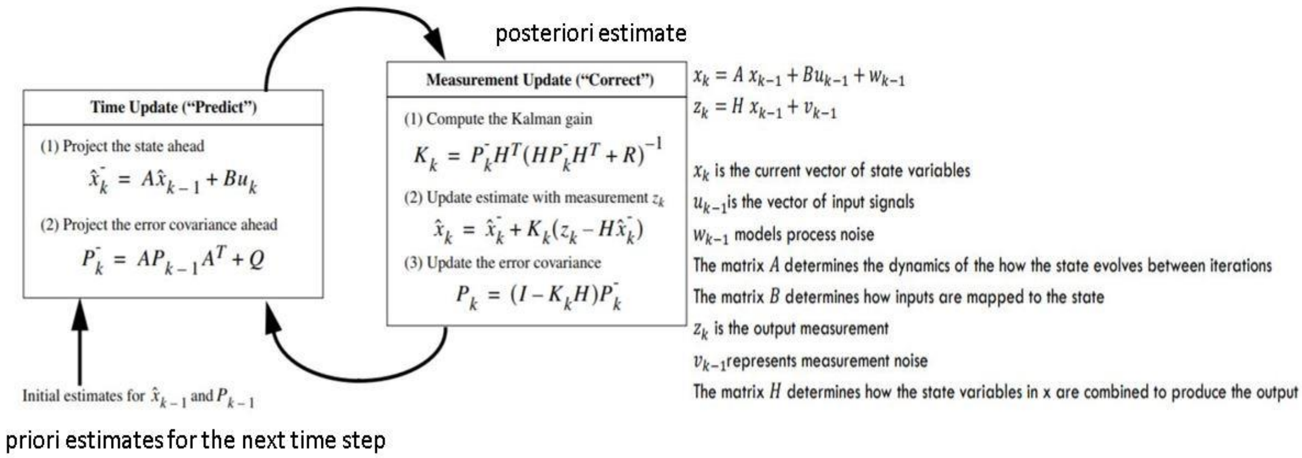
Q – denotes the process noise covariance R- denotes measurement noise covariance

- (Co- variance noise matrices might change with each time step/measurement/iteration)

P- covariance matrix of error in state estimate x

K- Kalman gain depend upon P and Q co-variances and H matrices

K changes in every iteration and determine weight for each prediction



Kalman Filter Software Tools

1. Visual Kalman Filter

- Kalman filtering designer that provides a visual method in windows to estimate the state of a process or removes noise from series of data[4]

- Support for discrete system and continuous system

- Easy, lite and handy to use

2. MATLAB (Built-in features, EKF/UKF Toolbox for MATLAB V1.3)

Functions available in Matlab are following:

- **Kalman**

Syntax: (Kalman filter design, Kalman estimator)

[kest,L,P]=kalman(sys,Qn,Rn,Nn)
[kest,L,P]=kalman(sys,Qn,Rn,Nn,sensors,known)
[kest,L,P,M,Z] = kalman(sys,Qn,Rn,...,type)

- **Kalmd**

[kest,L,P,M,Z] = kalmd(sys,Qn,Rn,Ts)

- **configureKalmanFilter (**Kalman filter function for object tracking)

kalmanFilter=configureKalmanFilter(MotionModel,InitialLocation,InitialEstimateError,MotionNoise,MeasurementNoise)

3. GNU OCTAVE (Autocovariance Least-Squares (ALS) Package tool also works with MATLAB

There are many other tools available based on computer languages such as java and python which are also useful for Kalman filter modelling.

1D-Kalman-Filter

To understand the working of the Kalman Filter, an example of a linear system was taken; A vehicle is moving on a straight road with a constant velocity (2m/s). It also has a GPS on board that gives it noisy readings. We are given an estimate of its initial position (we assume one, if it isn't given) and at EVERY time step (epoch) we try to obtain the best estimate of its position by fusing together GPS readings and constant velocity model.

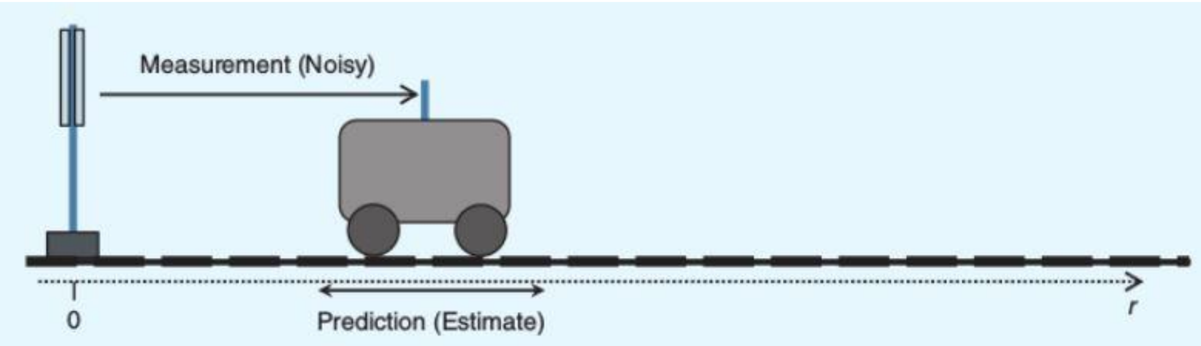


Fig1. The vehicle moves on a straight path, measuring its location with respect to a pole on the left side. It moves with a constant velocity.

System Model

For a Kalman filter based state estimator, the system must conform to a certain model. So, if system model conforms to model mentioned herein, then we can use a Kalman Filter to estimate the state of the system.[3]

Motion Model

x_new = A*x_old + B*u + w
x_new : current state
x_old : previous state
A : state transition matrix
B : control input matrix
w : process noise (from a 0 mean normal distribution with covariance Q)

Sensor Model

z = H*x + v

z : sensor measurement

H : sensor transformation matrix

x : current state

v : measurement noise (from a 0 mean normal distribution with covariance R)

Model Description

The state vector 'x' contains the state of the system i.e. the parameters that uniquely describe the current position of the system. In this case, the state vector is a single dimensional vector containing the location of the vehicle. It can also be a N dimensional vector containing position in different axes, velocity in different axes, temperature, state of sensors etc.
A is the state transition matrix, it applies the effect of each parameter of the previous state on the next state.
B is the control input matrix that applies the effect of the control signal given to the system onto the next state. So, in this system, the current position is based on the previous position added to the velocity*time. Mathematically,
x_new = 1*x_old + timedifference*velocity

- In the equation given above, A = 1 and B = time difference.
- No system is perfect, given the previous position and the velocity, the new location will not correspond to the equation given above. There will be noise in the system, this noise is modelled as a Gaussian distribution having mean = 0 and a certain standard deviation given by covariance matrix Q (explained later). In simple words, it is the difference between the ideal new location and the actual new location.

x_new - (A*x_old + B*u) = noise = w

The 'z' vector contains the sensor measurements given by the sensors. The Kalman filter is based on a Hidden Markov Model, meaning that the current 'z' depends ONLY on current state, and not any of the previous states as is evident in the sensor model equation.

The 'H' matrix maps the state vector parameters 'x' to the sensor measurements. In simple words, it tells us the sensor measurement that we *should* get given the current state 'x'.

Similar to 'w', 'v' is also a parameter representing the noise in sensor measurements. It is also taken from a 0 mean Gaussian distribution whose variance is taken from covariance matrix R.

Kalman Filter Equations

Initialization

We assume that we have a good knowledge of the vehicle's initial position and it can be represented by a gaussian whose mean is the initial known position and a covariance matrix having small values. This is shown in the image below.

Fig2. The state is represented by a gaussian as shown.

Prediction Step

The vehicle moves forward with a constant velocity 'u'. We can now have a prediction of the next state from the prediction equations. This prediction is represented by a gaussian having a mean and a variance. As we can see this variance is more than the previous variance, thus showing that we are more uncertain about its position.

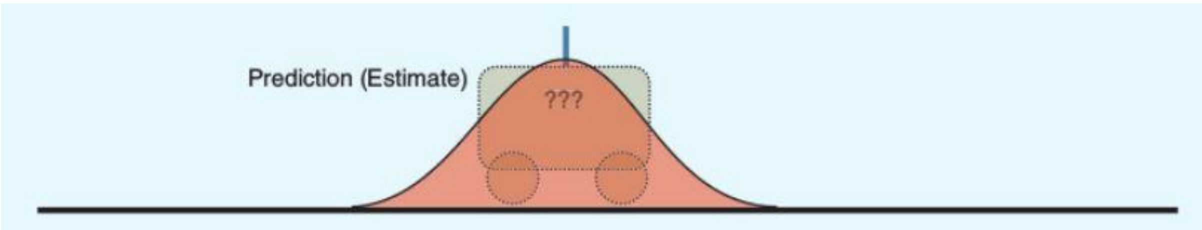


Fig3. When the vehicle moves, it becomes more uncertain about its position due to the control being noisy. Hence, the gaussian expands.

x = F*x + B*u
P = F*P*F' + Q
P : state covariance matrix
Q : process noise covariance matrix (explained previously)

- Apart from P and Q the other variables have been explained previously.
- The 'Q' is the process noise covariance matrix. The diagonal elements contain the variance(std_dev*std_dev) of each respective variable in the state vector 'x'. So if the state vector has 2 columns containin the x and y co-ordinates, then Q is a 2x2 matrix whose diagonals contain the variance of each of those variables. The non-diagonal variables are usually set to 0 except in the case of special circumstances. The variances are calculated from the noise variable 'w', the variance of these values is noted while observing the system.

- 'P' is the state covariance matrix, like 'Q', it models uncertainty in the system. It models uncertainty of the state vector 'x'. Each of the diagonal elements contain the variance (uncertainty in position) of each of those respective state variables in the state vector. For initialization for this matrix, if the state variable's initial location is known to a high degree, the corresponding diagonal element in P is a small. Vice-versa in case the state variable's initial location is not known well.

- In the prediction step, if we look at the second equation, we see that the value of P is increasing (due to the addition), this goes to show that in the prediction step, when we do not have any measurement and we only have control command 'u', the next state will be known with lesser certainty. The opposite happens in the Correction step.

Correction Step

In this step, the vehicle makes a measurement of its position using its onboard location sensor i.e it finds its distance from the pole using a sensor. This measurement is noisy and not exact. This measurement itself is represented by a gaussian having a mean and covariance. The value of the covariance depends on the accuracy of the sensor: If the sensor is more accurate the covariance value will be small, else it will be large.

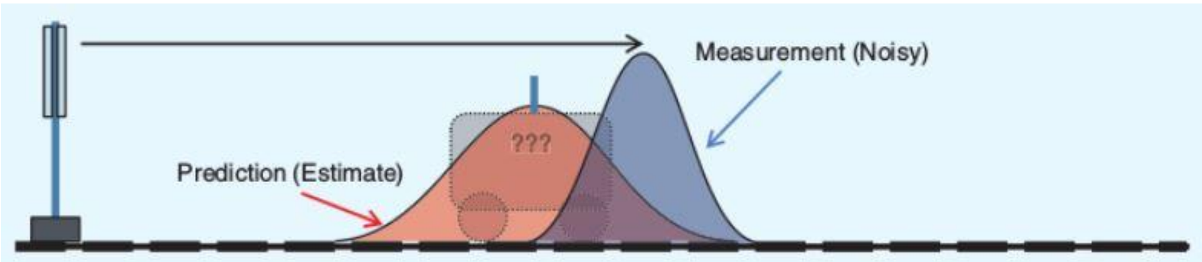


Fig4. The measurement is represented by a blue gaussian having a covariance smaller than the predicted state.

x = x_old + K*(z = H*x_old)
P = P_old - K*H*P_old
K = P*H' * (H*P*H' + R)^-1
K : Kalman Gain
R : Measurement noise covariance matrix

- 'K' is called the Kalman Gain. It is calculated from state covariance matrix and the measurement covariance matrix. If the measurement noise is more, then the value of K will be less, if the measurement noise is more, then its value will be less.

- In the first equation for 'x', we are approximately taking a weighted average of the predicted state vector and the state vector generated from the measurement. This weighting is decided by the Kalman gain.

- In the second equation for 'P', we see that the value of P is decreasing (subtraction), this is because we believe that the sensor is more accurate and our uncertainty about the vehicle's position decreases.

- If we look at it from an analytical perspective, we have two gaussians. They represent state vector and measured state. If we multiply these 2 gaussians we get another gaussian which is actually the best estimate of the position of the vehicle.

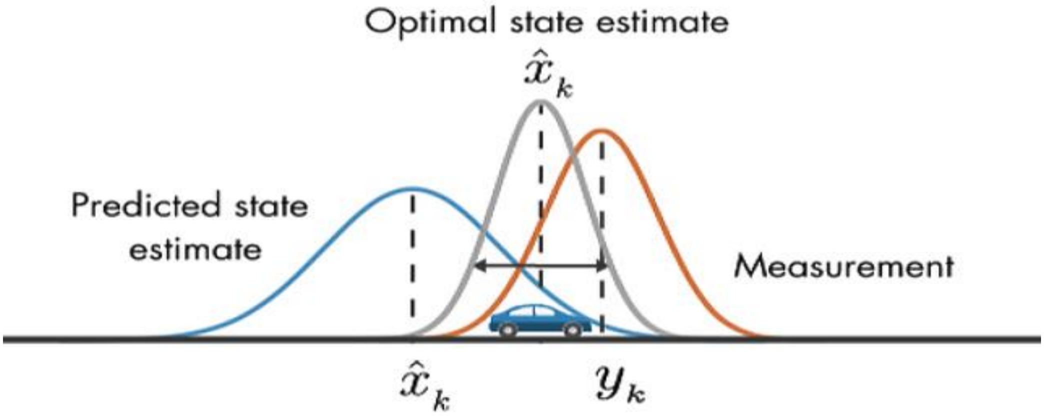
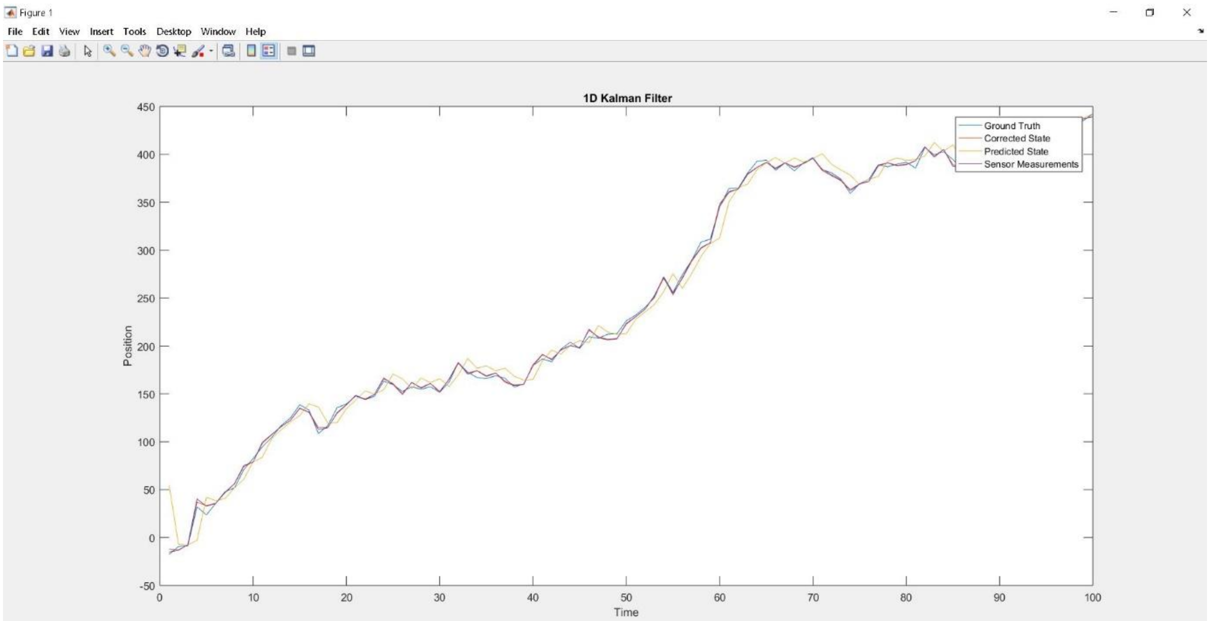


Fig5. We multiply the two gaussians to have the best estimate of the vehicle's position.

The following output is obtained and the MATLAB code is given in appendix for justifying this.



State Space Model and Kalman Filter

- State space model to describe the behaviour of a dynamic system

  (Dynamic system are systems that change in time)

- Systems are represented as a differential equation
- Differential equation for continuous/analogue systems
- Difference equation for discrete time/digital systems

- In the state space representation, a differential equation can be represented as a system of 1st order differential equations

The general form of the equations are in pair as:

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$
$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

u represents input vector,
x represents state vector,
y represents the output vector,
A represents the time invariant dynamic matrix,
B represents the time invariant input matrix,
C represents the time invariant measurement matrix.

State-of-Charge for Battery Using Kalman Filter

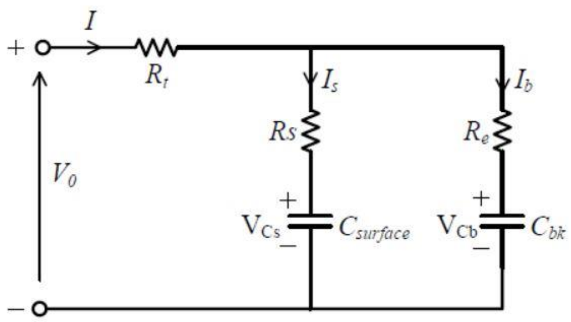
At this stage, the Kalman filter is applied on a battery circuit model. According to network theory the output state of the model can be represented in the form of linear differential equation. The state model is continuous and transformed into discrete to obtained own results.

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t)$$
$$\mathbf{y}(t) = C(t)\mathbf{x}(t) + D(t)\mathbf{u}(t)$$

The Objective is to study a state-space model containing state variables *VCb, VCs* and *V*0. State variables are mathematical description of the "state" of a dynamic system. In practice, the state of a system is used to determine its future behaviour. [5]

Models that contain first-order difference equations in pairs are in state-variable form.

C_{bk}	$C_{surface}$	R_e	R_s	R_t
88372.83 F	82.11 F	0.00375Ω	0.00375Ω	0.002745Ω



$V_0 = IR_t + I_bR_e + V_{Cb}$,

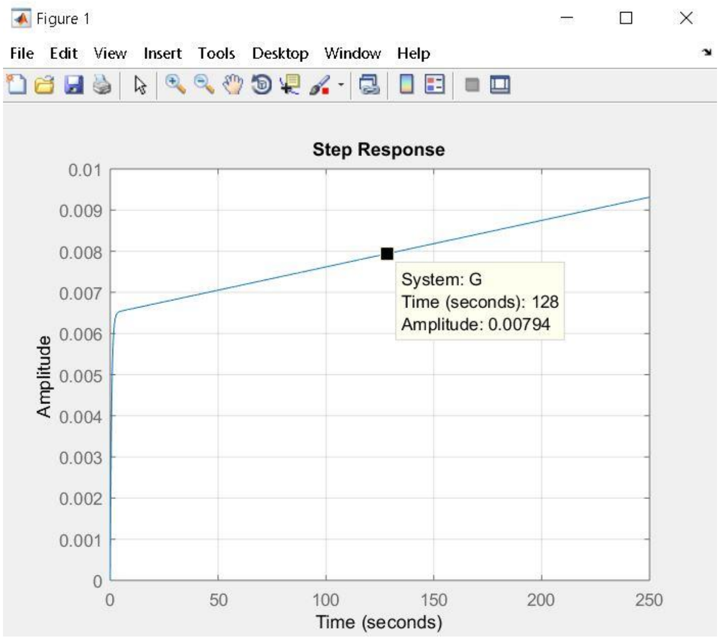
$V0 = IRt + IbRs + VCs$.

$A = \frac{1}{C_{bk}(R_e+R_s)}$ and $B = \frac{1}{C_{surface}(R_e+R_s)}$

This can be represented as:

$\dot{V}Cb = A \cdot IRs + A \cdot VCs - A \cdot VCb$,
$\dot{V}Cs = B \cdot IRe - B \cdot VCs + B \cdot VCb$,

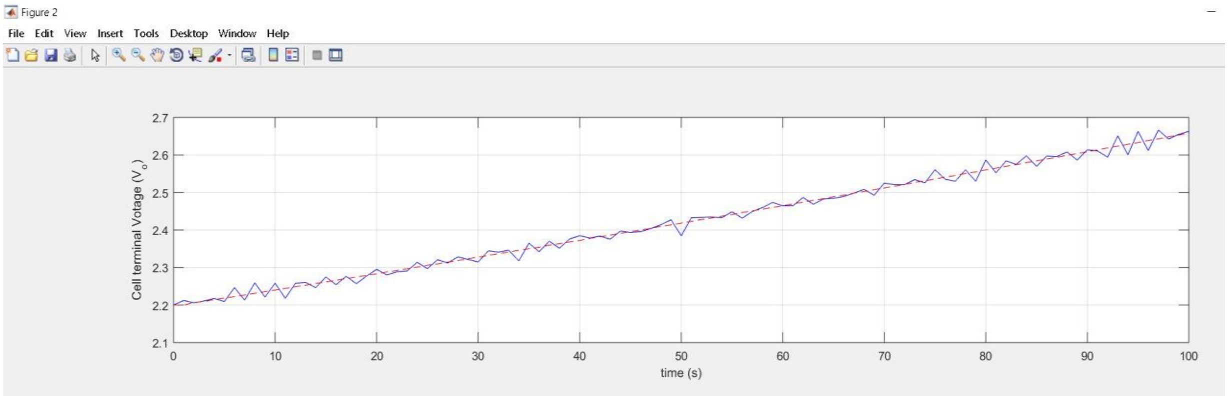
$$\begin{bmatrix} \dot{V}_{Cb} \\ \dot{V}_{Cs} \end{bmatrix} = \begin{bmatrix} -A & A \\ B & -B \end{bmatrix} \begin{bmatrix} V_{Cb} \\ V_{Cs} \end{bmatrix} + \begin{bmatrix} A \cdot R_s \\ B \cdot R_e \end{bmatrix} I.$$

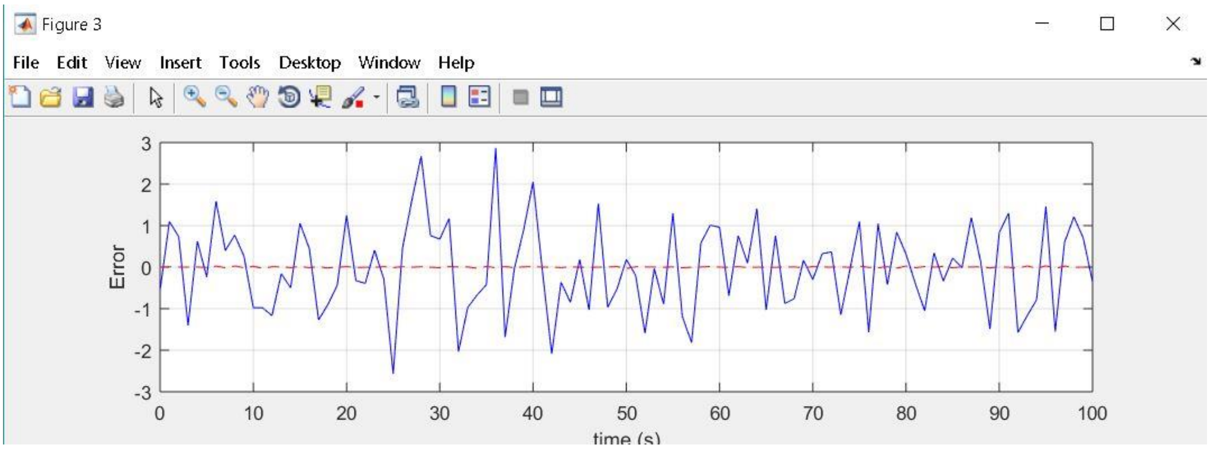


The figure of step response represents the open circuit terminal voltage V_0. As shown in graph the open circuit terminal voltage V0 increases linearly during charging operation after transient behaviour of some seconds.

The MATLAB code for these results are available in Appendix is given for proof of the results obtained from this model. Notice that Q and R process and measurement noise values are set as one. The results obtained are following:

The following figure shows the charging state and behaviour of battery. It shows that the battery charge between 2.2Volts - 2.7Volts within 100 seconds.





This is the transfer function of the above model.

```
G =

    0.01054 s^2 + 0.01714 s + 2.981e-05
  ------------------------------------
  s^3 + 3.248 s^2 + 2.637 s - 1.144e-18

Continuous-time transfer function.
```

The values of Measurement covariance and error covariance are obtained as below:

```
MeasErrCov =

   1.113849571615255


EstErrCov =

     1.771708966131576e-04

>>
```

Conclusion

I have learned about the concept of Kalman filtering background and its growing use in various sensor fusion and navigation and positioning and numerous applications related to internet of things. I have derived various results using MATLAB tool for practising One dimensional model of Kalman filter and its further application in state space model. Through this implementation the results estimated are elaborated in the report which clarifies the understanding of reduction in output error in case of any kind of application where Kalman filter applied and where data and fusion in the system is present.

References

[1] Welch, G and Bishop, G. 2001. “An introduction to the Kalman Filter”, [Online]. Available: http://www.cs.unc.edu/~welch/kalman/ [Accessed: 03- Apr- 2019].

[2]M. Hayes, *Statistical digital signal processing and modeling*. New Delhi: Wiley, 2014.

[3]GitHub.(2019).*akshaychawla/1D-Kalman-Filter*. [online] Available at: https://github.com/akshaychawla/1D-Kalman-Filter [Accessed 6 May 2019].

[4]"Visual Kalman Filter - Kalman filter designer for Windows", *Luckhan.com*, 2019. [Online]. Available: http://www.luckhan.com/kalman-filter-design.htm. [Accessed: 03- Apr- 2019].

[5] Ting, T., Man, K., Lim, E. and Leach, M. (2014). Tuning of Kalman Filter Parameters via Genetic Algorithm for State-of-Charge Estimation in Battery Management System. The Scientific World Journal, 2014, pp.1-11.

APPENDIX

MATLAB Code for One dimensional Kalman Filter

Editor - C:\Users\vimal\Desktop\oned.m

oned.m

```
%1d a Kalman Filter
X = 0;
dt = 1;
u = 5;              %speed/control = 5m/s
n = randn();
v = randn();

%Kalman Filter variables
x = 50;       %state vector
A = 1;        %state transition matrix
B = 1;        %control input matrix
P = 100;      %std_dev*std_dev = 10*10
Q = 100;      %process noise covariance matrix
R = 9;        %measurement noise covariance matrix
H = 1;

% Storing calculated values in these vectors for plotting
XX = zeros(1,100);
tt = zeros(1,100);
xx_predicted = zeros(1,100);
xx = zeros(1,100);
PP = zeros(1,100);
yy = zeros(1,100);

```

Editor - C:\Users\vimal\Desktop\oned.m

oned.m

```
PP = zeros(1,100);
yy = zeros(1,100);

for t=1:1:100
    %simulating the System
     n = sqrt(Q) * randn(); %random noise
     X = X + u*dt + n; % New Current State
     v = sqrt(R) * randn() ;
     y = H*X + v; % Measurements

    %Prediction Step
     x_predicted = A*x + B*u;    %predicting the new state (mean)
     P = A * P * A' + Q;          %predicting the new uncertainity (covariance)

    %Correction Step
     e = H*x_predicted;  %expectation: predicted measurement from the o/p
     E = H*P*H';         %Covariance of ^ expectation
     z = y - e;          %innovation: diff between the expectation and real sensor measurement
     Z = R + E;          %Covariance of ^ - sum of uncertainities of expectation and real measur
     K = P*H' * Z^-1;

     x = x_predicted + K*z;  %final corrected state
     P = P - K * H* P;       %uncertainity in corrected state

```

```
    %Saving the outputs
     xx_predicted(t) = x_predicted;
     xx(t) = x;
     PP(t) = P;
     XX(t) = X;
     tt(t) = t;
     yy(t) = y;
end

plot(tt,XX,tt,xx,tt,xx_predicted,tt,yy)
title(' 1D Kalman Filter')
legend('Ground Truth','Corrected State','Predicted State','Sensor Measurements')
xlabel('Time')
ylabel('Position')
```

MATLAB Code for Battery Charging state space models

Editor - C:\Users\vimal\Desktop\battery.m

battery.m

```
format long;
%Value for resistors and capacitors
Cs=82.11;
Cb=88372.83;
Re=0.00375;
Rs=0.00375;
Rt=0.002745;

a=1/(Cb*(Re+Rs));
b=1/(Cs*(Re+Rs));
d=(Re*Rs)/(Re+Rs);

%State variable matrices
A=[-a a 0 ; b -b 0 ; (-a+b) 0 (a-b)]
B=[a*Re; b*Re; a*(0.5*Rs-Rt-d)+ b*(0.5*Re+Rt+d)]
C=[0 0 1 ]
D=[0]

%Transfer function
figure(1); %Figure 1
[num, den]= ss2tf(A,B,C,D,1)
G=tf(num,den)
step(G),grid;

% For Kalman filter:
% Identity matrix + diagonal element
```

Editor - C:\Users\vimal\Desktop\battery.m

battery.m

```
% For Kalman filter:
% Identity matrix + diagonal element
A=[1-a a 0 ; b 1-b 0 ; (-a+b) 0 1+(a-b)]
B=[a*Re; b*Re; a* (0.5*Rs-Rt-d)+ b*(0.5*Re+Rt+d)]
C=[0 0 1 ]

Tc=1;
A=A*Tc;
B=B*Tc;
C=C;

% Sample time=-1 for discrete model
Plant = ss(A,[B B],C,0,-1,'inputname',{'u' 'w'},'outputname','y');
Q=1;
R=1;
[kalmf,L,P,M] = kalman(Plant,Q,R);
kalmf = kalmf(1,:);
kalmf
a = A;
b = [B B 0*B];
c = [C;C];
d = [0 0 0;0 0 1];
P = ss(a,b,c,d,-1,'inputname',{'u' 'w' 'v'},'outputname',{'y' 'yv'});

% Parallel connection of outputs ye and y
sys = parallel(P,kalmf,1,1,[],[])
```

Editor - C:\Users\vimal\Desktop\battery.m

battery.m

```
% Close loop around input #4 and output #2
SimModel = feedback(sys,1,4,2,1)

% Delete yv from I/O list
SimModel = SimModel([1 3],[1 2 3])
SimModel.inputname

t =[0:100]';
u= t/1.53;  % Current for discharge

n = length(t)
randn('seed',0);
w = sqrt(Q)*randn(n,1);
v = sqrt(R)*randn(n,1);
[out,x]= lsim(SimModel,[w,v,u]);

y0= 2.2 %This is initial terminal voltage
y = out(:,1)+y0; % true response,
ye = out(:,2)+ y0; % filtered response
yv = y + v; % measured response

figure(2); %Figure 2
subplot(211), plot(t,y,'b-', t,ye,'r--'), grid on;
xlabel('time (s)'),
ylabel('Cell terminal Votage (V_o)')

```

```

%Kalman filter response
figure(3); %Figure 3
plot(t,y-yv,'b-', t,y-ye, 'r--'), grid on;
xlabel('time (s)'), ylabel('Error')

%Calculate Errors
MeasErr = y-yv; %Measurement error
MeasErrCov= sum(MeasErr.*MeasErr)/length(MeasErr);
EstErr = y-ye; %Estimated error
EstErrCov = sum(EstErr.*EstErr)/length(EstErr);
%Display onto screen
MeasErrCov %Measurement error
EstErrCov %Estimated error

```