# KALMAN FILTER

**Introduction-**

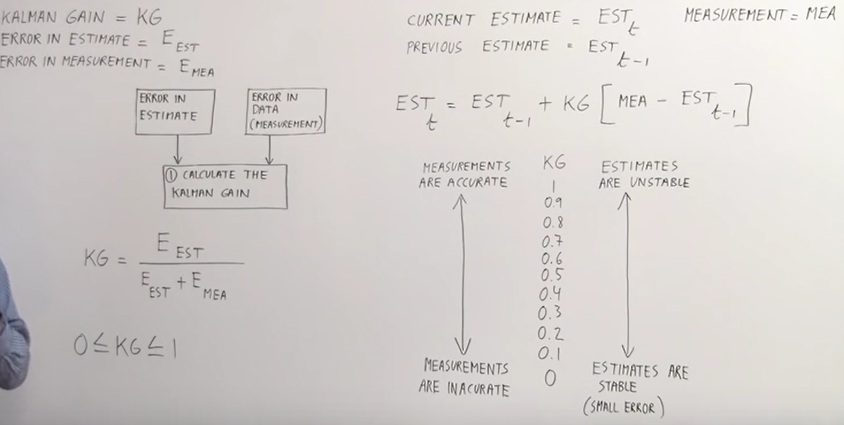
**Kalman filter**, also known as **linear quadratic estimation** (**LQE**), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone, by using Bayesian inference and estimating a joint probability distribution over the variables for each timeframe. Kalman filter is a very powerful algorithm to not only remove error in the measurement from sensors but also predict the right value.

Kalman Filter can also be applied to fuse the readings of sensors for example we can fuse the readings of accelerometer and gyroscope in the navigation of a robot. In case of predicting more than 1 parameter (multi dimensional), matrices are used. But for estimating 1 parameter (1-Dimensional) equations can be reduced to a simple iterative numerical process.

I have integrated Arduino and python using pyserial to remove glitches in the reading of ultrasonic sensor and predict the correct distance of an object.

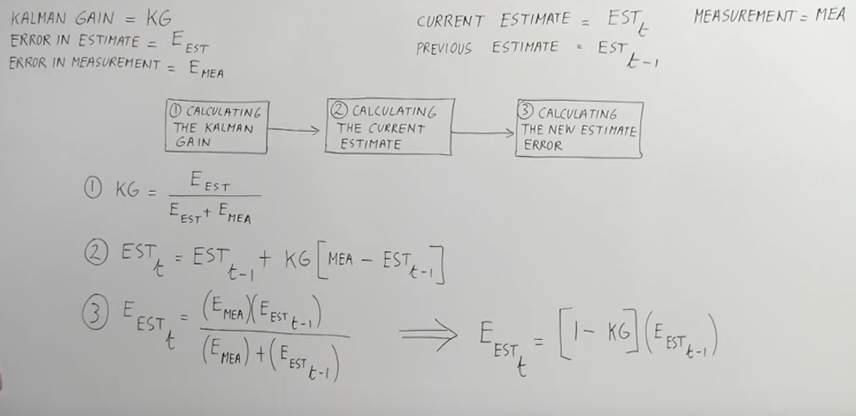
A Closer look to the 1-D Equations used in Kalman Filter Algorithm.

First we assume an estimate and error in estimate then apply the below equations to converge the estimate to actual value. Number of iterations depends on the initial value we choose.

****

Kalman Gain is the ratio of Error in Estimate to Error in Estimate + Error in Measurement. The value of 0 < = KG < = 1 determines that which of Measurement and Estimate is stable. If KG is near 0 then Estimate is stable, if KG is near 1 then measurement is pretty accurate.

**EQUATIONS-**

****

A simple code to access the reading coming from Serial Monitor using Pyserial & then implementing Kalman Filter Algorithm in Python to predict the correct distance of an object using Arduino and Ultrasonic Sensor is as follows.

**PYTHON CODE-**

import serial

import sys

from visual import \*

a=serial.Serial('com5', 9600) # a is an object for serial communication

def K\_Gain(E\_est,E\_mea):

k=(E\_est)/(E\_est+E\_mea) # estimating kalman gain(measure of what % of estimate and measurement we have to take)

return(k)

def Estimate(mea,k,Prev\_Estimate):

z=0.0

z=(Prev\_Estimate+k\*(mea-Prev\_Estimate)) # calculating estimate based upon the previous estimate

return (z)

def E\_estimate(K,Prev\_Error):

error=0.0

error=(Prev\_Error\*(1-K)) # calculating the error in the estimate

return(error)

while(1==1):

if(a.inWaiting()>0):

s=a.readline() # getting measurement from sensor

mea=float(s)

E\_est=10.0 # my guess is 10cm(can be any value)

E\_mea=2.0 # max error depending upon the sensor

estimate=100.0 # my guess is 100cm(can be any value)

while(E\_est>0.01): # my precion of estimate upto .01cm

k=K\_Gain(E\_est,E\_mea)

estimate=Estimate(mea,k,estimate)

E\_est=E\_estimate(k,E\_est)

print(estimate,mea)

**ARDUINO CODE-**

int trigPin=4;

int echoPin=2;

float Time;

float speedOfSound=346.0;

double Distance=0.0;

void setup() {

Serial.begin(9600);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2000);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

Time = pulseIn(echoPin, HIGH);

Distance=(Time)\*(speedOfSound)/(10000);

Serial.println(Distance);

delay(100);

}

## Links-

<http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures>

<http://www.sage.unsw.edu.au/wang/jgps/v2n1/v2n1pF.pdf>

<https://en.wikipedia.org/wiki/Kalman_filter>

## <https://www.youtube.com/watch?v=CaCcOwJPytQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT>