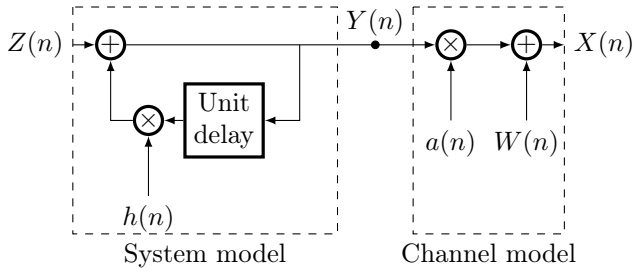
# Kalman Filter

Søren Bøgeskov Nørgaard

Started: October 30, 2014
Last update: November 3, 2014

## Abstract

Basic formulas for implementing a Kalman filter. Based on course note from the course in Stochastic Processes on Aalborg University, 2014.




## 1 Variables

$X(n)$ Measured data with noise.

$\hat{Y}(n)$ Estimate of data, based on $\hat{X(n)}$.

$R(n)$ Estimation error.

$h(n)$ Feedback coefficients of system.

$a(n)$ Attenuation factor of the channel.

$\sigma_Z^2$ Variance of noise added in system.

$\sigma_W^2$ Variance of noise added in channel.

The notation $X(n|n-1)$ means the $n$th value based on the previous value, i.e., $(n-1)$. One could basically write $X(n)$ instead and just overwrite it in the updating step.

## 2 Initialization

To get started, the estimate and error must be initialized. Some fitting estimates could be

$$\hat{Y}(0|0) = \mu_{Y(0)} \tag{1}$$
$$R(0|0) = \sigma^2_{Y(0)} \tag{2}$$

or some statistical estimates of these. The Kalman filter should correct these in a few iterations.

## 3 Prediction Step

First estimate of the output is done by trying to predict the current output, $Y(n)$, based on the previous estimate. The current error, $R(n)$, is calculated as well.

$$\hat{Y}(n|n-1) = h(n)\hat{Y}(n-1|n-1) \tag{3}$$
$$R(n|n-1) = h^2(n)R(n-1|n-1) + \sigma_Z^2(n) \tag{4}$$
$$\hat{X}(n|n-1) = a(n)\hat{Y}(n|n-1) \tag{5}$$

## 4 Updating Step

After having tried to predict the output based on all previous output, the output is corrected with knowledge of the *current* measured input.

$$\begin{aligned}\hat{Y}(n|n) &= \hat{Y}(n|n-1) \\ &\quad + b(n)[X(n) - \hat{X}(n|n-1)]\end{aligned} \tag{6}$$
$$R(n|n) = [1 - b(n)a(n)]R(n|n-1) \tag{7}$$

where

$$b(n) = \frac{a(n)R(n|n-1)}{a^2(n)R(n|n-1) + \sigma_W^2(n)} \tag{8}$$

is known as the *Kalman gain*.

## 5 Example Implementation

```
from numpy import *
from matplotlib.pyplot import *
rcParams['font.family'] = "Times New Roman"
rcParams['font.size'] = "10"

#   Variables
h = 0.95
sig2Z = 0.1
sig2W = 1
sig2Y = sig2Z/(1-h**2)
a = 1

#   Generate noise, data, and
#   noisy data measurement
N = 500
Z = sig2Z * random.randn(N)
W = sig2W * random.randn(N)

Y = zeros(N)
Y[0] = sig2Y * random.randn(1)
for n in range(1, N):
    Y[n] = h*Y[n-1] + Z[n]

X = Y+W  # noisy measurement

#   Estimate Y from X using Kalman filter
Yhat = zeros(N)
Xhat = zeros(N)
R = zeros(N)

#   Initialization
Yhat[0] = 0
R[0] = sig2Y

for n in range(1, N):
    #   Prediction -- based on old estimate
    Yhat[n] = h*Yhat[n-1]
    R[n] = h**2 * R[n-1] + sig2Z
    Xhat[n] = a*Yhat[n]

    #   Updating -- overwriting prediction
    b = a*R[n]/(a*R[n] + sig2W)
    Yhat[n] = Yhat[n] + b*(X[n] - Xhat[n])
    R[n] = (1 - b*a)*R[n]

#   Plot
figure(figsize=(3.5,3.5))
plot(X, color='orange', alpha=0.5, label='Measuement')
plot(Yhat, color='red', label='Actual')
plot(Y, color='black', label='Kalman estimate')
xlabel('Sample')
ylabel('Amplitude')
legend(prop={'size': 8})
grid(True)
tight_layout()
savefig('example.pdf')
```

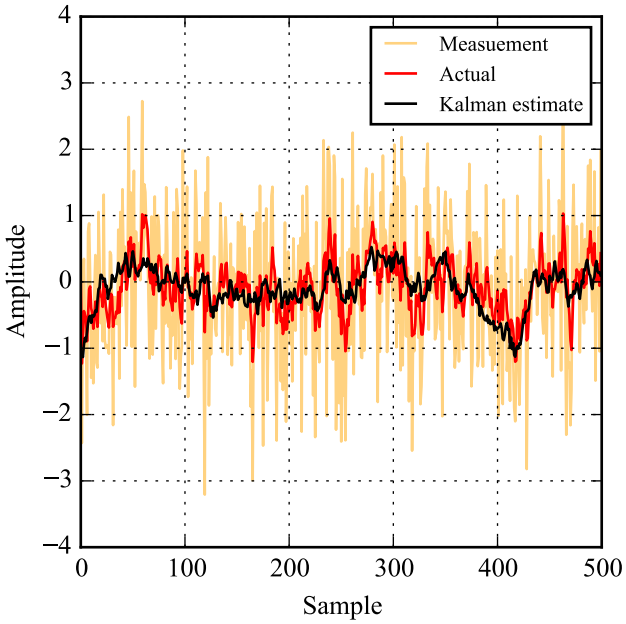



Figure 1: Output of example implementation.