

The *Kalman filter* is a very powerful algorithm to optimally include *uncertain information* from a *dynamically changing system* to come up with the best educated guess about the *current state of the system*.

Applications include (car) navigation and stock forecasting. If you want to understand how a Kalman filter works and build a toy example in R, read on!

Because we are dealing with uncertainty here, we need a probability distribution. A good choice often is a *Gaussian* or *normal distribution*. It is defined by two parameters, the *mean* and the *variance* (or *standard deviation* which is just the square root of the variance). These two parameters have to be updated by incoming information which is itself uncertain.

This can be interpreted as some form of *Bayesian updating*

The *update step* in R code:

update <- function(mean1, var1, mean2, var2) {

# calculates new position as multiplication of two Gaussians: # prior probability and new information (noisy measurement) new\_mean <- (var2\*mean1 + var1\*mean2) / (var1 + var2) new\_var <- 1/(1/var1 + 1/var2)

return(c(new\_mean, new\_var))

}

After calculating the position based on new information we also have to take into account the motion itself, this is done in the *prediction step*:

Here is the R implementation:

predict <- function(mean1, var1, mean2, var2) {

# Calculates new postion as sum (= convolution) of two Gaussians: # prior probability and new information (noisy movement) new\_mean <- mean1 + mean2

new\_var <- var1 + var2 return(c(new\_mean, new\_var))

}

Now, we are putting both steps together to form a cycle:

As an example let us create a trajectory along a sine wave with measurements and motion affected by noise (= uncertainty):

var\_measure <- 5 # variance measure var\_motion <- 2 # variance motion

pos <- c(0, 10000) # Starting values position and variance

## Kalman calculation set.seed(123)

pos\_real <- 10 \* sin(seq(1, 20, 0.1)) motion\_real <- diff(pos\_real

measure <- pos\_real + rnorm(length(pos\_real), 0, sqrt(var\_measure) motion <- motion\_real + rnorm(length(motion\_real), 0, sqrt(var\_motion) kalman\_update <- c()

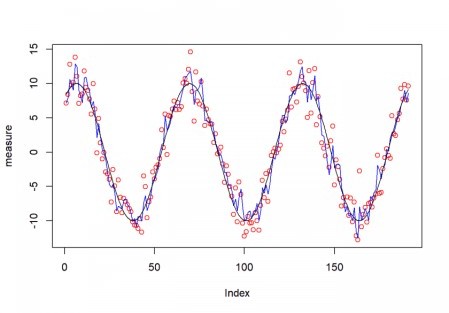
for (i in 1:length(measure)) {

pos <- update(pos[1], pos[2], measure[i], var\_measure) kalman\_update <- c(kalman\_update, pos[1])

pos <- predict(pos[1], pos[2], motion[i], var\_motion)

}

plot(measure, col = "red") lines(kalman\_update, col = "blue") lines(pos\_real, col = "black")



As you can see the resulting blue curve is much more stable than the noisy measurements (small red circles)!