Kalman Filter and its Application in Financial Markets

White Paper Article

What is a Kalman Filter

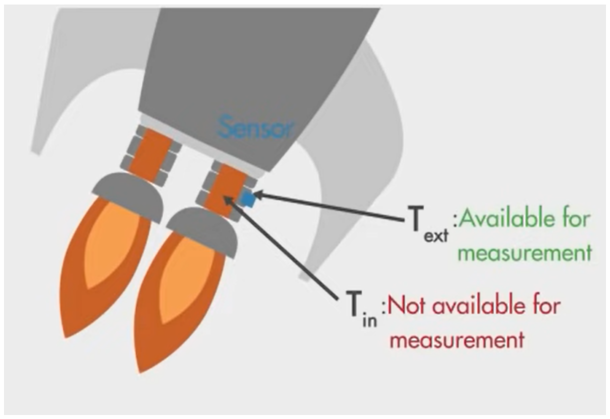
A Kalman Filter is a 2-step recursive (prediction and correction) optimal state estimation algorithm, or rephrased in simpler words, an iterative mathematical process to estimate the true/underlying/actual value of the variable(s) of interest (called state) by constantly correcting/updating the employed mathematical model (for predictions) with the newest measurements fed.

When is it used:

Kalman Filter is used when:

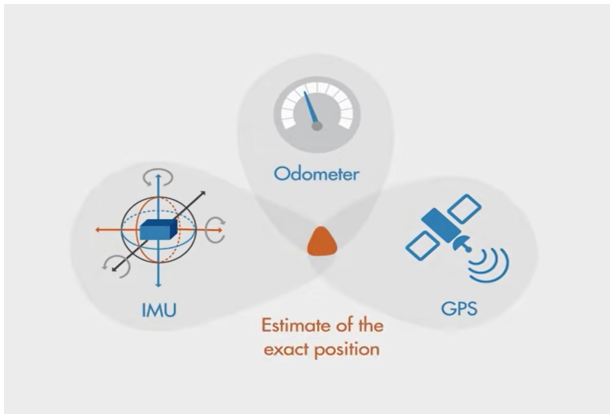
1. The variables of interest can only be measured indirectly.
2. Measurements are available from various sensors but might be subject to noise.

Getting intuition of when a Kalman filter is used

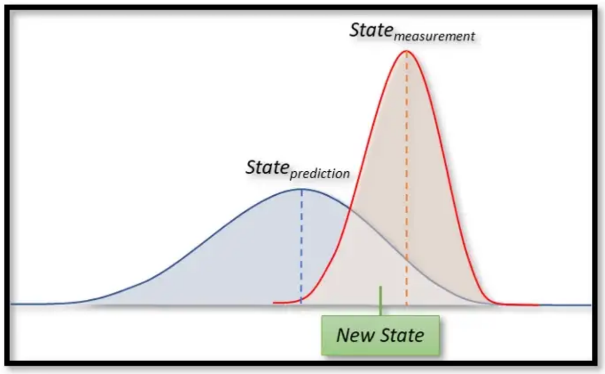

Sensor
T_{ext}:Available for measurement
T_{in}:Not available for measurement

Let's look into an example of the first use. Consider we want to monitor the temperature inside the ignition chamber of a spacecraft to be able to control the fuel supplied to it. Now any sensor placed inside the chamber will melt out because of very high temperatures, so the best we can do is to place the sensor a little away from the chamber at a cooler spot and estimate the internal temperature using a suitable mathematical model and external temperature measurements all with the help of a Kalman filter. This is how a variable of interest can be measured indirectly using a Kalman filter.

Now let's see an example of the second use. A car positioning system could consist of multiple onboard sensors like the Inertial Measurement Unit(IMU), the odometer and the GPS. Measurements from all these sensors are prone to noise be it from the system(like drift error from the IMU while calculating relative distance) or from the readings of the GPS. A Kalman filter can be used to fuse these three measurements to find an optimal estimate of the exact position of the car.



State update and intuition behind Kalman gain



Before we move on to the equations, let's understand the idea behind the weighted sum for the state update. The Kalman filter considers the errors to be gaussian distributions with 0 mean, thus the predictions for the new state given the measurement and the state model will also be gaussian distributions, but with different mean and variance, as shown in the adjoining Figure. Considering that the blue distribution is the new state based on the state model and the red distribution is the new state based on the measurement, we will update the state putting more "weight" to the distribution that is more "certain" (i.e. has smaller variance). This is the idea behind the Kalman Gain factor (***Kn***). In the example of the given figure, the uncertainty of measurement (red) is smaller, so we will give it more "weight" when updating the final state.

For a one-dimensional example, the formula for the Kalman Gain is simply:

$$K_n = \frac{Uncertainty_{prediction}}{Uncertainty_{prediction} + Uncertainty_{measurement}} = [0, 1]$$

The state estimation will be given by :

$State = K_n * State_{measurement} + (1 - K_n) * State_{prediction}$, or:
$State = State_{prediction} + K_n(State_{measurement} - State_{prediction})$

The Kalman Gain is the relation between the estimate's uncertainty and the total uncertainty (estimate + measurement uncertainty) and it tells us how much we should change the estimate by , given a measurement.

Kalman equations

Now that we understand the idea behind the Kalman filter, let's take a look at the multidimensional equations it feeds on.

The first equations we need to define are the state model and the measurement equation. We can define the state model as follows:

$x_t = Ax_{t-1} + Bu_t + w_{t-1}$
where:
$x_t \rightarrow$ state we are trying to predict $(x \in \Re^n)$
$x_{t-1} \rightarrow$ previous state
$A \rightarrow n \times n$ matrix
$u_t \rightarrow$ (optional) command input $(x \in \Re^l)$
$B \rightarrow n \times l$ matrix
$w_{t-1} \rightarrow$ process noise

Note that this equation relates the current state with the state at a previous time step plus an external (optional) control, or command, such as the steering of a car for example.

The second equation (measurement equation) relates the state with the measurement:
$z_t = Hx_t + v_t$,
where:
$z_t \rightarrow$ the current measurement $z \in \Re^m$
$H \rightarrow m \times n$ matrix
$v_t \rightarrow$ measurement noise

Usually, we cannot measure directly the state we are trying to track (like we had seen we need the temperature inside a chamber but we place the sensors outside to prevent them from melting). In these cases, the ***H*** matrix is also called the observation matrix and it is responsible for mapping from the state space to the measurement space. In the special case where we measure directly the state, ***H = I*** (identity matrix), and the following equations can be simplified accordingly.

By definition, the errors *v* and *w* are Normal (Gaussian) distributions with 0 mean and (co)variances Q and R, respectively. Q and R will be covariance instead of variance in the case of more than one variable (multidimensional) process, so we have:

$$p(w) \sim N(0, Q)$$
$$p(v) \sim N(0, R)$$

Iteration Process

The iteration process has two well-defined moments:
1 — Time update (Prediction)
2 — Measurement update (Correction)

Time update

The time update step is responsible for predicting (projecting forward in time) the current state and the (co)variance error from the previous step. We have two equations:

- The time update step is responsible for predicting (projecting forward in time) the current state and the (co)variance error from the previous step. We have two equations:
  $$\hat{x}_t = Ax_{t-1} + Bu_t$$

- Uncertainty or (co)variance prediction: this is the error intrinsic to the state model, and at each step, it is updated with the noise (co)variance (*Q*).
  $$\hat{P}_t = A * P_{t-1}A^T + Q$$

Measurement update

The measurement update will introduce the measurement to the previously predicted state. The amount of the update in the state will be given by the Kaman gain that depends on the uncertainties. We have 3 equations for this step:

- Kalman Gain: In the **Intuition** section, we have seen that the Kalman Gain for the unidimensional case is just a relationship between the uncertainties. For multidimensional, the ***Kt*** will be a ***n x m*** matrix and it will map from the measurements space to the state space:

  $$K_t = P_t H^T (H P_t H^T + R)^{-1}$$

- State update: Once we have the state prediction (from the time update) we will perform the “weighted” sum and update our state with the prediction and the measurement. In the Intuition section, we saw how it would be performed in the unidimensional scenario. For multidimensional, we have:

  $$x_t = \hat{x}_t + K_n(z_n - H\hat{x}_t)$$

- Simplified Uncertainty update (or simplified covariance update): In each iteration, the uncertainty process (***P***) will be attenuated by (***1-Kn***). In matrix form:

  $$P_t = (I - K_t H)\hat{P}_t$$

At each iteration, the algorithm will update the prediction and the prediction uncertainty, based on the state model and then introduce the new measurement to retrieve the final estimated state (also considering its uncertainty *R*). Now, let’s see how to use it in a real example.

Application of financial strategies using the Kalman filter:

Let’s have a look at how the Kalman filter can be employed to improve the implementation of financial strategies giving market momentums/entry-exit signals.

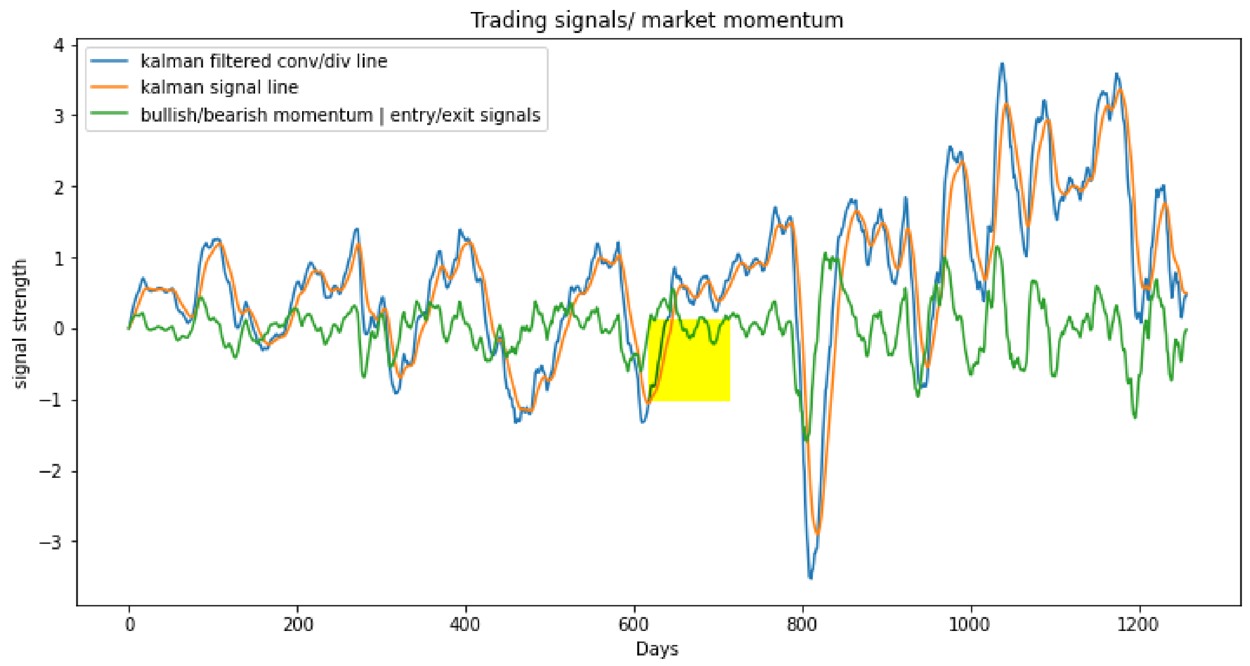
We’ll be dealing with 2 types of models while generating trading signals:

A) Self-correcting models: We essentially want a model that self-corrects itself when unforeseeable events occur. There are various models which accomplish this like the Simple Moving Average(SMA) or the Exponential Moving Average(EMA). With the establishment of these models, a trading strategy called the Moving Average Convergence Divergence (MACD) is implemented.

The following strategy takes note of the same and the smoothenings are carried out with the help of three different Kalman Filters with different transition covariances. Find the implementation of the strategy herewith:

kalman_filtered_convergence_divergence

Results generated:

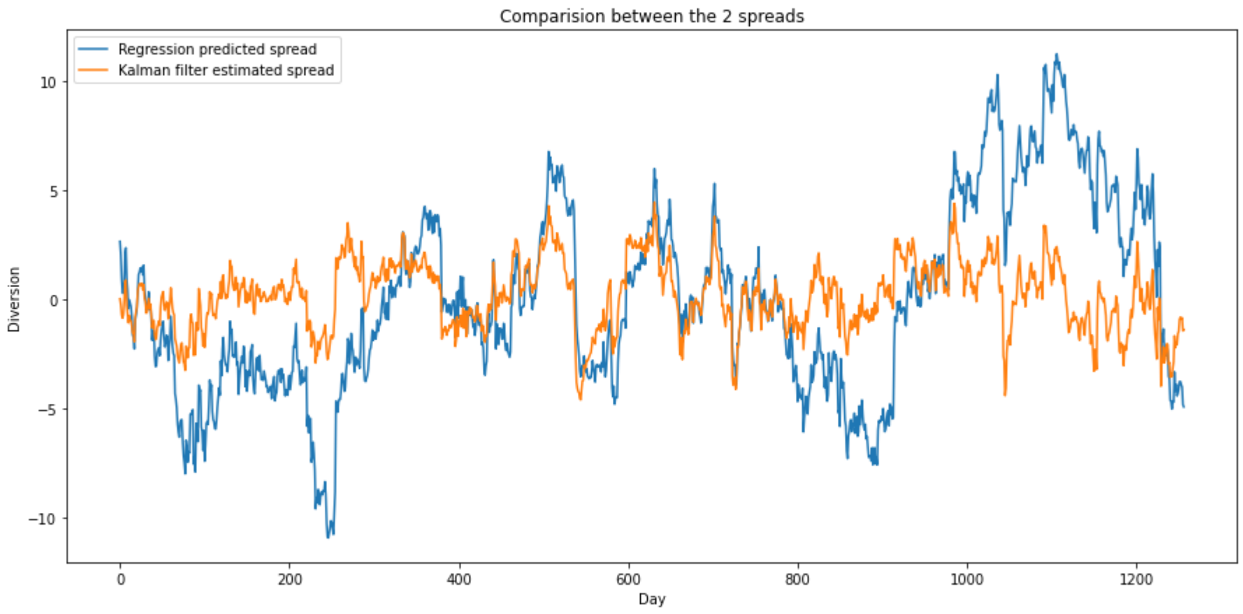


B) Mean Reversion Models: Mean reversion trading in equities tries to capitalize on extreme changes in the pricing of a particular security, assuming that it will revert to its previous state. This theory can be applied to both buying and selling, as it allows a trader to profit on unexpected upswings and to save on abnormal lows. One such mean reversion strategy is the Pairs Trading strategy.

The following strategy aims at repeatedly estimating the Hedge Ratio while trading pairs by deploying a Kalman filter for the same and thereby comparing it with that calculated using a Linear Regression. Find the implementation of the strategy herewith:

Pairs_trading_with_kalman_filter

Results Generated:


Comparision between the 2 spreads
Regression predicted spread
Kalman filter estimated spread
Diversion
Day