An Introduction to Stock Price Forecasting with Kalman Filter

Guangming Lang, gmlang@cabaceo.com

2021-08-09

Kalman Filter

Suppose there is a system and its state can be measured with some accuracy at each time step. Suppose we can also formulate a model that roughly describes the process of how the state changes over time. Then we can use the process model to predict the state of the system at the next time step. The Kalman filter goes one step further. It combines the prediction from the process model and the measurement we get at the current time to form an estimate of the state at the next time step.

The Kalman filter always gives better estimates than the predictions from the process model alone because it also incorporates measurements. It predicts better even when our process model is pretty accurate and measurments are not accurate. This seems counter intuitive but if we run a thought experiment, we'd realize that adding information, albeit inaccurate information, will always improve forecast. Therefore, the key insight behind the Kalman filter is never throw away information.

A system can have one or more states. For example, a robot that moves in a 2D plane can be thought as a system of one state, namely, its position (though each position is a 2D vector). It can also be thought as a system of two states, namely, its position and velocity (also a 2D vector: direction and size). We said earlier that the state of a system needs to be observable and measurable, what we meant was that at least one of the states needs to be observable. In other words, if a system has two or more states, one or more states are allowed to be hidden and unobservable. For example, in the robot example where the system is composed of position and velocity, position would be an observable state whereas velocity would be a hidden state. This means that as time passes by, we'd measure the robot position (with a sensor for instance), and we'd not measure its velocity. In general, the Kalman filter algorithm requires measurements to be taken on the observable state of the system over time.

Let's now formally describe the components of the Kalman filter in mathematical notations. Let $x(t)$ be the state of the system at time t. In the most general case, $x(t)$ is a n dimensional tensor. Let $F(t+1|t)$ be the state transition matrix that describes the process. In addition, we need to assume that $x(t)$ has a multivariate Gaussian distribution with a mean of μ_t and a covariance matrix of P_t. We also need to assume that the process follows a multivariate Gaussian distribution with a mean of $F(t+1|t)x(t)$ and a covariance matrix of Q_t. The Kalman filter algorithm uses the propoerties of Gaussian distributions to predict and update the estimates of the mean vector and covariance matrix of the state variables. Finally, we will use $z(t)$ to denote the measurements or observations.

General Description of the Algorithm

We can look at the same Kalman filter algorithm from two perspectives, namely, the Bayesian perspective (Thrun, 2018) and the othognal projection perspective (Labbe, 2020). Both perspective are important, although the Bayesian perspective might be more helpful for building some intuition.

The Bayesian Perspective

Convolution (or addition intuitively) is performed in the predict step, and Bayesian posterior calculation (or multiplication intuitively) is performed in the update step.

Initialization

1. Initialize the state of the filter.
2. Initialize our belief in the state.

Predict

1. Use process model to predict state at the next time step.
2. Adjust belief to account for the uncertainty in prediction.

Update

1. Get a measurement and associated belief about its accuracy.
2. Compute how likely it is the measurement matches each state.
3. Update state belief with this likelihood.

The Othognal Projection Perspective

The algorithm described from the othognal projection perspective is everywhere identical with the Bayesian perspective except in the update step, where portioning the residual with the Kalman gain factor is mathematically equivalent to the computation of Bayesian posterior.

Initialization

1. Initialize the state of the filter.
2. Initialize our belief in the state.

Predict

1. Use process model to predict state at the next time step.
2. Adjust belief to account for the uncertainty in prediction.

Update

1. Get a measurement and associated belief about its accuracy.
2. Compute residual between the process-model-predicted state and measurement.
3. Compute scaling factor based on whether the measurement or prediction is more accurate. This scaling factor is often called the Kalman gain.
4. Set state between the prediction and measurement based on Kalman gain.
5. Update belief in the state based on how certain we are in the measurement.

Mathematical Equations of the Algorithm

Let's drop the subscript t in our notations to declutter and hence better focus on how the algorithm works. So we will write F, P, and Q instead of $F(t+1|t)$, P_t, and Q_t. Further more, let's use x to denote (the estimate of) the mean of $x(t)$, $\bar{x}$, and $\bar{P}$ as the predicted mean and covariance of the state after the predict step. The Kalman algorithm can then be precisely described by the following set of equations.

Predict

- $\bar{x} = Fx$
- $\bar{P} = FPF^T + Q$

Update

- $S = \bar{P} + R$
- $K = \bar{P}S^{-1}$, Kalman gain factor
- $y = z - \bar{x}$, residual
- $x = \bar{x} + Ky$, Kalman filter output
- $P = (I - KH)\bar{P}$, Kalman filter output

When there is no new information to gain from a measurement ($K = 0$), Kalman filter will simply use the prediction from our model of the system process as the forecast ($x = \bar{x}$). When there is 100% new information to gain from a measurement ($K = 0$), Kalman filter will just use the measurement as the forecast ($x = z$).

Forecast Next Day's Close Price

Kalman Filter Components Specification

If a stock price series is mean-reverting, we can trade on it and make a profit if we can accurately estimate its mean and the standard deviation for the future. Kalman filter is a good technique for doing that. Consider the following system:

$$x(t+1) = x(t) + \omega(t)$$

where $x(t)$ is the stock close price at time t with a normal distribution $N(\mu_t, P_t)$ and $\omega(t)$ is the process model distributed as $N(0, Q_t)$. In other words, we assume the spread between consecutive daily close prices is a white noise with a time varying variance.

Our measurement $z(t)$ would simply be the observed daily close price. Because there is no uncertainty in our "measurements" (i.e., the observed prices), it doesn't really make sense to talk about "measurement noise" in terms of measuring device (such as sensors) accuracy. Instead, we can interpret the measurement noise, $N(0, R_t)$, as our degree of belief that the observed price is meaningful. If we strongly believe the observed price has a large impact on future prices, i.e., if R_t is small, then we (and our Kalman filter) would update our prediction to be closer to the observed price. Conversely, if we are not sure if the observed price is impactful, i.e., if R_t is large, then we wouldn't want to use the observed price to augment our prediction. Let's assume that a close price with a large trading volume compared to the previous day is more likely to affect the next day's price. Specifically, let's assume

$$R_t = P_t * \frac{V_{t-1}}{min(V_{t-1}, V_t)}$$

where V_t is the daily trading volume at day t. This assumption works pretty well in practice (Sinclair, 2010).

Data & Initialization

We downloaded daily close price and volume data between January 1 and August 6, 2021 from Yahoo Finance for these five stock tickers: AAPL, AMZN, FB, GOOG, and TSLA. These daily close prices were used as the "measurements" $z(t)$. We initialized the price distribution $x(0) \sim N(\mu_0, P_0)$ with

- $\mu_0 = z(0)$, i.e., the close price on Jan 1, 2021.
- $P_0 =$ the variance of the 5-minute close prices on June 8th, 2021

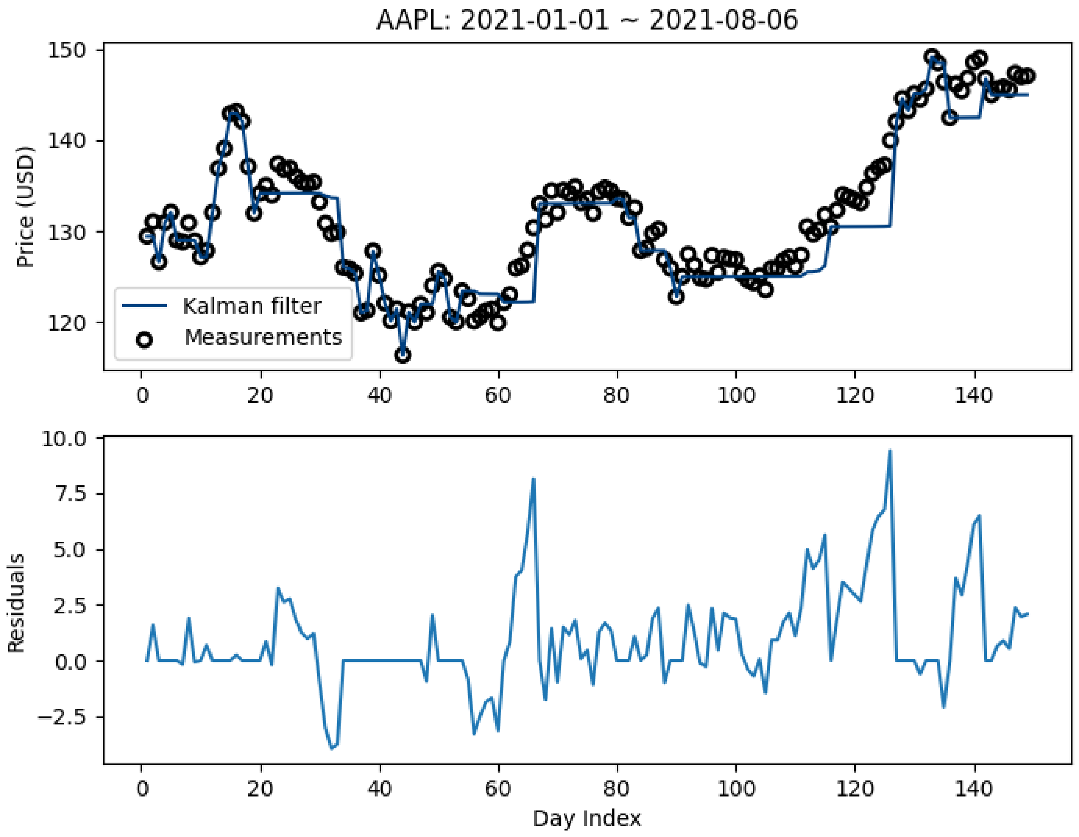
The choice of P_0 was determined by the fact that only the most recent 60 days of 5-minute data can be obtained from Yahoo Finance. But we experimented with different values and found that the Kalman filter forecasts were robust regardless of the initial value of the price variance.

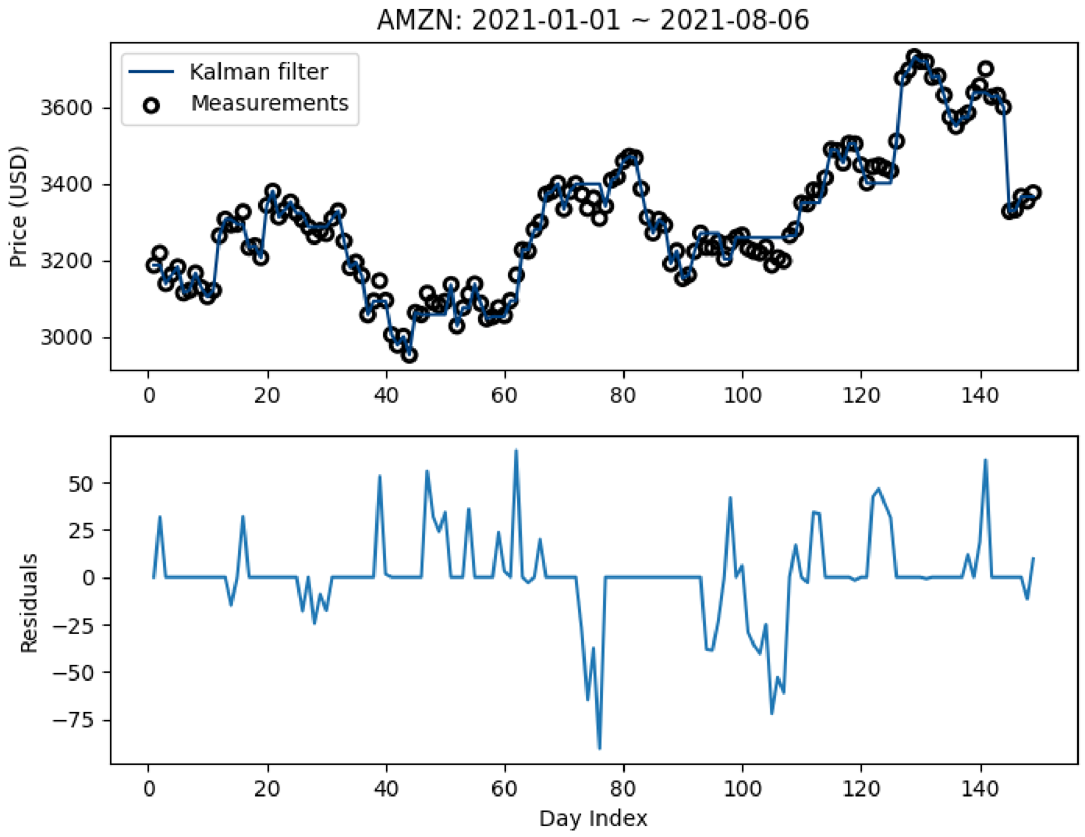
We initialized the variance Q_t of the process model $\omega(t)$ as $Q_0 = \frac{\delta}{1-\delta}$, where $\delta = 0.0001$ (Chan, 2013).

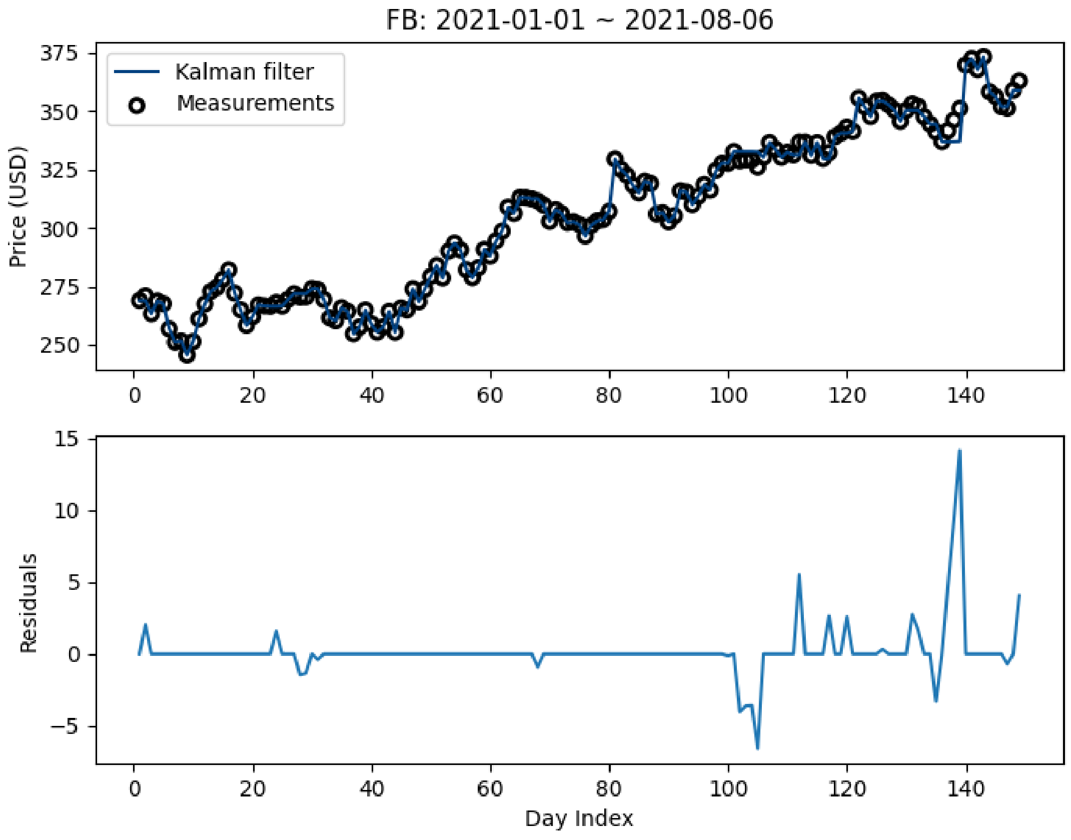
Results

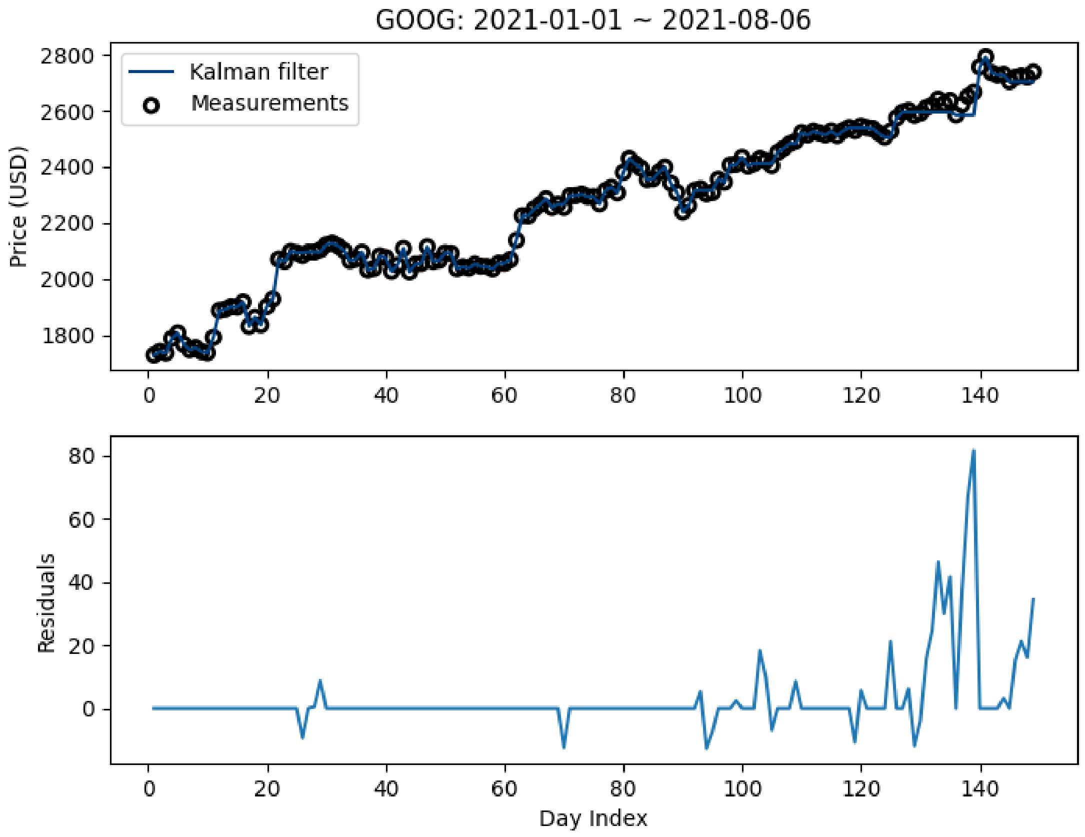
The figures in the following pages show our 1D Kalman filter forecasts along with the actual daily close prices for AAPL, AMZN, FB, GOOG, and TSLA between January 1 and August 6, 2021. We see that even with a simple 1D Kalman filter, we were able to spit out forecasts that track the actual prices really well! Moreover, our Kalman filter algorithm is smart enough to wait and see if a real trend develops before reflecting it in its forecasts. Finally, out of the five stocks, only TSLA showed a mean reverting pattern for this period. We can then investigate further and set up a mean reverting trading strategy for TSLA.

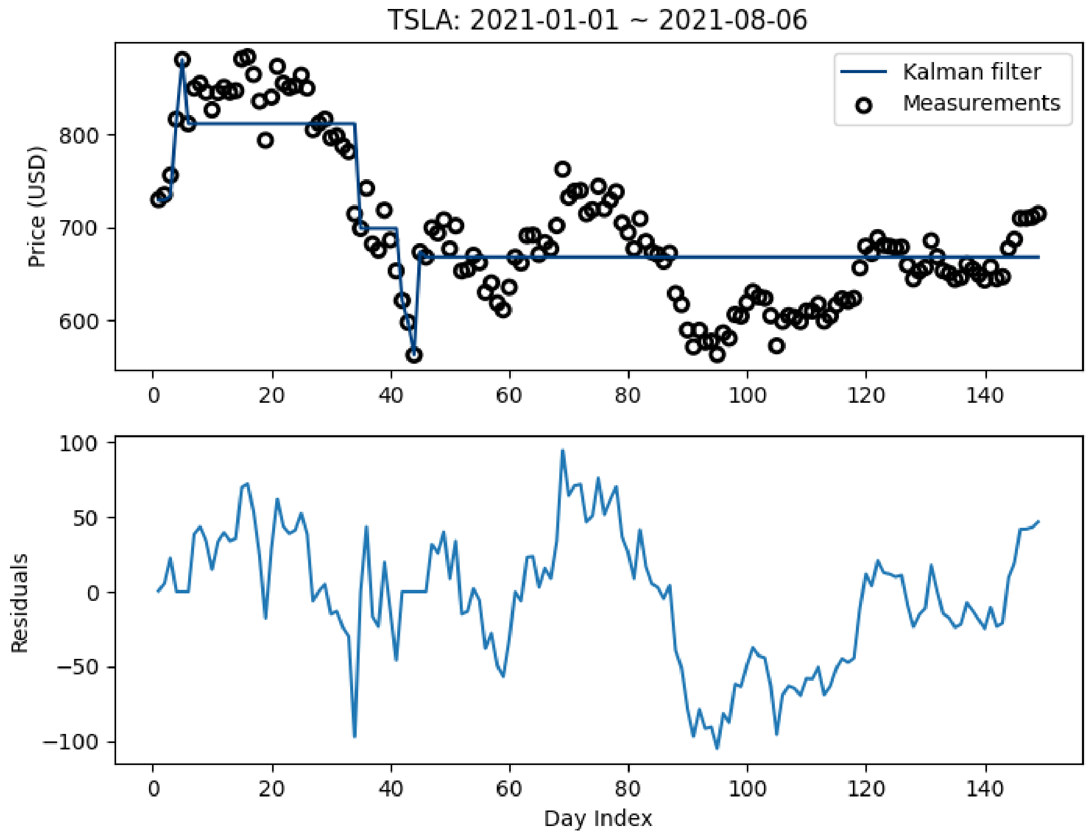
We also plotted the residuals underneath each Kalman filter forecast plot. By and large, the residuals are centered around zero with varying heights over time, so the residuals are not really white noise, which means the Gaussian distribution assumptions are somewhat violated. Nonetheless, our 1D kalman filter is very useful.


AAPL: 2021-01-01 ~ 2021-08-06
Price (USD)
Kalman filter
Measurements
Residuals
Day Index


AMZN: 2021-01-01 ~ 2021-08-06
Kalman filter
Measurements
Price (USD)
3600
3400
3200
3000
0
20
40
60
80
100
120
140
Residuals
50
25
0
−25
−50
−75
Day Index


FB: 2021-01-01 ~ 2021-08-06
Kalman filter
Measurements
Price (USD)
Residuals
Day Index


GOOG: 2021-01-01 ~ 2021-08-06
Kalman filter
Measurements
Price (USD)
Residuals
Day Index



Conclusions

We introduced Kalman filter and described how it works in this paper. The math equations we presented here were generally applicable for a system with n state variables. We also gave one application, namely, predicting the mean stock prices using daily close data and volume, and showed that a simple 1D Kalman filter can give very good forecasts. As a follow-up for future investigation, we can try a more complex 3D Kalman filter, introducing two hidden variables, namely the rate of change in the prices (velocity) and the curvature (acceleration). In practice, simpler models should be prefered since it's easier to overfit with complex models.

References

Labbe, Roger. Kalman and Bayesian Filters in Python. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2020.

Thrun, Sebastian. Artificial Intelligence for Robotics. https://www.udacity.com/course/cs373, 2018.

Sinclair, Euan. Option Trading: Pricing and Volatility Strategies and Techniques. Hoboken, NJ: John Wiley & Sons, 2010.

Chan, Ernest. Algorithmic Trading: Winning Strategies and Their Rationale. Hoboken, NJ: John Wiley, 2013.