Applications of Kalman Filtering in Financial Markets

Jin Hyun (Addy) Park
Jueun Kwon

May 17, 2024

1 Overview

One of the powers of model-based estimation techniques is its ability to estimate the "hidden states" based on observations, using both data and a model. While most applications in class were based on engineering examples (aircrafts, cars, etc.), model-based estimation techniques also have applications in finance. In the context of stock markets (or markets for any publicly traded assets), a parameter of particular interest is the market efficiency. Intuitively, the market efficiency represents *how quickly the market price of an asset adjusts to new information.* In an efficient market, because all new information is *immediately* reflected in the price, no investor can consistently achieve excess returns by trading on that information. However, this parameter is not directly observable. Instead, it must be estimated using a model and observed data. This is where filtering techniques can be applied. The paper "Estimation of market efficiency process within time-varying autoregressive models by extended Kalman filtering approach" explores the application of Kalman filtering in stock markets. Our goal in this project was to implement the standard and extended Kalman filters using the models they introduce and make meaningful conclusions about the US stock market.

2 Project Objectives

We had two objectives for this project: (1) identify different models used to model the dynamics of stock returns and (2) implement two different filters that estimate the market efficiency. More specifically, we will first apply a standard Kalman filter coupled with maximum likelihood estimation to estimate the parameters that represent the market efficiency. In this first approach, we assume that the market efficiency does not change with time. These parameters are parameters of the market that determine the state transition matrix, just like how stiffness and damping coefficients determine the state transition matrix for a spring-mass-damper system. Then in the second approach, we relax the assumption that these parameters are time-independent and implement an extended Kalman filter that performs joint state and parameter estimation.

3 Technical Approach

3.1 Terminology

Because we were not familiar with a lot of the finance jargons used in the paper, we had to start by learning different finance concepts.

Log returns series Instead of looking at the raw stock prices, the paper instead looks at the log returns series. This seems to be commonplace in finance. The log return is defined as the following:

$$r_t = log\left(\frac{P_t}{P_{t-1}}\right) = log(P_t) - log(P_{t-1})$$

Using log returns rather than the raw stock prices has a few advantages:

1. **Logarithmic returns are additive over time**: This additive property allows us to express the log return at the next time step as a linear combination of the log returns at the previous time steps. As will be discussed later, this allows our dynamic model to be linear. Using the log returns is equivalent to performing a nonlinear coordinate transformation on a nonlinear system to make it linear (like what's done in feedback linearization).

2. **Stationarity**: The statistics of raw stock prices tend to be non-stationary due to trends in prices (drift) and changing volatility (time-varying covariance). Using the log return makes the data statistics more stationary. Also, the log returns are closer to a Gaussian distribution than raw stock prices since the next value in the signal is a sum of the previous values (central limit theorem) and this approximation should get better with increasing autoregressive order n. Kalman filtering assumes that the random variables are normally distributed so this is useful.

3. **Independent of units and scale**: Because the log returns are normalized, this lends itself to easier comparisons between different assets that are measured in different units and/or scales.

Weak form market efficiency The weak form of market efficiency posits that current stock prices fully reflect all past trading information, meaning that future prices depend only on historical prices and not on other factors such as news, earnings reports, or economic indicators. Mathematically, this can be expressed as $y_{k+1} = f(y_{0:k})$. While other models incorporate additional factors like earnings reports, these are often difficult to quantify. Thus, for the purposes of this project, we will assume the weak form of market efficiency.

Autoregressive model The autoregressive model of order n, denoted $AR(n)$, is given by the following equation:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_n X_{t-n} + \epsilon_t$$

where X_t is the value of the series at time t, $\phi_1, \phi_2, ..., \phi_n$ are the coefficients of the model, and ϵ_t is a white noise at time t, which is usually assumed to be normally distributed with zero mean and constant variance. The autoregressive model is just a generalization of a Markov process. Instead of the next value in a time series depending exclusively on the value of the signal at the current time (as is for a Markov process), it is allowed to depend on the value of the signal up to n previous time steps. In other words, the system has some "memory" of its past, up to n time steps behind, given that the coefficients aren't zero. The log returns of stock prices can be modeled as following an autoregressive process:

$$y_t = \sum_{i=1}^{n} \beta_i y_{t-i} + \epsilon_t$$

where $\epsilon_t \sim N(0, \sigma_\epsilon^2)$.

Adaptive Market Hypothesis (AMH) This hypothesis models the β coefficients as being time-dependent. This is a more realistic model of the market as it can model changing market efficiency that can arise from changing investor behavior. With the coefficient being time-dependent, the model becomes:

$$y_t = \sum_{i=1}^{n} \beta_{i,t} y_{t-i} + \epsilon_t$$

Market efficiency The test for market efficiency is to estimate $\beta_1, \beta_2, ..., \beta_n$ and see if $\beta_1 = \beta_2 = ... = \beta_n = 0$. This condition would imply that the log return at one time step is completely uncorrelated with all past log returns. In other words, knowledge of all past log returns gives you zero information about what the next log return will be. This *instant* decorrelation implies that, in a completely efficient market, the log returns follow a white noise process.

The reason why the coefficients equalling zero corresponds to the market being described as "efficient" is for the same reason that the Kalman filter is considered "efficient." For a Kalman filter, the measurement residual (aka innovation) is a white noise process. This means that the residual at the next time step is completely uncorrelated with all past residuals. If it were not white noise, it would imply that the residual at the previous time step contained useful information that was not utilized (bc there exists nonzero correlation between past and present). This unutilized information could have been exploited to obtain a better estimate for the next time step. The Kalman filter, being an optimal estimator, maximizes the use of all available information from the residual, ensuring no information is left unused. Similarly, in the context of market efficiency, if the coefficients of an autoregressive model are zero, it implies that the past values do not provide any useful information about the future values. Hence, the market is considered "efficient" because there are no patterns in the time series data that can be exploited to predict future prices.

3.2 Filter Implementation

3.2.1 Maximum likelihood estimation with Kalman filter

The simplest dynamic model for the market dynamics is an autoregressive process of order n where the β coefficients are invariant with time:

$$\begin{bmatrix} y(t_{k+1}) \\ y(t_k) \\ y(t_{k-1}) \\ \vdots \\ y(t_{k-n+2}) \end{bmatrix} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n-1} & \beta_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} y(t_k) \\ y(t_{k-1}) \\ y(t_{k-n+2}) \\ \vdots \\ y(t_{k-n+1}) \end{bmatrix} + \begin{bmatrix} \epsilon t_{k+1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$z(t_k) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \mathbf{x}(t_k) + v(t_k)$$

where $v(t_k) \sim N(0, R)$.

To estimate $\beta_1, ..., \beta_n$ as well as the unknown process noise covariance σ_ϵ^2, maximum likelihood estimation with Kalman filter was used. A standard Kalman filter suffices as the dynamic and measurement models are linear. In doing this, we are effectively looking for the values of the parameters $[\beta_1, ..., \beta_n, \sigma_\epsilon^2]$ that lead to the Kalman filter estimates $\hat{\mathbf{x}}(t_{0:n})$ matching the observations (the real data) $z(t_{0:k})$ as closely as possible.

To implement this, we first need to find the log-likelihood function. For measurements with Gaussian noise, the log-likelihood function is given by:

$$log(\mathcal{L}(\beta)) = -\frac{1}{2} \sum_{k=1}^{N} \left(log(2\pi R) + \frac{(z(t_k) - H\mathbf{x}(t_k))^2}{R} \right)$$

Intuitively, this function calculates how well the KF state estimates $\hat{\mathbf{x}}(t_{0:n})$ produced by a specific values for parameters $[\beta_1, ..., \beta_n, \sigma_\epsilon^2]$ match all of the observed data $z(t_{0:k})$. To estimate the values for the parameters that best fit the real data, we defined a function that takes in specific values for parameters $[\beta_1, ..., \beta_n, \sigma_\epsilon^2]$ and computes the KF estimates $\hat{\mathbf{x}}(t_{0:n})$ using these values for the parameters and calculates the log-likelihood for the resulting estimates $\hat{\mathbf{x}}_{0:n}$. Then, we negated it and passed this function as a function of $[\beta_1, ..., \beta_n, \sigma_\epsilon^2]$ to `fminsearch` in order to find the parameter values for $[\beta_1, ..., \beta_n, \sigma_\epsilon^2]$ that maximizes the log-likelihood function. Note that, even though the measurement is a perfect measurement of the state, we still have to throw in some non-zero sensor noise term for numerical stability of the filter.

3.2.2 Maxmimum likelihood estimation with extended Kalman filter

In our second approach, we used an autoregression model that allows the β coefficient to vary with time. This model is given by the following state-space model:

$$\mathbf{x}(t_{k+1}) = \begin{bmatrix} \beta_1(t_{k+1}) \\ y(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \beta_1(t_k) + w(t_{k+1}) \\ \beta_1(t_{k+1}) y(t_k) + \epsilon(t_{k+1}) \end{bmatrix}$$

$$z(t_k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t_k) \\ x_2(t_k) \end{bmatrix} + v(t_k)$$

where the state vector at time t_k is $\mathbf{x}(t_k) = \begin{bmatrix} \beta_1(t_k) & y(t_k) \end{bmatrix}^T$ and $v(t_k) \sim N(0, R)$. With some substitution, this turns into:

$$\mathbf{x}(t_{k+1}) = \begin{bmatrix} \beta_1(t_{k+1}) \\ y(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \beta_1(t_k) + w(t_{k+1}) \\ \beta_1(t_k) y(t_k) + y(t_k) w(t_{k+1}) + \epsilon(t_{k+1}) \end{bmatrix}$$

Unlike before, β is no longer static and has its own dynamics. The dynamic model for the β_1 just follows a random walk and is a separate stochastic process from the log returns. Because this model is complex, the paper uses an autoregressive model of order 1, which just reduces to a Markov process. We took the same approach as the paper.

In the first approach, as shown in Section 3.2.1, after the maximum likelihood β coefficients were determined, the Kalman filter estimated the log returns. That is, parameter estimation and state estimation were treated as separate problems. In this approach, because we must estimate both the returns and the β parameters simultaneously in real

time (because β is changing with time), this is effectively a joint state and parameter estimation that we saw in the EKF lecture and homework 4. Therefore, we augment the state vector with the β coefficients, which turns the linear dynamics we had in the first approach into a nonlinear one because β is now a state variable (i.e. $\beta_i y_i$ becomes a product of two state variables). Since it is nonlinear, instead of a standard Kalman filter, we have to implement an extended Kalman filter.

We first tested the filter with arbitrarily chosen values for $[\sigma_w, \sigma_\epsilon, \beta_1(0)]$ and it appeared that the time series for β was very sensitive to σ_w. So we needed a better way to estimate these parameters. The paper suggests that the unknown parameters $[\sigma_w, \sigma_\epsilon, \beta_1(0)]$ could be estimated using real data. To estimate this, we just used the same approach as we did in the previous section. We used maximum likelihood estimation to find the set of parameters that best fit the data. Like before, we defined a function in MATLAB that takes in specific values for the parameters $[\sigma_w, \sigma_\epsilon, \beta_1(0)]$ and runs an EKF using these parameters to estimate $\hat{\mathbf{x}}(t_{0:n})$ and compute its likelihood. Then we passed this function to `fminsearch()` to find the set of parameters that maximize the likelihood (i.e. best fits the observed data). This method of parameter estimation is also common in engineering applications. For example, suppose we want to find the characteristics of the disturbance acting on an aircraft in cruise, for the controller design. To characterize the disturbance, you might collect real data of the aircraft's motion in cruise (position, velocity, acceleration, etc.) and use a technique like MLE to find the disturbance characteristics that best fits the real data. Then, you could use the disturbance characteristics to go design something like an LQG controller, which requires knowledge of PSD of the disturbance (to construct a linear system driven by white noise).

4 Results

4.1 Maximum Likelihood Estimation with Kalman Filter

The β values we estimated using approach in Section 3.2.1 and the AR(15) model were the following:

$$\beta_1 = -0.1468, \beta_2 = 0.0604, \beta_3 = 0.0320, \beta_4 = -0.0467, \beta_5 = 0.0673, \beta_6 = -0.1534, \beta_7 = 0.1321, \beta_8 = -0.0950,$$

$$\beta_9 = 0.0795, \beta_{10} = -0.0278, \beta_{11} = 0.0240, \beta_{12} = -0.0076, \beta_{13} = -0.0043, \beta_{14} = -0.0015, \beta_{15} = -0.0060$$

The estimated value of σ_ϵ^2 was 0.2342. We expected the β coefficients to decrease from β_1 to β_{15} because the log return at the next time step should have a stronger dependency on the recent values of the log returns (i.e. autocorrelation is usually a decreasing function). This seems to hold true for our β values. The values of β_{10} to β_{15} seem to be significantly smaller than the values of β_1 to β_9. So, the results make intuitive sense. If we had used monthly data for S&P 500 instead of daily, the decreasing trend in β coefficients should be more apparent. Running the filter with these values of β produces the following estimates:

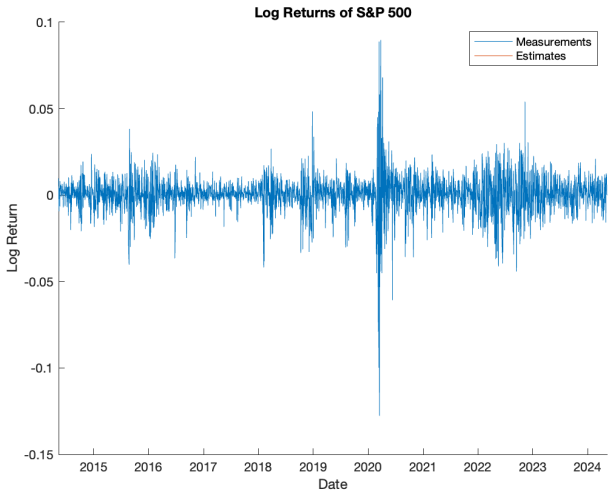

Figure 1: Kalman filter estimates for log returns using estimated β coefficients.

As expected, because we tuned β values such that the KF estimates fit the data, the estimates and data align almost perfectly. Also, because we set the measurement noise R to be very small (10^{-6}) (since the measurements are pretty much exact measurements of the states), the estimates would actually align with the data exactly regardless of what value of β we used, since the filter will trust the measurements almost entirely over the model (Kalman gain K was very close to 1). In other words, since the covariance of the measurements is so small, and all that the KF cares about is minimizing the error covariance, trusting the sensor entirely is the most optimal thing to do from the filter's perspective.

One of the questions that struck us was: why do we even bother using a filter in the first place when the measurement is exact? The answer we came to was that our goal is not to estimate the returns but is to estimate the unobserved ("hidden") states, $[\beta_1, \beta_2, ..., \beta_n]$, using observations and a model which can't be deduced from measurements alone.

Because a lot of the β values that were estimated clearly indicate nonzero autocorrelation (β_1 is 0.1468, β_6 is -0.1534, β_7 is 0.1321 which indicate significant correlation), this method indicates that the market from 2015 to 2024 was not efficient.

In addition, the peak observed in the log return graph, as shown in in Figure 1, through both the measurements and the estimates, represent a period of unusually high market volatility. Such a peak typically indicates that there was a significant market event that caused a sharp increase or decrease in stock prices over a short period of time. In our case, the sharp peak shown slightly after 2020 is caused by the COVID19 pandemic.

4.2 Maximum Likelihood Estimation with Extended Kalman Filter

The values for the parameters $[\sigma_w^2, \sigma_\epsilon^2, \beta_1(0)]$ estimated using MLE were:

$$\sigma_w^2 = 1.3049 \cdot 10^{-6}, \sigma_\epsilon^2 = 2.8100 \cdot 10^{-6}, \beta_1(0) = 0.0523$$

Running the filter with these values of $[\sigma_w^2, \sigma_\epsilon^2, \beta_1(0)]$ produced the following estimates for the log returns:

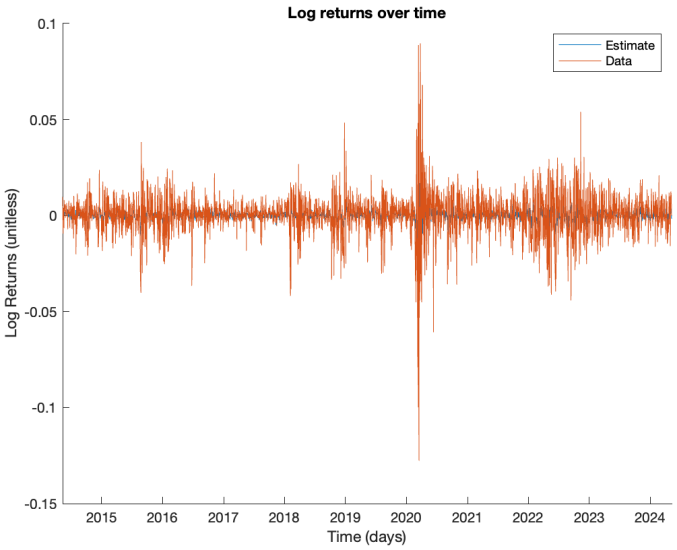

Figure 2: Extended Kalman filter estimates for log returns using estimated parameters.

Again, because the sensor noise covariance is so small, the filter trusts the measurements almost entirely, leading to the estimates matching the data. Plot of the estimates computed by running the filter with the estimated values of $[\sigma_w^2, \sigma_\epsilon^2, \beta_1(0)]$ is on the next page, in Figure 3.

For the estimate of the β coefficients, an interesting observation is that there is a downward spike at around February 24, 2020. This spike in the β coefficient around this time was due to COVID19. Around this time, the S&P500 index plummeted (Figure 4. The large β coefficient tells us that this sharp decrease was not due to random noise in the market but rather, there was a legitimate correlational pattern in the index.

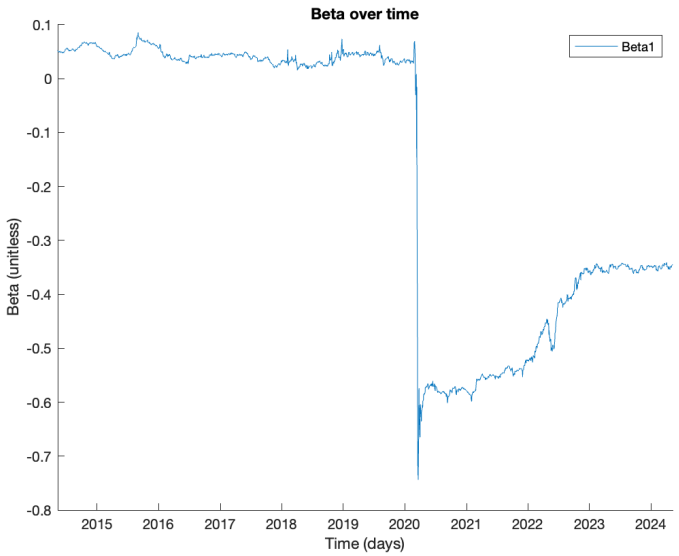

Figure 3: Extended Kalman filter estimates for β_1 using estimated parameters.

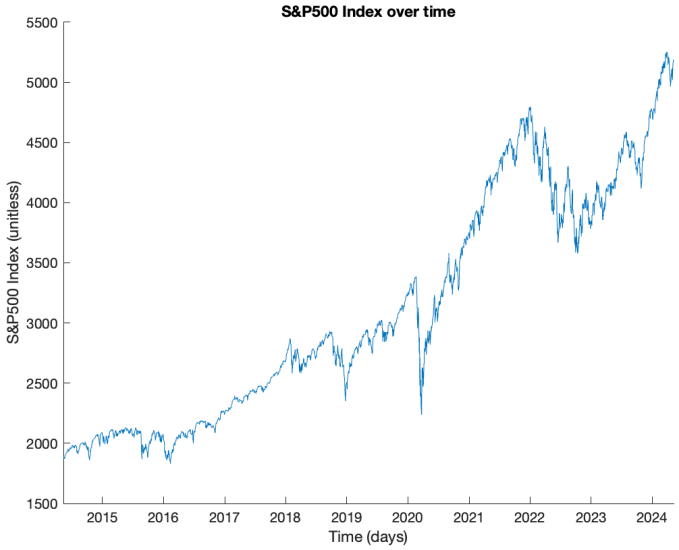

Figure 4: S&P 500 Index Over Time

As you can see in Figure 4, after the pandemic broke out in 2020, the market was more volatile than before. An observable pattern after 2020 is that sharp declines are usually followed by a rally in the market and vice versa. This is reflected in the β coefficient being in the negative range (-0.38 to -0.75). This method tells us that, prior to 2020, the log returns were more or less white noise (but not entirely) as seen by the small β value. However, after 2020, the market has remained inefficient and in a state of volatility as reflected in the large, negative value for β. If the market was efficient during the pandemic, the market should have equilibrated to the new price immediately and returned to following a white noise.

4.3 Eigenvalue Analysis

We can also look at the eigenvalues of the system to characterize market efficiency. The eigenvalues of the transition matrix were plotted for the approach used in Section 3.2.1 and is shown in Figure 5. There are a total of fifteen poles/eigenvalues for this system (F is 15×15). Most of the poles are near zero while there is one pole at around -0.15. For a *discrete* time system, the larger the magnitude of the real part, the slower the system response. This plot tells us that, of the fifteen modes of the system, one responds slowly (large rise time) to "new information" while the modes with zero eigenvalue do not contribute to the system's response.

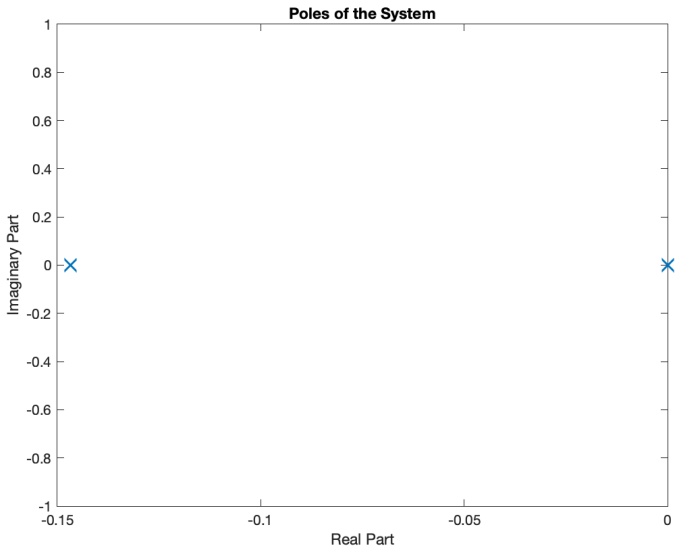

Figure 5: Eigenvalues plotted for MLE+KF approach

For the approach in Section 3.2.2, we plotted the real part of the eigenvalue of the transition matrix as a function of time as shown in Figure 6. Prior to 2020, the real part of the eigenvalue was positive and small (poles stable as long as within the unit circle). This means that the system responded quickly to new information. But after 2020, the real part became negative and large. This means that the system responded slowly to new information, which indicates market increased inefficiency after 2020.

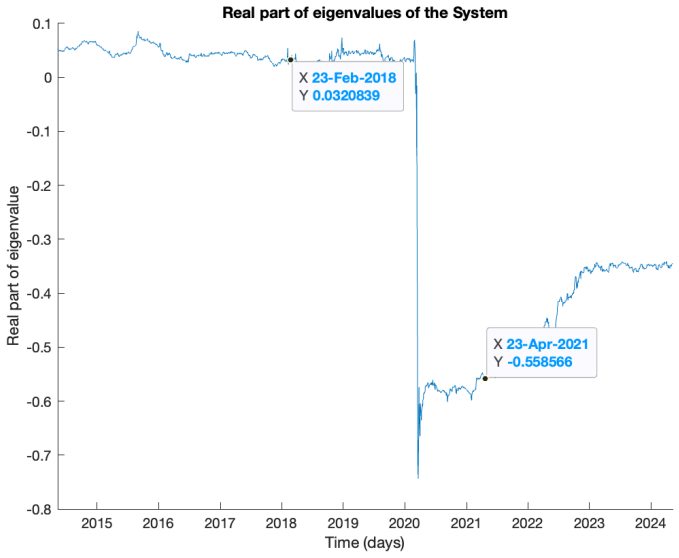

Figure 6: Eigenvalues plotted over time for MLE+EKF approach

5 Takeaway

Our maximum likelihood estimation with Kalman filter and extended Kalman filter allows us to estimate the market efficiency in real time. Our takeaway from this project is that the market is most profitable when the market efficiency is both large and positive. A large positive β value implies a strong positive autocorrelation in the log returns. This means that if returns were positive and high in the past, it is likely to be positive and high in the current market as well. In general, a large positive autocorrelation indicates a "momentum" effect where assets that have performed well in the past continue to perform well in the future while assets that have performed poorly in the past continue to perform poorly.

References

[1] Kulikova, M. V., & Kulikov, G. Yu. (2022). Estimation of market efficiency process within time-varying autoregressive models by extended Kalman filtering approach. *Digital Signal Processing*, *128*. https://doi.org/10.1016/j.dsp.2022.103619

A Appendices

A.1 MATLAB Code for Maximum Likelihood Estimation with Kalman Filter

```
%%%%%%%%%%%%%%%%%%%%%%% Setup and Initializations %%%%%%%%%%%%%%%%%%%%%%%
clc,clear,close all

n = 15; % order of autoregressive model

% Load data
filename = 'SP500_daily.csv';
dataold = readtable(filename);
data = rmmissing(dataold);
dates = data.DATE;
price = data.SP500;

% Compute log returns
logReturns = diff(log(price));

z = logReturns; % msmts
nt = size(logReturns,1); % num of msmts

% Noise statistics
sigeps_initial_guess = 0.1^2;
R = 10e-6; % sensor noise

% Initial state
x0 = [];
for i=1:n
    x0 = [logReturns(end-i) x0]; % fill x0 with n msmts starting from t=0
end
x0 = x0';
% Other initializaitons
P0 = diag(zeros(1,n));
beta_initial_guess = 0.01*ones(1,15);
param_initial_guess = [sigeps_initial_guess beta_initial_guess];

%%%%%%%%%%%%%%%%%%% Calculate Maximum Likelihood Beta %%%%%%%%%%%%%%%%%%%
fun = @(param) KF_log_likelihood(param,z,R,x0,P0); % fun to minimize wrt beta
options = optimset('MaxIter', 100000, 'MaxFunEvals', 100000);
param_estimated = fminsearch(fun,param_initial_guess,options) % find beta & Q
    that maximizes likelihood

%%%%%%%%%%%%%%%%%%%% Run KF with estimated beta and Q %%%%%%%%%%%%%%%%%%%%
Q_estimated = diag([param_estimated(1) zeros(1,n-1)]);
beta_estimated = param_estimated(2:end);

xhatp=zeros(n,nt);xhatp(:,1)=x0;
xhatu=zeros(n,nt);xhatu(:,1)=x0;
Pp=zeros(n,n,nt);Pp(:,:,1)=P0;
Pu=zeros(n,n,nt);Pu(:,:,1)=P0;
F = zeros(n,n);
F(1, :) = beta_estimated;
G = 1;
H = [1, zeros(1, n-1)];
% Run KF using estimated beta
for k=1:(nt-1)
```

```
    %Predict
    xhatp(:,k+1) = F*xhatu(:,k);
    Pp(:,:,k+1) = F*Pu(:,:,k)*F' + G*Q_estimated*G';
    %Kalman gain
    K = Pp(:,:,k+1)*H'*inv(H*Pp(:,:,k+1)*H' + R);
    %Update
    xhatu(:,k+1) = xhatp(:,k+1) + K *(z(k+1) - H*xhatp(:,k+1));
    Pu(:,:,k+1) = (eye(n)-K*H)*Pp(:,:,k+1)*(eye(n)-K*H)' + K*R*K';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot log return and estimate
figure(1);
hold on
plot(dates(2:end), logReturns);
title('Log Returns of S&P 500');
xlabel('Date');
ylabel('Log Return');
plot(xhatu(1))
legend("Measurements","Estimates")

%%%%%%%%%%%%%%%%%%%%%%% Plot eigenvalues and poles %%%%%%%%%%%%%%%%%%%%%%%
eigenvalues = eig(F);
figure(2)
plot(real(eigenvalues), imag(eigenvalues), 'x', 'MarkerSize', 10, 'LineWidth',
    2);
xlabel('Real Part');
ylabel('Imaginary Part');
title('Poles of the System');

%%%%%%%%%%%%%%%%%%%%%%%% KF & MLE Function calls %%%%%%%%%%%%%%%%%%%%%%%%
function log_likelihood = KF_log_likelihood(param,z,R,x0,P0)
% Function to compute the negative log-likelihood using Kalman Filter
% Inputs:
% beta: transition matrix (A matrix) parameters
% z: measurements
% R: measurement noise covariance
% Q: process noise covariance
% x0: initial condition
% P0: initial covaraince
%
% Output:
% log_likelihood: negative log-likelihood
Q = param(1);
beta = param(2:end);

N = length(z); % num of msmts
n = length(beta); % dim of state vec

F = zeros(n, n); % Define the transition matrix F based on beta
F(1, :) = beta;
G = 1;
for i = 2:n
    F(i, i-1) = 1;
end
H = [1, zeros(1, n-1)];
log_likelihood = 0;
```

```
xhatp = x0;
xhatu = x0;
Pp = P0;
Pu = P0;
for k=1:(N-1)
    % Predict
    xhatp = F*xhatu;
    Pp = F*Pu*F' + G*Q*G';
    % Kalman gain
    K = Pp*H'*inv(H*Pp*H' + R);
    % Update
    xhatu = xhatp + K *(z(k+1) - H*xhatp);
    Pu = (eye(n)-K*H)*Pp*(eye(n)-K*H)' + K*R*K';
    inn = z(k+1) - H * xhatp;
    % Accumulate log likelihood
    log_likelihood = log_likelihood-0.5*(log(2*pi*R)+(inn'*inv(R)*inn));
end
log_likelihood = -log_likelihood; % negate for minimization
end
```

A.2 MATLAB Code for Maxmimum likelihood Estimation with Extended Kalman Filter

```
%%%%%%%%%%%%%%%%%%%%%% Setup and Initializations %%%%%%%%%%%%%%%%%%%%%%
clc,clear all;close all;
rng(100);

global nt ns nw neps logReturns rand_w rand_eps

% Load data
filename = 'SP500_daily.csv';
dataold = readtable(filename);
data = rmmissing(dataold);
dates = data.DATE;
returns = data.SP500;

% Compute log returns
logReturns = diff(log(returns));

z = logReturns; % msmts
nt = size(logReturns,1); % num of msmts

% Simuate noise
nw = 1; % dim of disturbance on beta
neps = 1; % dim of disturbance on log return
ns = 2; % num of states
P0 = diag([0 0.5]);
R = 1e-6;
rand_w = randn(nw,nt);
rand_eps = rand(neps,nt);

%%%%%%%%%% Maximum likelihood estimation of Qw, Qeps, beta0 %%%%%%%%%%
param_init_guess = [0.0023 7.51e-7 0.8269];

fun = @(param) EKF_log_likelihood(param,z,R,P0); % function to minimize wrt
    param
```

```
options = optimset('MaxIter', 1000000000, 'MaxFunEvals', 1000000000);
estimated_param = fminsearch(fun,param_init_guess,options)

%%%%%%%%%%%%%%%%%% Run EKF using estimated parameters %%%%%%%%%%%%%%%%%%
Qw = estimated_param(1);
Qeps = estimated_param(2);
beta0 = estimated_param(3);
Q = diag([Qw Qeps]);
w=sqrtm(Qw)*randn(nw,nt);
eps=sqrtm(Qeps)*randn(neps,nt);

% Initialize xhatp, xhatu, Pp, Pu for EKF
x0 = [beta0 logReturns(1)]';
P0 = diag([0 0.5]);
H = [0 1];
R = 10e-6; % sensor noise
xhatp=zeros(ns,nt);xhatp(:,1)=x0;
xhatu=zeros(ns,nt);xhatu(:,1)=x0;
Pp=zeros(ns,ns,nt);Pp(:,:,1)=P0;
Pu=zeros(ns,ns,nt);Pu(:,:,1)=P0;

for k=1:(nt-1)
    % Predict state
    xhatp(:,k+1)=predict_state(xhatu(:,k),w(k+1),eps(k+1));
    % Predict covariance
    [F,G]=getFG(xhatu(:,k));
    Pp(1:ns,1:ns,k+1) = F*Pu(1:ns,1:ns,k)*F' + G*Q*G';
    % Kalman Gain
    K = Pp(1:ns,1:ns,k+1)*H'*inv(H*Pp(1:ns,1:ns,k+1)*H' + R);
    % Update
    xhatu(:,k+1) = xhatp(:,k+1) + K *(z(k+1) - H*xhatp(:,k+1));
    Pu(1:ns,1:ns,k+1) = (eye(ns)-K*H)*Pp(1:ns,1:ns,k+1)*(eye(ns)-K*H)' + K*R*K';
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plots %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)
hold on
plot(dates(2:end),xhatu(2,:))
plot(dates(2:end),logReturns)
title("Log returns over time")
xlabel("Time (days)")
ylabel("Log Returns (unitless)")
legend("Estimate","Data")

figure(2)
hold on
plot(dates(2:end),xhatu(1,:))
title("Beta over time")
xlabel("Time (days)")
ylabel("Beta (unitless)")
legend("Beta1")

figure(3)
hold on
plot(dates(2:end),returns(2:end))
title("S&P500 Index over time")
xlabel("Time (days)")
```

```
ylabel("S&P500 Index (unitless)")

%%%%%%%%%%%%%%%%%%%%%% Plot eigenvalues and poles %%%%%%%%%%%%%%%%%%%%%%
figure(4)
hold on
plot(dates(2:end),xhatu(1,:))
xlabel('Time (days)');
ylabel('Real part of eigenvalue');
title('Real part of eigenvalues of the System');

%%%%%%%%%%%%%%%%%%%%%% EKF & MLE Function calls %%%%%%%%%%%%%%%%%%%%%%
function Xkp1=predict_state(Xk,wkp1,epskp1)
% State prediction from k to k+1. Assumes mean of process noise is zero.
% Inputs:
% Xk: state vector at time k
% wk: disturbance input on beta at time k+1
% epsk: disturbance input on y at time k+1
% Output:
% Xpk1: state vector at time k+1
%
beta = Xk(1);
y = Xk(2);
G = [1 0;
     y 1];
Xkp1 = [beta;beta*y] + G*[wkp1;epskp1];
%
end

function [F,G]=getFG(Xk)
% Find the linearized system matrices F and G.
% Inputs:
% Xk: state vector at time k
% Outputs:
% [F,G]: lineraized system matrices F and G
%
beta = Xk(1);
y = Xk(2);

F = [1 0;
     y beta];
G = [1 0;
     y 1];
%
end

function log_likelihood = EKF_log_likelihood(param,z,R,P0)
% Function to compute the negative log-likelihood using Kalman Filter
% Inputs:
% param: system parameters [Qw,Qeps,beta0]
% z: measurements
% R: measurement noise covariance
% P0: initial covaraince
%
% Output:
% log_likelihood: negative log-likelihood
%
```

```
global ns nt  nw neps logReturns rand_w rand_eps % too lazy to pass in arguments
    ...
% Extract parameters
Qw = param(1);
Qeps = param(2);
Q = diag([Qw Qeps]);
beta0 = param(3);

% Generate noise
w = sqrtm(Qw)*rand_w;
eps = sqrtm(Qeps)*rand_eps;
% Initialize xhatp, xhatu, Pp, Pu for EKF
x0 = [beta0 logReturns(1)]';
H = [0 1];
xhatp = x0;
xhatu = x0;
Pp = P0;
Pu = P0;
log_likelihood = 0;
% EKF
for k=1:(nt-1)
    % Predict state
    xhatp = predict_state(xhatu,w(k+1),eps(k+1));
    % Predict covariance
    [F,G]=getFG(xhatu);
    Pp = F*Pu*F' + G*Q*G';
    inn = z(k+1) - H * xhatp;
    % Kalman Gain
    K = Pp*H'*inv(H*Pp*H' + R);
    %Update
    xhatu = xhatp + K *(z(k+1) - H*xhatp);
    Pu = (eye(ns)-K*H)*Pp*(eye(ns)-K*H)' + K*R*K';
    % Accumulate
    log_likelihood = log_likelihood-0.5*(log(2*pi*R)+(inn'*inv(R)*inn));
end
log_likelihood = -log_likelihood;
%
end
```

A.3 S&P 500 Historical Data

The historical S&P 500 stock prices were taken from Federal Reserve Economic Data (FRED) and is available here: https://fred.stlouisfed.org/series/SP500.