Black-Litterman-Bayes, Kalman Filter, ICA

Kaiwen Zhou, Youran Pan, Erding Liao

Contents

1 Topic: Black-Litterman-Bayes 2

2 Topic: Important Properties of the Kalman Filter 4

3 Topic: Kalman filter's Application on Financial Problems — Index Tracking Portfolios 7

Useful Theorem and Lemma

Theorem 0.1. ***(Binomial Inverse Theorem - Woodbury Matrix Identity)*** *If* $\mathbf{A}$, $\mathbf{U}$, $\mathbf{B}$, $\mathbf{V}$ *are matrices of sizes* $n \times n, n \times k, k \times k, k \times n$, *respectively, then*

(a) If $\mathbf{A}$ *and* $\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}$ *are nonsingular:*

$$(\mathbf{A} + \mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UB}\left(\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}\right)^{-1}\mathbf{BVA}^{-1}$$

(b) And (a) can be simplified to

$$(\mathbf{A} + \mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}.$$

Solution:

(a) Prove by verification. First notice that

$$(\mathbf{A} + \mathbf{UBV})\mathbf{A}^{-1}\mathbf{UB} = \mathbf{UB} + \mathbf{UBVA}^{-1}\mathbf{UB} = \mathbf{U}\left(\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}\right).$$

Now multiply the matrix we wish to invert by its alleged inverse

$$\begin{aligned}
&(\mathbf{A} + \mathbf{UBV})\left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UB}\left(\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}\right)^{-1}\mathbf{BVA}^{-1}\right) \\
&= \mathbf{I}_n + \mathbf{UBVA}^{-1} - \mathbf{U}\left(\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}\right)\left(\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB}\right)^{-1}\mathbf{BVA}^{-1} \\
&= \mathbf{I}_n + \mathbf{UBVA}^{-1} - \mathbf{UBVA}^{-1} \\
&= \mathbf{I}_n
\end{aligned}$$

which verifies that it is the inverse.

(b) Since $\mathbf{B} + \mathbf{BVA}^{-1}\mathbf{UB} = \mathbf{B}\left(\mathbf{I} + \mathbf{VA}^{-1}\mathbf{UB}\right)$ is nonsingular, we must have $\mathbf{B}$ is invertible. Then the two B terms flanking the quantity inverse in the right-hand side can be replaced with $\left(\mathbf{B}^{-1}\right)^{-1}$, which simplifies (a) to

$$(\mathbf{A} + \mathbf{UBV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}$$

Lemma 0.2. *If a multivariate normal random variable* $\boldsymbol{\theta}$ *has density* $p(\boldsymbol{\theta})$ *and*

$$-2\log p(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - 2\boldsymbol{\eta}^\top\boldsymbol{\theta} + (\textit{terms without } \boldsymbol{\theta})$$

then $\mathbb{V}[\boldsymbol{\theta}] = \boldsymbol{H}^{-1}$ *and* $\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{H}^{-1}\boldsymbol{\eta}$.

Solution: For $\boldsymbol{H}$ symmetric, we have

$$\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - 2\boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{\theta} = (\boldsymbol{\theta} - \boldsymbol{v})^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{v}) - \boldsymbol{v}^\top \boldsymbol{H}\boldsymbol{v}$$

Set $\boldsymbol{v} = \boldsymbol{H}^{-1}\boldsymbol{\eta}$ in the above equation, we obtain

$$-2\log p(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - 2\boldsymbol{\eta}^\top\boldsymbol{\theta} + (\text{terms without } \boldsymbol{\theta}) = (\boldsymbol{\theta} - \boldsymbol{H}^{-1}\boldsymbol{\eta})^\top \boldsymbol{H}(\boldsymbol{\theta} - \boldsymbol{H}^{-1}\boldsymbol{\eta}) + (\text{ terms without } \boldsymbol{\theta})$$

Therefore, we must have $\mathbb{V}[\boldsymbol{\theta}] = \boldsymbol{H}^{-1}$ and $\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{H}^{-1}\boldsymbol{\eta}$.

Definition 0.3. ***(Prior Predictive Distribution)*** *The prior predictive distribution is for predicting distribution for* x *BEFORE any sample of* x *has been gathered/observed. The only information we have at this stage is our belief about the prior,* $\pi(\boldsymbol{\theta})$, *and sampling distribution i.e.* $p(\boldsymbol{x} \mid \boldsymbol{\theta})$. *Then, the prior predictive distribution is given by*

$$p(\boldsymbol{x}) = \int_\Theta p(\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\theta} = \int_\Theta p(\boldsymbol{x} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

After the sample has been gathered, we obtain new information, i.e., the likelihood. Then, we can derive the posterior distribution $\boldsymbol{\theta}|\boldsymbol{x}_{observed}$ *and use that to predict new values/distribution for* x *which is given by the posterior predictive distribution:*

$$p(\boldsymbol{x}_{\text{ new }} | \boldsymbol{x}_{\text{ observed}}) = \int_\Theta p(\boldsymbol{x}, \boldsymbol{\theta} \mid \boldsymbol{x}_{\text{ observed}}) d\boldsymbol{\theta} = \int_\Theta p(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{x}_{\text{ observed}}) p(\boldsymbol{\theta} \mid \boldsymbol{x}_{\text{ observed}}) d\boldsymbol{\theta}$$

Lemma 0.4. *If X, Y, W, and V are real-valued random variables and a, b, c, d are real-valued constants, then the following facts are a consequence of the definition of covariance:*

$$\begin{aligned}
\operatorname{cov}(X, a) &= 0 \\
\operatorname{cov}(X, X) &= \operatorname{var}(X) \\
\operatorname{cov}(X, Y) &= \operatorname{cov}(Y, X) \\
\operatorname{cov}(aX, bY) &= ab \operatorname{cov}(X, Y) \\
\operatorname{cov}(X + a, Y + b) &= \operatorname{cov}(X, Y) \\
\operatorname{cov}(aX + bY, cW + dV) &= ac \operatorname{cov}(X, W) + ad \operatorname{cov}(X, V) + bc \operatorname{cov}(Y, W) + bd \operatorname{cov}(Y, V)
\end{aligned}$$

1 Topic: Black-Litterman-Bayes

Problem 1.1. *This question refers to the article "On The Bayesian Interpretation Of Black-Litterman" [1].*

(a) Derive formulas (10)-(11) using the properties of the multivariate normal distribution in the slides "Bayesian Modeling: Introduction".

*(b) **(Extra credit)** Derive formulas (22)-(26) using the same properties.*

Classic Black-Litterman Model from Bayesian Perspective:

Suppose we have $\boldsymbol{r} \sim N(\boldsymbol{\theta}, \boldsymbol{\Sigma})$, and since Black and Litterman were motivated by the guiding principle that, in the absence of any sort of information/views which could constitute alpha over the benchmark, the optimization procedure should simply return the global CAPM equilibrium portfolio, with holdings denoted $\boldsymbol{h}_{eq}$. Hence in the absence of any views, and with prior mean equal to Π, the investor's model of the world is that

$$\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Sigma}) \quad \text{and} \quad \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\Pi}, \boldsymbol{C}) \qquad \text{where } \boldsymbol{r}, \boldsymbol{\theta}, \boldsymbol{\Pi} \in \mathbb{R}^n, \boldsymbol{\Sigma}, \boldsymbol{C} \in \mathbb{R}^{n \times n} \tag{1}$$

A key aspect of the model is that the practitioner must also specify a level of uncertainty or "error bar" for each view, which is assumed to be an independent source of noise from the volatility already accounted for in a model such as $\boldsymbol{\Sigma}$. This is expressed as the following:

$$\boldsymbol{P}\boldsymbol{\theta} = \boldsymbol{q} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Omega}), \quad \boldsymbol{\Omega} = \operatorname{diag}(\omega_1, \ldots, \omega_k), \omega_i > 0 \tag{2}$$

where $\boldsymbol{P} \in \mathbb{R}^{k \times n}$, $\boldsymbol{\Omega} \in \mathbb{R}^{k \times k}$ and $\boldsymbol{q}, \boldsymbol{\epsilon} \in \mathbb{R}^k$

(a) **Solution:** In this question, we derive the mean $\boldsymbol{\nu}$ and covariance matrix $\boldsymbol{H}$ for the posterior.

Since the posterior is proportional to the product of the likelihood and the prior, to simplify our computation, we neglect the constant coefficients of related probability density functions in our derivation.

From Equation (2) and Equation (1), we have the likelihood function and the prior to be

$$f(\boldsymbol{q} \mid \boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{P}\boldsymbol{\theta} - \boldsymbol{q})^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{P}\boldsymbol{\theta} - \boldsymbol{q})\right], \quad \pi(\boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\Pi})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta} - \boldsymbol{\Pi})\right]$$

Leveraging the Bayes's formula, we have $f(\boldsymbol{\theta} \mid \boldsymbol{q}) \propto f(\boldsymbol{q} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$. It follows that (neglecting terms that do not contain $\boldsymbol{\theta}$)

$$-2 \log f(\boldsymbol{\theta} \mid \boldsymbol{q}) \propto (\boldsymbol{P}\boldsymbol{\theta} - \boldsymbol{q})^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{P}\boldsymbol{\theta} - \boldsymbol{q}) + (\boldsymbol{\theta} - \boldsymbol{\Pi})^\top \mathbf{C}^{-1}(\boldsymbol{\theta} - \boldsymbol{\Pi}) \xrightarrow[\text{drop terms without } \boldsymbol{\theta}]{\text{completing the squares}} \boldsymbol{\theta}^\top \left[\boldsymbol{P}^\top \boldsymbol{\Omega}^{-1} \mathbf{P} + \boldsymbol{C}^{-1}\right] \boldsymbol{\theta} - 2\left(\boldsymbol{q}^\top \boldsymbol{\Omega}^{-1} \mathbf{P} + \boldsymbol{\Pi}^\top \mathbf{C}^{-1}\right) \boldsymbol{\theta}$$

By Lemma 0.2, we obtain

$$\boldsymbol{\theta} \mid \boldsymbol{q} \sim \mathcal{N}\left(\boldsymbol{\nu}, \boldsymbol{H}^{-1}\right), \quad \boldsymbol{\nu} = \left[\boldsymbol{P}^\top \boldsymbol{\Omega}^{-1} \boldsymbol{P} + \boldsymbol{C}^{-1}\right]^{-1} \left[\boldsymbol{P}^\top \boldsymbol{\Omega}^{-1} \boldsymbol{q} + \boldsymbol{C}^{-1} \boldsymbol{\Pi}\right] \text{ and } \boldsymbol{H}^{-1} = \left[\boldsymbol{P}^\top \boldsymbol{\Omega}^{-1} \boldsymbol{P} + \boldsymbol{C}^{-1}\right]^{-1}$$

as desired.

APT Model:

Leveraging the powerful APT model (Roll & Ross, 1980; Ross, 1976), the parameter vector $\boldsymbol{\theta}$ could be generalized to represent the means of unobservable latent factors in the APT model, which assumes a linear functional form:

$$\boldsymbol{r} = \boldsymbol{X}\boldsymbol{f} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{D}) \text{ and } \boldsymbol{D} = \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_n^2\right), \sigma_i^2 > 0, \forall i \tag{3}$$

where $\boldsymbol{r}$ is an n-dimensional random vector containing the cross-section of returns in excess of the risk-free rate over some time interval $[t, t+1]$, and $\boldsymbol{X}$ is a (non-random) $n \times k$ matrix that is known before time t. The variable $\boldsymbol{f}$ in Equation (3) denotes a k-dimensional latent random vector process, and information about the $\boldsymbol{f}$-process must be obtained via statistical inference. Specifically, we assume that the $\boldsymbol{f}$-process has finite first and second moments given by

$$\mathbb{E}[\boldsymbol{f}] = \boldsymbol{\mu}_f \text{ and } \mathbb{V}[\boldsymbol{f}] = \boldsymbol{F}$$

In the Black-Litterman-Bayes model, we choose

$$\boldsymbol{\theta} = \boldsymbol{\mu}_f, \quad \boldsymbol{f} \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{F})$$

Likelihood Function (Views):

For our understanding, let's assume we are considering two latent factors: value and momentum. Typically, a quantitative portfolio manager might have views on individual factors: (1) a view on the value premium, and (2) another view on the momentum premium. It would be atypical for portfolio managers to have views on a combination (e.g. linear) of factors. Hence to keep things simple but still useful, we take the views equation to be:

$$\boldsymbol{\theta} = \boldsymbol{q} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}), \quad \boldsymbol{\Omega} = \operatorname{diag}\left(\omega_1^2, \ldots, \omega_k^2\right)$$

then, $\boldsymbol{q} \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{\Omega})$ and the corresponding likelihood function is therefore

$$f(\boldsymbol{q} \mid \boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{q})^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{\theta} - \boldsymbol{q})\right] = \prod_{i=1}^{k} \exp\left[-\frac{1}{2\omega_i^2}(\theta_i - q_i)^2\right]$$

Prior:

What prior for $\boldsymbol{\theta}$ should we choose? First, let's set the prior $\pi(\boldsymbol{\theta})$ to be

$$\pi(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\xi}, \boldsymbol{V})$$

where $\boldsymbol{\xi} \in \mathbb{R}^k$ and $\boldsymbol{V} \in S_{++}^k$, the set of symmetric positive definite $k \times k$ matrices.

Choosing a prior then amounts to choosing $\boldsymbol{\xi}$ and $\boldsymbol{V}$, and once a prior is chosen, we instantly obtain the prior predictive distribution $p(\boldsymbol{r})$ for $\boldsymbol{r}$ using the APT model and an associated prior (benchmark) portfolio with holdings $\boldsymbol{h}_B$ where $\boldsymbol{h}_B$ maximizes expected utility of wealth, where the expectation is taken with respect to the a priori distribution on asset returns

$$p(\boldsymbol{r}) = \int p(\boldsymbol{r} \mid \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}, \quad \boldsymbol{h}_B = \arg\max_{\boldsymbol{h}} \mathbb{E}\left[u\left(\boldsymbol{h}^\top \boldsymbol{r}\right)\right] = \arg\max_{\boldsymbol{h}} \int u\left(\boldsymbol{h}^\top \boldsymbol{r}\right) p(\boldsymbol{r}) d\boldsymbol{r}$$

Let's see what our prior (benchmark) portfolio will look like.

Since $\boldsymbol{r} = \boldsymbol{X}\boldsymbol{f} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{D})$ and $\boldsymbol{f} \mid \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\theta}, \boldsymbol{F})$, we can equivalently rewrite it as $\boldsymbol{f} = \boldsymbol{\theta} + \boldsymbol{w}$ where $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{F})$, and $\boldsymbol{w}$ is independent of $\boldsymbol{\theta}$. Then, by properties of multivariate normal distribution, we have

$$\boldsymbol{r} = \boldsymbol{X}\boldsymbol{f} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon} \quad \Longrightarrow \quad \boldsymbol{r}_\pi \sim \mathcal{N}\left(\boldsymbol{X}\boldsymbol{\xi}, \boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}\right)$$

The a priori optimal portfolio under CARA utility, $u(x) = -e^{-\lambda x}$, $\lambda \neq 0$, is then

$$\boldsymbol{h}_B := \boldsymbol{h}^* = (\lambda \mathbb{V}_\pi[\boldsymbol{r}])^{-1} \mathbb{E}_\pi[\boldsymbol{r}] = \lambda^{-1}\left(\boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}\right)^{-1} \boldsymbol{X}\boldsymbol{\xi}$$

Remark 1.2. In the original paper, the author derived the prior predictive mean and variance as

$$\mathbb{E}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{V}^{-1}\boldsymbol{\xi} \text{ and } \mathbb{V}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1}$$

where $\boldsymbol{H} := \boldsymbol{V}^{-1} + \boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{X}$ and $\boldsymbol{\Sigma} := \boldsymbol{D} + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top$.

We first note that due to his careless mistake, these results are wrong! The correct results are:

$$\mathbb{E}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{V}^{-1}\boldsymbol{\xi} \quad \text{and} \quad \mathbb{V}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1}$$

Then, substitute $\mathbf{A} = \boldsymbol{\Sigma}$, $\mathbf{U} = \mathbf{X}$, $\mathbf{V} = \mathbf{X}^\top$ and $\mathbf{B} = \mathbf{V}$ in the Woodbury identity (Theorem 0.1), we have

$$\mathbb{V}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{\Sigma} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}$$

It follows that

$$\left(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1}\boldsymbol{X} = \left(\boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{\Sigma}\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{X} + \boldsymbol{X} = \boldsymbol{X}\boldsymbol{V}\left(\boldsymbol{V}^{-1} + \boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{X}\right) = \boldsymbol{X}\boldsymbol{V}\boldsymbol{H}$$

The mean $\mathbb{E}_\pi[\boldsymbol{r}]$ then follows as

$$\mathbb{E}_\pi[\boldsymbol{r}] = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{X}^\top\boldsymbol{\Sigma}^{-1}\right)^{-1} \boldsymbol{\Sigma}^{-1}\boldsymbol{X}\boldsymbol{H}^{-1}\boldsymbol{V}^{-1}\boldsymbol{\xi} = \boldsymbol{X}\boldsymbol{V}\boldsymbol{H}\boldsymbol{H}^{-1}\boldsymbol{V}^{-1}\boldsymbol{\xi} = \boldsymbol{X}\boldsymbol{\xi}$$

Thus, we obtain

$$\boldsymbol{r}_\pi \sim \mathcal{N}\left(\boldsymbol{X}\boldsymbol{\xi}, \boldsymbol{X}\boldsymbol{V}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}\right)$$

which is consistent with our results. △

(b) **Solution:** In this question, we derive the posterior distribution for $\boldsymbol{r} \mid \boldsymbol{q}$ and the mean-variance optimal portfolio holdings.

Step 1: Calculate the posterior for $\boldsymbol{\theta} \mid \boldsymbol{q}$.

Since $\pi(\boldsymbol{\theta})$ is normal and the likelihood $f(\boldsymbol{q} \mid \boldsymbol{\theta})$ is also normal, the prior is a conjugate prior with respect to the normal likelihood. Then, the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{q})$ is also normal. Using this relation and $p(\boldsymbol{\theta} \mid \boldsymbol{q}) \propto f(\boldsymbol{q} \mid \boldsymbol{\theta})\pi(\boldsymbol{\theta})$, we have (neglecting terms that do not contain $\boldsymbol{\theta}$)

$$-2\log p(\boldsymbol{\theta} \mid \boldsymbol{q}) = (\boldsymbol{\theta} - \boldsymbol{q})^\top \boldsymbol{\Omega}^{-1}(\boldsymbol{\theta} - \boldsymbol{q}) + (\boldsymbol{\theta} - \boldsymbol{\xi})^\top \mathbf{V}^{-1}(\boldsymbol{\theta} - \boldsymbol{\xi}) = \boldsymbol{\theta}^\top\left(\boldsymbol{\Omega}^{-1} + \mathbf{V}^{-1}\right)\boldsymbol{\theta} - 2\left(\boldsymbol{q}^\top\boldsymbol{\Omega}^{-1} + \boldsymbol{\xi}^\top\mathbf{V}^{-1}\right)\boldsymbol{\theta} + (\text{ terms without } \boldsymbol{\theta})$$

Setting $\mathbf{H} = \left(\boldsymbol{\Omega}^{-1} + \mathbf{V}^{-1}\right)$ and $\boldsymbol{\eta}^\top = \left(\boldsymbol{q}^\top\boldsymbol{\Omega}^{-1} + \boldsymbol{\xi}^\top\mathbf{V}^{-1}\right)$ in Lemma 0.2, we have

$$\mathbb{V}[\boldsymbol{\theta} \mid \boldsymbol{q}] = \mathbf{H}^{-1} = \left(\boldsymbol{\Omega}^{-1} + \boldsymbol{V}^{-1}\right)^{-1}, \quad \mathbb{E}[\boldsymbol{\theta} \mid \boldsymbol{q}] = \mathbf{H}^{-1}\boldsymbol{\eta} = \mathbb{V}[\boldsymbol{\theta} \mid \boldsymbol{q}]\left(\boldsymbol{V}^{-1}\boldsymbol{\xi} + \boldsymbol{\Omega}^{-1}q\right) = \left(\boldsymbol{V}^{-1} + \boldsymbol{\Omega}^{-1}\right)^{-1}\left(\boldsymbol{V}^{-1}\boldsymbol{\xi} + \boldsymbol{\Omega}^{-1}\boldsymbol{q}\right)$$

Therefore, the posterior must satisfy

$$\boldsymbol{\theta} \mid \boldsymbol{q} \sim \mathcal{N}\left(\widetilde{\boldsymbol{\xi}}, \widetilde{\boldsymbol{V}}\right)$$

where $\widetilde{\boldsymbol{V}} = \left(\boldsymbol{V}^{-1} + \boldsymbol{\Omega}^{-1}\right)^{-1}$ and $\widetilde{\boldsymbol{\xi}} = \left(\boldsymbol{V}^{-1} + \boldsymbol{\Omega}^{-1}\right)^{-1}\left(\boldsymbol{V}^{-1}\boldsymbol{\xi} + \boldsymbol{\Omega}^{-1}\boldsymbol{q}\right)$.

Step 2: Calculate the posterior for $\boldsymbol{r} \mid \boldsymbol{q}$.

The a posteriori distribution of asset returns $\boldsymbol{r}$ (also called the posterior predictive density) is given by

$$p(\boldsymbol{r} \mid \boldsymbol{q}) = \int p(\boldsymbol{r} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{q}) d\boldsymbol{\theta}$$

For this one, we can view $p(\boldsymbol{\theta} \mid \boldsymbol{q})$ as the prior density for $\boldsymbol{\theta}$, then the situation here is the same as in deriving the prior (benchmark) portfolio. Hence, use the previous result and make the substitution $\boldsymbol{\xi} \to \widetilde{\boldsymbol{\xi}}$ and $\boldsymbol{V} \to \widetilde{\boldsymbol{V}}$, we obtain

$$\boldsymbol{r} \mid \boldsymbol{q} \sim \mathcal{N}\left(\boldsymbol{X}\widetilde{\boldsymbol{\xi}}, \boldsymbol{X}\widetilde{\boldsymbol{V}}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}\right)$$

Step 3: Calculate the mean-variance optimal portfolio.

The posterior optimal portfolio under CARA utility is then

$$\boldsymbol{h}^* = (\lambda \mathbb{V}[\boldsymbol{r} \mid \boldsymbol{q}])^{-1} \mathbb{E}[\boldsymbol{r} \mid \boldsymbol{q}] = \lambda^{-1}\left(\boldsymbol{X}\widetilde{\boldsymbol{V}}\boldsymbol{X}^\top + \boldsymbol{X}\boldsymbol{F}\boldsymbol{X}^\top + \boldsymbol{D}\right)^{-1} \boldsymbol{X}\widetilde{\boldsymbol{\xi}}$$

where $\widetilde{\boldsymbol{V}} = \left(\boldsymbol{V}^{-1} + \boldsymbol{\Omega}^{-1}\right)^{-1}$ and $\widetilde{\boldsymbol{\xi}} = \left(\boldsymbol{V}^{-1} + \boldsymbol{\Omega}^{-1}\right)^{-1}\left(\boldsymbol{V}^{-1}\boldsymbol{\xi} + \boldsymbol{\Omega}^{-1}\boldsymbol{q}\right)$.

2 Topic: Important Properties of the Kalman Filter

State Space Models:

Consider the following discrete linear dynamical system, which also referred to as linear state space model:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_t \\ \mathbf{y}_t &= \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \end{aligned} \quad \text{where} \quad \mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t), \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t) \tag{4}$$

Additionally, the initial state $\mathbf{x}_0$ and the noise terms $\mathbf{w}_t, \mathbf{v}_t$ are all assumed to be mutually independent.

Sequences $\{\mathbf{x}_\tau\}_{\tau=0}^t$ and $\{\mathbf{y}_\tau\}_{\tau=0}^t$ are called the hidden/latent state and observation sequences, respectively. It follows that the first and second equations are referred to the state and observation equations, respectively. In a state space model the state sequences $\{\mathbf{x}_\tau\}_{\tau=0}^t$ are not observable; however, the observation sequences $\{\mathbf{y}_\tau\}_{\tau=0}^t$ are fully observable.

The Filtering Problem:

The filtering problem is to estimate

$$\hat{\mathbf{x}}_t = \mathbb{E}(\mathbf{x}_t \mid \mathbf{Y}_t)$$

where $\mathbf{Y}_t := \{\mathbf{y}_0, \ldots, \mathbf{y}_t\}$. In other words, given the noisy observations, $\{\mathbf{y}_\tau\}_{\tau=0}^t$, we seek to estimate the expected state. And the optimal solution for our setup is given by the Kalman filter, also referred to as linear quadratic estimation (LQE).

Derive the Kalman Filter:

The derivation of the Kalman filter is by induction, showing that the observation and state sequences are conditionally Gaussian.

Base Case: Since $\mathbf{x}_0$ is Gaussian, $\mathbf{x}_1, \mathbf{y}_1$ are also Gaussians. It follows that

$$\begin{aligned} \mathbf{x}_1 &\sim \mathcal{N}\left(\mathbf{F}_1 \boldsymbol{\mu}_0, \mathbf{F}_1 \mathbf{P}_0 \mathbf{F}_1^\top + \mathbf{Q}_1\right) \\ \mathbf{y}_1 &\sim \mathcal{N}\left(\mathbf{H}_1 \mathbf{F}_1 \boldsymbol{\mu}_0, \mathbf{H}_1\left(\mathbf{F}_1 \mathbf{P}_0 \mathbf{F}_1^\top + \mathbf{Q}_1\right)\mathbf{H}_1^\top + \mathbf{R}_1\right) \end{aligned}$$

Induction Step: To perform the induction step, we assume that

$$\mathbf{x}_{t-1} \sim \mathcal{N}\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{P}_{t-1|t-1}\right)$$

where the subscript $t \mid t'$ to denote our belief about the state $\mathbf{x}_t$ given observations $\mathbf{Y}_{t'}$. We immediately obtain from Equation (4)

$$\begin{aligned} \mathbf{x}_t \mid \mathbf{Y}_{t-1} &\sim \mathcal{N}\left(\mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}, \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t\right) \\ \mathbf{y}_t \mid \mathbf{Y}_{t-1} &\sim \mathcal{N}\left(\mathbf{H}_t \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}, \mathbf{H}_t\left(\mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t\right)\mathbf{H}_t^\top + \mathbf{R}_t\right) \end{aligned}$$

As $\mathbf{x}_t$, $\mathbf{w}_t$ and $\mathbf{v}_t$ are independent, the covariance of $\mathbf{x}_t$ and $\mathbf{y}_t$ is given by

$$\operatorname{Cov}(\mathbf{x}_t, \mathbf{y}_t) = \operatorname{Cov}(\mathbf{x}_t, \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t) = \operatorname{Cov}(\mathbf{x}_t, \mathbf{H}_t \mathbf{x}_t) + \operatorname{Cov}(\mathbf{x}_t, \mathbf{v}_t) = \mathbb{E}\left[\mathbf{x}_t \mathbf{x}_t^\top \mathbf{H}_t^\top\right] = \left(\mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t\right)\mathbf{H}_t^\top$$

IS/Predict Step of the Kalman Filter: From the distribution of $\mathbf{x}_t \mid \mathbf{Y}_{t-1}$, we introduce the notation

$$\begin{aligned} \hat{\mathbf{x}}_{t|t-1} &:= \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} \\ \mathbf{P}_{t|t-1} &:= \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t \end{aligned}$$

Then, we see that the joint distribution of $\mathbf{x}_t$ and $\mathbf{y}_t$, conditioned on $\mathbf{Y}_{t-1}$, is given by the multivariate Gaussian

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{t|t-1} \\ \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{t|t-1} & \mathbf{P}_{t|t-1}\mathbf{H}_t^\top \\ \mathbf{H}_t \mathbf{P}_{t|t-1} & \mathbf{H}_t \mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{R}_t \end{bmatrix}\right)$$

IS/Update Step of the Kalman Filter: Let us now assume $\mathbf{y}_t$ is observed. Applying the conditional formulas for the multivariate Gaussian we obtain

$$\mathbf{x}_t \mid \mathbf{Y}_t \sim \mathcal{N}\left(\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}\right)$$

where

$$\hat{\mathbf{x}}_{t|t} := \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1}\right), \quad \mathbf{P}_{t|t} := \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right)\mathbf{P}_{t|t-1}, \quad \mathbf{K}_t := \mathbf{P}_{t|t-1}\mathbf{H}_t^\top \mathbf{S}_t^{-1}, \quad \mathbf{S}_t := \mathbf{H}_t \mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{R}_t$$

Specifically, $\mathbf{K}_t$ is called the Kalman gain. Clearly, our belief about $\mathbf{x}_t$ given $\mathbf{Y}_t$ is a multivariate Gaussian distribution.

Remark 2.1. From the above filter formulas, it follows that we can apply the predict and update steps recursively as new observations arrive. This makes the Kalman filter ideal for online real-time processing of information. △

Problem 2.2. *Choose three out of the four subproblems. It is optional to solve the other one for extra credit.*

(a) In class we derived the Kalman filter under the assumption that $\mathbb{E}\left[\mathbf{w}_t \mathbf{v}_t^\top\right] = \mathbf{0}$. Now, derive the Kalman filter under the assumption that $\mathbb{E}\left[\mathbf{w}_t \mathbf{v}_t^\top\right] = \mathbf{M}_t$.

Solution: We assume that

$$\mathbf{x}_{t-1} \sim \mathcal{N}\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{P}_{t-1|t-1}\right)$$

where the subscript $t \mid t'$ to denote our belief about the state $\mathbf{x}_t$ given observations $\mathbf{Y}_{t'}$.

Find $\mathbf{x}_t \mid \mathbf{Y}_{t-1}$ and $\mathbf{y}_t \mid \mathbf{Y}_{t-1}$:

From Equation (4), we get $\mathbf{x}_t = \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{w}_t$. Then, for $\mathbf{x}_t \mid \mathbf{Y}_{t-1}$, it's easy to deduce that

$$\mathbf{x}_t \mid \mathbf{Y}_{t-1} \sim \mathcal{N}\left(\mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t\right)$$

Since $\mathbf{y}_t = \mathbf{H}_t\mathbf{x}_t + \mathbf{v}_t = \mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t$, we have $\mathbb{E}\left[\mathbf{y}_t \mid \mathbf{Y}_{t-1}\right] = \mathbf{H}_t\mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1}$. By Lemma 0.4, we obtain

$$\begin{aligned}
\mathbb{V}\left[\mathbf{y}_t \mid \mathbf{Y}_{t-1}\right] = \mathbb{V}\left[\mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right] &= \operatorname{Cov}\left(\mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t, \mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right) \\
\mathbf{x}_{t-1} \text{ independent of } \mathbf{w}_t \text{ and } \mathbf{v}_t \Longrightarrow &= \operatorname{Cov}\left(\mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1}\right) + \operatorname{Cov}\left(\mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t, \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right) \\
&= \mathbf{H}_t\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top\mathbf{H}_t^\top + \mathbb{E}\left[\left(\mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right)\left(\mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right)^\top\right] \\
&= \mathbf{H}_t\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top\mathbf{H}_t^\top + \mathbf{H}_t\mathbb{E}\left[\mathbf{w}_t\mathbf{w}_t^\top\right]\mathbf{H}_t^\top + \mathbb{E}\left[\mathbf{v}_t\mathbf{v}_t^\top\right] + \mathbb{E}\left[\mathbf{H}_t\mathbf{w}_t\mathbf{v}_t^\top\right] + \mathbb{E}\left[\mathbf{v}_t\mathbf{w}_t^\top\mathbf{H}_t^\top\right] \\
&= \mathbf{H}_t\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top\mathbf{H}_t^\top + \mathbf{H}_t\mathbf{Q}_t\mathbf{H}_t^\top + \mathbf{R}_t + \mathbf{H}_t\mathbf{M}_t + \mathbf{M}_t^\top\mathbf{H}_t^\top \\
&= \mathbf{H}_t\left(\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t\right)\mathbf{H}_t^\top + \mathbf{R}_t + \mathbf{H}_t\mathbf{M}_t + \mathbf{M}_t^\top\mathbf{H}_t^\top
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathbf{x}_t \mid \mathbf{Y}_{t-1} &\sim \mathcal{N}\left(\mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t\right) \\
\mathbf{y}_t \mid \mathbf{Y}_{t-1} &\sim \mathcal{N}\left(\mathbf{H}_t\mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{H}_t\left(\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t\right)\mathbf{H}_t^\top + \mathbf{R}_t + \mathbf{H}_t\mathbf{M}_t + \mathbf{M}_t^\top\mathbf{H}_t^\top\right)
\end{aligned}$$

Find $\mathbf{x}_t \mid \mathbf{Y}_t$:

As $\mathbf{x}_{t-1}$, $\mathbf{w}_t$ and $\mathbf{x}_{t-1}$, $\mathbf{v}_t$ are independent, the covariance of $\mathbf{x}_t$ and $\mathbf{y}_t$ is given by

$$\operatorname{Cov}\left(\mathbf{x}_t, \mathbf{y}_t\right) = \operatorname{Cov}\left(\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{w}_t, \mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right) = \operatorname{Cov}\left(\mathbf{F}_t\mathbf{x}_{t-1}, \mathbf{H}_t\mathbf{F}_t\mathbf{x}_{t-1}\right) + \operatorname{Cov}\left(\mathbf{w}_t, \mathbf{H}_t\mathbf{w}_t + \mathbf{v}_t\right) = \left(\mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t\right)\mathbf{H}_t^\top + \mathbf{M}_t$$

- IS/Predict Step of the Kalman Filter: From the distribution of $\mathbf{x}_t \mid \mathbf{Y}_{t-1}$, we introduce the notation

$$\begin{aligned}
\hat{\mathbf{x}}_{t|t-1} &:= \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} \\
\mathbf{P}_{t|t-1} &:= \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^\top + \mathbf{Q}_t
\end{aligned}$$

Then, we see that the joint distribution of $\mathbf{x}_t$ and $\mathbf{y}_t$, conditioned on $\mathbf{Y}_{t-1}$, is given by the multivariate Gaussian

$$\begin{bmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{t|t-1} \\ \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{t|t-1} & \mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{M}_t \\ \mathbf{H}_t\mathbf{P}_{t|t-1} + \mathbf{M}_t^\top & \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{R}_t + \mathbf{H}_t\mathbf{M}_t + \mathbf{M}_t^\top\mathbf{H}_t^\top \end{bmatrix}\right)$$

- IS/Update Step of the Kalman Filter: Let us now assume $\mathbf{y}_t$ is observed. Applying the conditional formulas for the multivariate Gaussian we obtain

$$\mathbf{x}_t \mid \mathbf{Y}_t \sim \mathcal{N}\left(\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}\right)$$

where

$$\hat{\mathbf{x}}_{t|t} := \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}\right), \quad \mathbf{P}_{t|t} := \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\right)\mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{M}_t^\top, \quad \mathbf{K}_t := \mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{M}_t, \; \mathbf{S}_t := \mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{R}_t + \mathbf{H}_t\mathbf{M}_t + \mathbf{M}_t^\top\mathbf{H}_t^\top$$

Specifically, $\mathbf{K}_t$ is called the Kalman gain. Clearly, our belief about $\mathbf{x}_t$ given $\mathbf{Y}_t$ is a multivariate Gaussian distribution.

(b) *Using the notation from class, let us denote the error between the true and estimated states by the random variable $\widetilde{\mathbf{x}}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t$. We denote by $\mathbf{W}_t$ a known positive definite matrix. Show that the Kalman filter is the solution to the problem*

$$\min_{\hat{\mathbf{x}}_t} \mathbb{E}\left[\widetilde{\mathbf{x}}_t^\top\mathbf{W}_t\widetilde{\mathbf{x}}_t\right] \tag{5}$$

Solution: Since $\mathbf{W}_t$ is a known positive definite matrix, by Cholesky decomposition, we have $\mathbf{W}_t = \mathbf{L}_t\mathbf{L}_t^\top$. Recall that the Kalman filter approximates $\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]$ as $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t}$, and using the fact that $\widetilde{\mathbf{x}}_t^\top\mathbf{W}_t\widetilde{\mathbf{x}}_t \in \mathbb{R}$, we can rewrite the loss function as

$$\begin{aligned}
\mathbb{E}\left[\widetilde{\mathbf{x}}_t^\top\mathbf{W}_t\widetilde{\mathbf{x}}_t\right] &= \mathbb{E}\left[\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)^\top\mathbf{W}_t\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right] + \mathbb{E}\left[\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)^\top\mathbf{W}_t\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)\right] \\
&\quad + 2\cdot\mathbb{E}\left[\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)^\top\mathbf{W}_t\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right] \\
&= \mathbb{E}\left[\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)^\top\mathbf{L}_t\mathbf{L}_t^\top\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right] + \mathbb{E}\left[\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)^\top\mathbf{L}_t\mathbf{L}_t^\top\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)\right] \\
\phi(\hat{\mathbf{x}}_t) = \left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)^\top\mathbf{W}_t \Longrightarrow &\quad + 2\mathbb{E}\left\{\mathbb{E}\left[\phi(\hat{\mathbf{x}}_t)\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right) \mid \mathbf{Y}_t\right]\right\} \\
&= \mathbb{E}\left[\left\|\mathbf{L}_t^\top\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right\|^2\right] + \mathbb{E}\left[\left\|\mathbf{L}_t^\top\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)\right\|^2\right] + 2\mathbb{E}\left\{\mathbb{E}\left[\phi(\hat{\mathbf{x}}_t)\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right]\right\} \\
&= \mathbb{E}\left[\left\|\mathbf{L}_t^\top\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right\|^2\right] + \mathbb{E}\left[\left\|\mathbf{L}_t^\top\left(\mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right] - \hat{\mathbf{x}}_t\right)\right\|^2\right] \\
&\geq \mathbb{E}\left[\left\|\mathbf{L}_t^\top\left(\mathbf{x}_t - \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]\right)\right\|^2\right]
\end{aligned}$$

where the equality can be attained when $\hat{\mathbf{x}}_t = \mathbb{E}\left[\mathbf{x}_t \mid \mathbf{Y}_t\right]$.

Therefore, the Kalman filter is the solution to the problem $\min_{\hat{\mathbf{x}}_t}\mathbb{E}\left[\widetilde{\mathbf{x}}_t^\top\mathbf{W}_t\widetilde{\mathbf{x}}_t\right]$.

(c) ***(The Kalman filter is the Best LINEAR Predictor under Mean-Squared Error if noise is NOT Gaussian (Non-linear Predictors could be better.))*** *Now let us assume that $\mathbf{w}_t$ and $\mathbf{v}_t$ have zero mean, are uncorrelated with covariance matrices $\mathbf{Q}_t$ and $\mathbf{R}_t$, respectively (but they are no longer Gaussian). Show that the Kalman filter is the best linear solution to Equation (5).*

Solution: We assume that the estimate is a linear weighted sum of the prediction and the new observation and can be described by the equation,

$$\hat{\mathbf{x}}_{t|t} := \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \left(\mathbf{y}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1} \right)$$

where $\mathbf{K}_t$ is called the gain matrice. Our problem now is to reduced to finding the $\mathbf{K}_t$ and $\mathbf{K}'_t$ that minimise the conditional mean squared estimation error where of course the estimation error is given by:

$$\widetilde{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t} - \mathbf{x}_t$$

Unbiasedness: Assume that at state $t-1$, our predictor $E\left[\hat{\mathbf{x}}_{t-1|t-1}\right] = E\left[\mathbf{x}_{t-1}\right]$ is unbiased. Notice that

$$E\left[\hat{\mathbf{x}}_{t|t-1}\right] = E\left[\mathbf{F}_{t-1}\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{w}_{t-1}\right] = \mathbf{F}_{t-1} E\left[\hat{\mathbf{x}}_{t-1|t-1}\right] = E\left[\mathbf{x}_t\right]$$

Taking the expectation yields

$$E\left[\hat{\mathbf{x}}_{t|t}\right] = E\left[\hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}\right)\right] = E\left[\mathbf{x}_t\right] + \mathbb{E}\left[\mathbf{K}_t\mathbf{H}_t\mathbf{x}_t + \mathbf{K}_t\mathbf{v}_t - \mathbf{K}_t\mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}\right] = E\left[\mathbf{x}_t\right]$$

Therefore, the result given by the Kalman filter formulated as above is unbiased provided $E\left[\hat{\mathbf{x}}_{t|t}\right] = E\left[\mathbf{x}_k\right]$.

Formulate the Error Covariance:

We determined the prediction error covariance in equation (10). We now turn to the updated error covariance

$$E\left[\widetilde{\mathbf{x}}_{t|t}\widetilde{\mathbf{x}}_{t|t}^{\top} \mid \mathbf{Y}_k\right] = E\left[\left(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t}\right)\left(\mathbf{x}_k - \hat{\mathbf{x}}_{t|t}\right)^{\top}\right] = \mathbf{P}_{t|t} = \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\right)\mathbf{P}_{t|t-1}$$

Our goal is now to minimise the conditional mean-squared estimation error with respect to the Kalman gain, $\mathbf{K}$.

$$L = \min_{\mathbf{K}_t} E\left[\widetilde{\mathbf{x}}_{t|t}^{\top}\widetilde{\mathbf{x}}_{t|t} \mid \mathbf{Y}_t\right] = \min_{\mathbf{K}_k} \operatorname{trace}\left(E\left[\widetilde{\mathbf{x}}_{t|t}\widetilde{\mathbf{x}}_{t|t}^{\top} \mid \mathbf{Y}_t\right]\right) = \min_{\mathbf{K}_k} \operatorname{trace}\left(\mathbf{P}_{t|t}\right)$$

For any matrix $\mathbf{A}$ and a symmetric matrix $\mathbf{B}$

$$\frac{\partial}{\partial \mathbf{A}}\left(\operatorname{trace}\left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)\right) = 2\mathbf{A}\mathbf{B}$$

(to see this, consider writing the trace as $\sum_i \mathbf{a}_i^T\mathbf{B}\mathbf{a}_i$ where $\mathbf{a}_i$ are the columns of $\mathbf{A}^T$, and then differentiating w.r.t. the $\mathbf{a}_i$).

Differentiating with respect to the gain matrix (using the relation above) and setting equal to zero yields

$$\frac{\partial L}{\partial \mathbf{K}_t} = -2\left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t\right)\mathbf{P}_{t|t-1}\mathbf{H}_t^T + 2\mathbf{K}_t\mathbf{R}_t = \mathbf{0}$$

Re-arranging gives an equation for the gain matrix

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T\left[\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t\right]^{-1}$$

This is exactly what we have. Thus, the result given by the Kalman filter formulated as above is the best unbiased linear predictor provided $E\left[\hat{\mathbf{x}}_{t|t}\right] = E\left[\mathbf{x}_k\right]$.

(d) In class we derived the Kalman filter under the assumption that $\mathbf{w}_t$ and $\mathbf{v}_t$ are uncolored (i.e. each is serially uncorrelated).

(i) Derive the Kalman filter under the assumption that $\mathbf{w}_t$ is a VAR(1) process with known system matrix.

(ii) Derive the Kalman filter under the assumption that $\mathbf{v}_t$ is a $\mathrm{VAR}(1)$ process with known system matrix.

Hint: For (d), both (i) and (ii) can be solved by properly augmenting the state equations. Can you find a solution to (ii) where one does not have to augment the state? Is it possible to do so for (i) - why, or why not?

Solution:

Trends in Financial Data Science - Homework 2

Kaiwen Zhou, Youran Pan, Erding Liao

October 28, 2023

1 Pairs Trading

3 (a) Given the continuous-time Ornstein-Uhlenbeck process as

$$ds_t = \kappa(\theta - s_t)dt + \sigma dB_t. \tag{1}$$

Using the Euler-Maruyama method for discretizing stochastic differential equations, we have

$$s_k - s_{k-1} = \kappa\left(\theta - s_{k-1}\right)\Delta t + \sigma\Delta B_k, \tag{2}$$

where Δt is the time step size, and ΔB_k is the increment of the Brownian motion over the interval $[t_{k-1}, t_k]$.

Rearrange terms to express s_k in terms of s_{k-1} and a stochastic term

$$s_k = \theta\kappa\Delta t + s_{k-1}\left(1 - \kappa\Delta t\right) + \sigma\Delta B_k. \tag{3}$$

The expectation and variance of s_k given s_{k-1} are

$$\mathbb{E}\left[s_k \mid s_{k-1}\right] = \theta\kappa\Delta t + s_{k-1}\left(1 - \kappa\Delta t\right), \tag{4}$$

$$\mathbb{V}\left[s_k \mid s_{k-1}\right] = \sigma^2\Delta t. \tag{5}$$

We can now write the discrete-time OU process in the desired form by defining $\varepsilon = \sigma\Delta B_k$

$$s_k = \mathbb{E}\left[s_k \mid s_{k-1}\right] + \varepsilon_k, \tag{6}$$

where $k = 1, 2, \ldots,$ and ε_k is a random process with zero mean and variance $\sigma^2\Delta t$. Which demonstrates that the discrete time solution of $ds_t = \kappa(\theta - s_t)dt + \sigma dB_t$ is Markovian, as required.

3 (b) **Model formulation:** First, discretize the Ornstein-Uhlenbeck process to fit the Kalman filter framework

$$s_k - s_{k-1} = \kappa\left(\theta - s_{k-1}\right)\Delta t + \sigma\Delta B_k \tag{7}$$

$$s_k = \theta\kappa\Delta t + s_{k-1}\left(1 - \kappa\Delta t\right) + \varepsilon_k, \tag{8}$$

where ε_k is the process noise.

Next, formulate the state-space representation suitable for the Kalman filter.

Since being aware that given the state variable, the observation process is a nonlinear combination of the state variables.

Thus, instead of using the most common linear Kalman filter (KF), we choose to use the extended Kalman filter (EKF).

Extended Kalman filter implementation: Implement the extended Kalman filter with the formulated state-space model involves:

Predict:

$$\hat{\mathbf{x}}_{t|t-1} = f\left(\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_t\right), \tag{9}$$
$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t' + \mathbf{Q}_t, \tag{10}$$

which are aimed at predicting state estimate and Predicting covariance estimate, respectively.

Update:

$$\tilde{\mathbf{y}}_t = \mathbf{z}_t - h\left(\hat{\mathbf{x}}_{t|t-1}\right), \tag{11}$$
$$\mathbf{S}_t = \mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t' + \mathbf{R}_t, \tag{12}$$
$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t' \mathbf{S}_t^{-1}, \tag{13}$$
$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \tilde{\mathbf{y}}_t, \tag{14}$$
$$\mathbf{P}_{t|t} = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \mathbf{P}_{t|t-1}, \tag{15}$$

which are aimed at calculating measurement residual, residual covariance, near-optimal Kalman gain, updated state estimate, and updated covariance estimate, respectively.

where the state transition and observation matrices are defined to be the following Jacobians

$$\mathbf{F}_t = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t-1|t-1}, \mathbf{u}_k}, \tag{16}$$
$$\mathbf{H}_t = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{t|t-1}}. \tag{17}$$

Parameter estimation: Utilize the state estimates provided by the extended Kalman filter to update the estimates of θ and κ over time.

State variables:

$$\mathbf{x}_k = \begin{bmatrix} \kappa \\ \theta \end{bmatrix} \tag{18}$$

Process model:

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{19}$$
$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{w}_t, \tag{20}$$
$$\mathbf{w}_t \sim \mathcal{N}\left(0, \mathbf{Q}_t\right), \tag{21}$$

where $\mathbf{w}_t$ is the process noises, and $\mathbf{Q}_t$ is the corresponding Gaussian covariance matrix.

Measurement model:

$$h\left(\hat{\mathbf{x}}_{t|t-1}\right) = \kappa(\theta - \mathbf{S}_t)\Delta t, \tag{22}$$
$$\mathbf{z}_t = \Delta \mathbf{S}_t = \kappa(\theta - \mathbf{S}_t)\Delta t + \mathbf{v}_t, \tag{23}$$
$$\mathbf{v}_t \sim \mathcal{L}\left(0, \mathbf{R}_t\right), \tag{24}$$
$$\mathbf{H}_t = \frac{\partial h\left(\hat{\mathbf{x}}_{t|t-1}\right)}{\partial \mathbf{x}_{t|t-1}} = \begin{bmatrix} (\theta - \mathbf{S}_t)\,\Delta t & \kappa \Delta t, \end{bmatrix} \tag{25}$$

where $\mathbf{v}_t$ is the process noises, and $\mathbf{R}_t$ is the corresponding Gaussian covariance matrix.

Trading strategy: With the estimated parameters θ and κ, employ a pairs trading strategy

Entry: When the spread s_t deviates significantly from the mean θ (i.e., the deviation exceeds a predefined threshold), enter a trade:

(a) If $s_t > \theta + \text{threshold}$, short stock A and long stock B.

(b) If $s_t < \theta + \text{threshold}$, long stock A and short stock B.

Exit: Exit the trade when the spread reverts back towards the mean, i.e., s_t approaches θ, or after a predefined holding period or loss limit.

Position sizing: Utilize the estimated speed of mean reversion κ to inform the size of the positions. Faster mean reversion might justify larger position sizes.

Q3_ParisTrading_(c)(d)

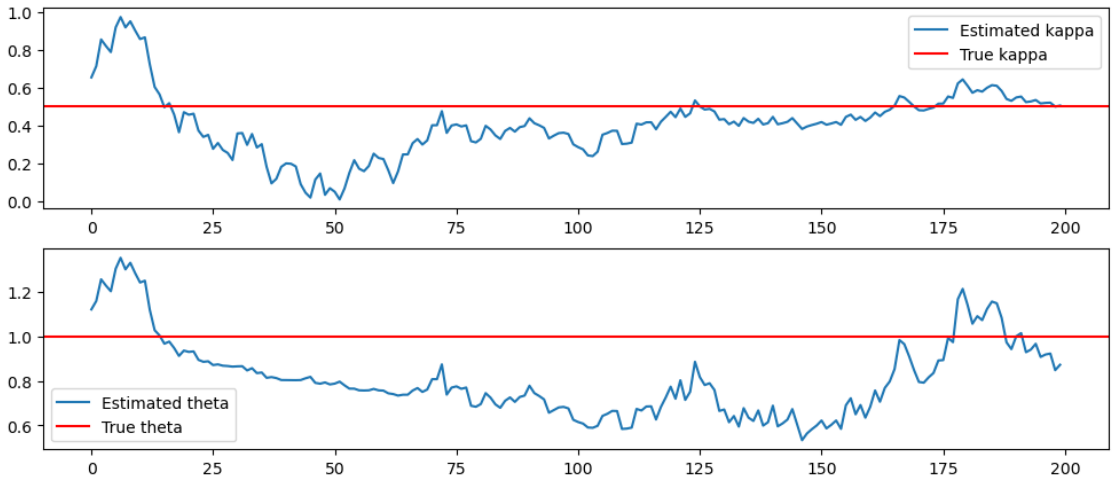
October 27, 2023

(c)

[1]:
```
import numpy as np
from numpy.random import randn
import matplotlib.pyplot as plt

# Define the Jacobian and measurement function as per your provided code
def H_Jacobian(x, dt, St):
    a = (x[1, 0] - St) * dt
    b = x[0, 0] * dt
    return np.array([[a, b]])

def hx(x, dt, St):
    return x[0, 0] * (x[1, 0] - St) * dt

# Define the OU simulation class
class OU_simulation:
    def __init__(self, dt, kappa, theta, St, sigma, n):
        self.kappa = kappa
        self.theta = theta
        self.St = St
        self.dt = dt
        self.sigma = sigma
        self.n = n

    def initial_guess(self):
        n_kappa = self.kappa  + .1*randn()
        n_theta = self.theta + .1*randn()
        return np.array([[n_kappa],[n_theta]])

    def generate_ou_process(self):
        x = np.zeros(self.n)
        x[0] = self.St
        for i in range(1,self.n):
            dw = np.sqrt(self.dt) * np.random.randn()
            dx = self.kappa * (self.theta - x[i-1]) * self.dt + self.sigma * dw
            x[i] = x[i-1] + dx
        return x
```

```
# Define the EKF class
class ExtendedKalmanFilter:
    def __init__(self, F, Q, R):
        self.F = F
        self.Q = Q
        self.R = R
        self.P = np.eye(2)
        self.K = np.zeros((2,1))

    def predict(self, x):
        x = np.dot(self.F, x)
        self.P = np.dot(np.dot(self.F, self.P), self.F.T) + self.Q
        return x

    def update(self, x, z, dt, St):
        H = H_Jacobian(x, dt, St)
        S = np.dot(H, np.dot(self.P, H.T)) + self.R
        self.K = np.dot(np.dot(self.P, H.T), np.linalg.inv(S))
        y = z - hx(x, dt, St)
        x += np.dot(self.K, np.array([[y]]))
        I = np.eye(2)
        self.P = np.dot(I - np.dot(self.K, H), self.P)
        return x

# Simulate the OU process
dt = 0.01
np.random.seed(42)
kappa = 0.5
theta = 1
St = 0.1
sigma = 0.4
ou = OU_simulation(dt, kappa, theta, St, sigma, n=200)
R = 0.4 * dt
Q = np.diag([0.01, 0.01])
F = np.array([[1, 0], [0, 1]])

# Initial guesses
x = ou.initial_guess()
ekf = ExtendedKalmanFilter(F, Q, R)

# Generate the OU process
process = ou.generate_ou_process()

# Apply EKF
kappa_estimates = []
theta_estimates = []
```

```
for i in range(len(process)):
    x = ekf.predict(x)
    if i == 0:
        z = process[i]
    else:
        z = process[i] - process[i-1]
    x = ekf.update(x, z, dt, process[i-1])
    kappa_estimates.append(x[0, 0])
    theta_estimates.append(x[1, 0])

# Plotting the results
plt.figure(figsize=(12,5))
plt.subplot(2,1,1)
plt.plot(np.arange(len(process)), kappa_estimates, label='Estimated kappa')
plt.axhline(y=ou.kappa, color='r', linestyle='-', label='True kappa')
plt.legend()
plt.subplot(2,1,2)
plt.plot(np.arange(len(process)), theta_estimates, label='Estimated theta')
plt.axhline(y=ou.theta, color='r', linestyle='-', label='True theta')
plt.legend()
plt.show()
```



How do you obtain a good estimate of σ? We decided to use a l2 loss function that measures the sum of squared residuals between the actual data $\mathbf{s}_t$ and the predicted data $\hat{\mathbf{s}}_t$ generated by the Ornstein-Uhlenbeck process:

$$J(\sigma) = \sum_{t=1}^{n} (\mathbf{s}_t - \hat{\mathbf{s}}_t)^2.$$

Minimize the objective function $J(\sigma)$ with respect to $ $, while keeping $ $ and $ $ fixed at their

estimated values from the Extended Kalman Filter (EKF) procedure:

$$\hat{\sigma} = \arg\min_{\sigma} J(\sigma).$$

Utilize a numerical optimization routine to find the value of $ $ that minimizes $J(\sigma)$, starting from an initial guess σ_0.

The optimized value $\hat{\sigma}$ obtained from the minimization procedure is taken as the estimated value of σ.

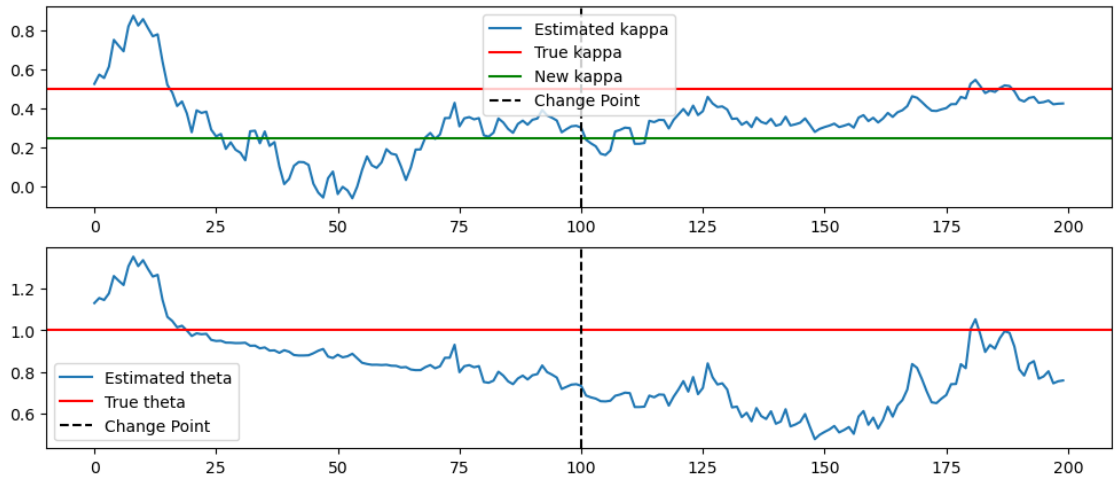
[2]:
```
from scipy.optimize import minimize
import numpy as np

np.random.seed(42)

# Objective function for calibrating sigma
def objective(sigma, data, dt, kappa, theta):
    n = len(data)
    residuals = np.zeros(n-1)
    for i in range(1, n):
        pred = data[i-1] + kappa * (theta - data[i-1]) * dt + sigma * np.
 ↪sqrt(dt) * np.random.randn()
        residuals[i-1] = data[i] - pred
    return np.sum(residuals**2)

# Calibrate sigma with fixed kappa and theta
def calibrate_sigma(data, dt, kappa, theta):
    initial_guess = [0.39]
    result = minimize(objective, initial_guess, args=(data, dt, kappa, theta))
    return result.x[0]

# Generate the OU process
process = ou.generate_ou_process()

# Parameters from EKF
kappa_est = kappa_estimates[-1]  # Take the last estimate as the representative␣
 ↪value
theta_est = theta_estimates[-1]  # Take the last estimate as the representative␣
 ↪value

# Calibrate sigma with fixed kappa and theta
sigma_cal = calibrate_sigma(process, dt, kappa_est, theta_est)

# Output
print(f"True Sigma: {ou.sigma}")
print(f"Calibrated Sigma: {sigma_cal}")
```

```
True Sigma: 0.4
Calibrated Sigma: 0.39
```

(d)

[3]:
```python
np.random.seed(42)

class OU_simulation_changing_kappa(OU_simulation):
    def __init__(self, change_point, new_kappa, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.change_point = change_point
        self.new_kappa = new_kappa

    def generate_ou_process(self):
        x = np.zeros(self.n)
        x[0] = self.St
        for i in range(1, self.n):
            dw = np.sqrt(self.dt) * np.random.randn()
            if i >= self.change_point:
                kappa = self.new_kappa
            else:
                kappa = self.kappa
            dx = kappa * (self.theta - x[i-1]) * self.dt + self.sigma * dw
            x[i] = x[i-1] + dx
        return x

# Simulate the OU process with changing kappa
change_point = 100  # Index at which kappa changes
new_kappa = kappa/2  # New value of kappa
ou_changing = OU_simulation_changing_kappa(change_point, new_kappa, dt, kappa,␣
 ↪theta, St, sigma, n=200)
process_changing = ou_changing.generate_ou_process()

# Reset the initial guesses and EKF
x = ou.initial_guess()
ekf = ExtendedKalmanFilter(F, Q, R)

# Apply EKF on the new process
kappa_estimates_changing = []
theta_estimates_changing = []
for i in range(len(process_changing)):
    x = ekf.predict(x)
    if i == 0:
        z = process_changing[i]
    else:
        z = process_changing[i] - process_changing[i-1]
    x = ekf.update(x, z, dt, process_changing[i-1])
    kappa_estimates_changing.append(x[0, 0])
```

```
    theta_estimates_changing.append(x[1, 0])

# Plotting the results
plt.figure(figsize=(12,5))
plt.subplot(2,1,1)
plt.plot(np.arange(len(process_changing)), kappa_estimates_changing,␣
 ↪label='Estimated kappa')
plt.axhline(y=kappa, color='r', linestyle='-', label='True kappa')
plt.axhline(y=new_kappa, color='g', linestyle='-', label='New kappa')  # New␣
 ↪line for new_kappa
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.subplot(2,1,2)
plt.plot(np.arange(len(process_changing)), theta_estimates_changing,␣
 ↪label='Estimated theta')
plt.axhline(y=theta, color='r', linestyle='-', label='True theta')
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.show()
```

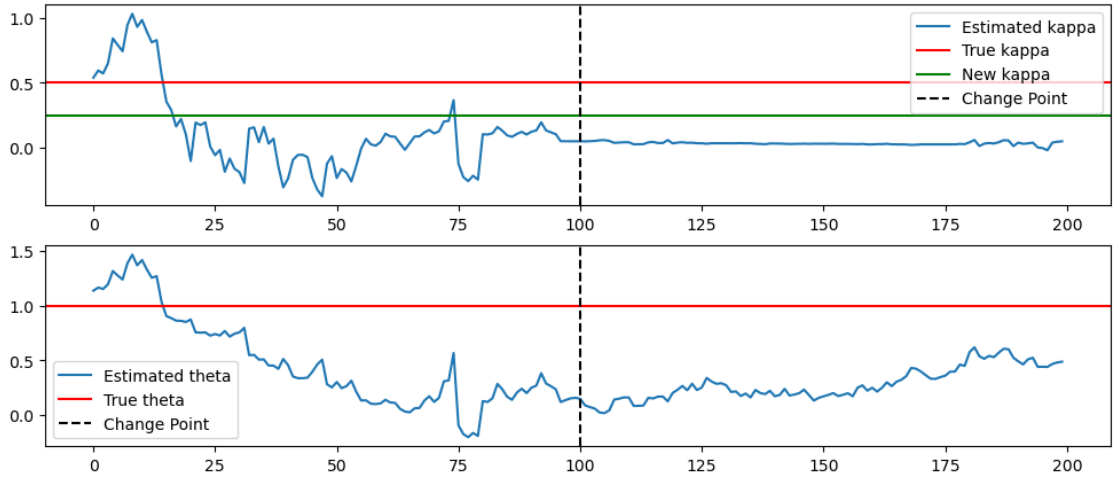


Speeding up the adjustment Method 1: Increading the observed noise R.

[4]:
```
np.random.seed(42)

class OU_simulation_changing_kappa(OU_simulation):
    def __init__(self, change_point, new_kappa, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.change_point = change_point
        self.new_kappa = new_kappa
```

```
    def generate_ou_process(self):
        x = np.zeros(self.n)
        x[0] = self.St
        for i in range(1, self.n):
            dw = np.sqrt(self.dt) * np.random.randn()
            if i >= self.change_point:
                kappa = self.new_kappa
            else:
                kappa = self.kappa
            dx = kappa * (self.theta - x[i-1]) * self.dt + self.sigma * dw
            x[i] = x[i-1] + dx
        return x

# Simulate the OU process with changing kappa
change_point = 100  # Index at which kappa changes
new_kappa = kappa/2  # New value of kappa
ou_changing = OU_simulation_changing_kappa(change_point, new_kappa, dt, kappa,␣
 ↪theta, St, sigma, n=200)
process_changing = ou_changing.generate_ou_process()

# Reset the initial guesses and EKF
x = ou.initial_guess()
Q_faster = Q = np.diag([0.1, 0.1])
ekf = ExtendedKalmanFilter(F, Q_faster, R)

# Apply EKF on the new process
kappa_estimates_changing = []
theta_estimates_changing = []
for i in range(len(process_changing)):
    x = ekf.predict(x)
    if i == 0:
        z = process_changing[i]
    else:
        z = process_changing[i] - process_changing[i-1]
    x = ekf.update(x, z, dt, process_changing[i-1])
    kappa_estimates_changing.append(x[0, 0])
    theta_estimates_changing.append(x[1, 0])

# Plotting the results
plt.figure(figsize=(12,5))
plt.subplot(2,1,1)
plt.plot(np.arange(len(process_changing)), kappa_estimates_changing,␣
 ↪label='Estimated kappa')
plt.axhline(y=kappa, color='r', linestyle='-', label='True kappa')
plt.axhline(y=new_kappa, color='g', linestyle='-', label='New kappa')  # New␣
 ↪line for new_kappa
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
```

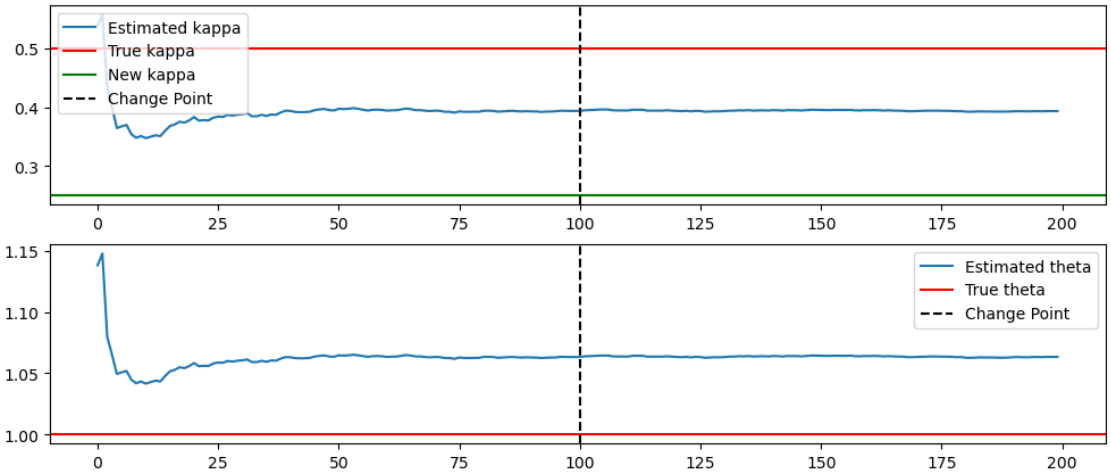
```
plt.legend()
plt.subplot(2,1,2)
plt.plot(np.arange(len(process_changing)), theta_estimates_changing,␣
 ↪label='Estimated theta')
plt.axhline(y=theta, color='r', linestyle='-', label='True theta')
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.show()
```



Method 2: Adaptive EKF, i.e., employ an adaptive filtering technique where the process and measurement noise covariances (Q and R) are updated dynamically based on the innovations (the differences between predicted and actual measurements).

[5]:
```
np.random.seed(42)

class AdaptiveExtendedKalmanFilter(ExtendedKalmanFilter):
    def update(self, x, z, dt, St):
        H = H_Jacobian(x, dt, St)
        S = np.dot(H, np.dot(self.P, H.T)) + self.R
        self.K = np.dot(np.dot(self.P, H.T), np.linalg.inv(S))
        y = z - hx(x, dt, St)
        # Adaptive update of Q and R based on innovation
        self.R = (1 - self.K[0, 0]) * self.R + self.K[0, 0] * y**2
        self.Q = self.Q + np.outer(self.K, self.K) * y**2  # Corrected line
        x += np.dot(self.K, np.array([[y]]))
        I = np.eye(2)
        self.P = np.dot(I - np.dot(self.K, H), self.P)
        return x

# Simulate the OU process with changing kappa
```

```
change_point = 100  # Index at which kappa changes
new_kappa = kappa/2  # New value of kappa
ou_changing = OU_simulation_changing_kappa(change_point, new_kappa, dt, kappa,␣
 ↪theta, St, sigma, n=200)
process_changing = ou_changing.generate_ou_process()

# Reset the initial guesses and EKF
x = ou.initial_guess()
aekf = AdaptiveExtendedKalmanFilter(F, Q, R)  # Using Adaptive EKF

# Apply EKF on the new process
kappa_estimates_changing = []
theta_estimates_changing = []
for i in range(len(process_changing)):
    x = aekf.predict(x)
    if i == 0:
        z = process_changing[i]
    else:
        z = process_changing[i] - process_changing[i-1]
    x = aekf.update(x, z, dt, process_changing[i-1])
    kappa_estimates_changing.append(x[0, 0])
    theta_estimates_changing.append(x[1, 0])

# Plotting the results
plt.figure(figsize=(12,5))
plt.subplot(2,1,1)
plt.plot(np.arange(len(process_changing)), kappa_estimates_changing,␣
 ↪label='Estimated kappa')
plt.axhline(y=kappa, color='r', linestyle='-', label='True kappa')
plt.axhline(y=new_kappa, color='g', linestyle='-', label='New kappa')  # New␣
 ↪line for new_kappa
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.subplot(2,1,2)
plt.plot(np.arange(len(process_changing)), theta_estimates_changing,␣
 ↪label='Estimated theta')
plt.axhline(y=theta, color='r', linestyle='-', label='True theta')
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.show()
```

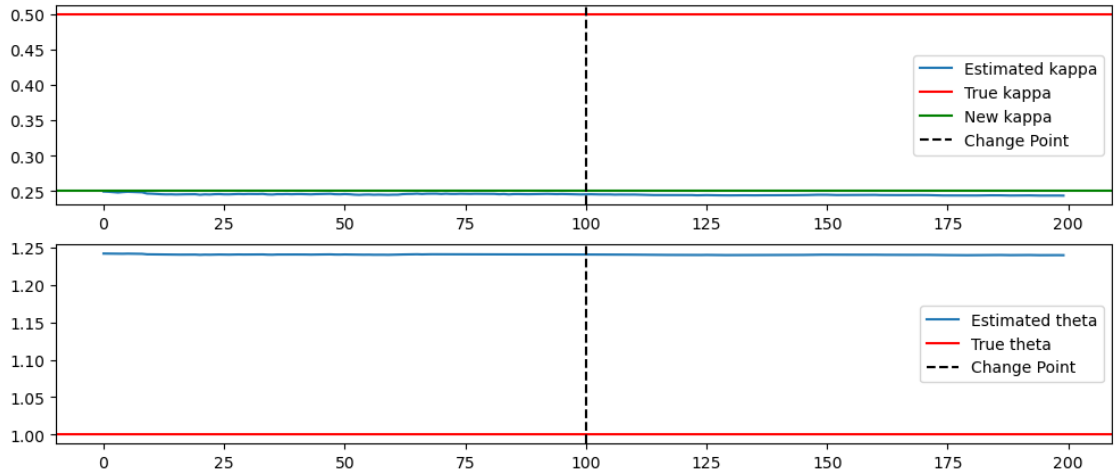


It seems that adaptive EKF works a bit more harder for current cituation.

Method 3: Resetting EKF: reset the filter state and covariance matrix whenever a significant change is detected. One simple change detection method could be to monitor the residuals (innovations) and reset the filter when a large residual is observed.

[6]:
```
class ResettingExtendedKalmanFilter(ExtendedKalmanFilter):
    def update(self, x, z, dt, St):
        H = H_Jacobian(x, dt, St)
        S = np.dot(H, np.dot(self.P, H.T)) + self.R
        self.K = np.dot(np.dot(self.P, H.T), np.linalg.inv(S))
        y = z - hx(x, dt, St)
        # Reset filter if large residual is observed
        if np.abs(y) > 0.1:  # Threshold may need to be tuned
            self.P = np.eye(2)
        x += np.dot(self.K, np.array([[y]]))
        I = np.eye(2)
        self.P = np.dot(I - np.dot(self.K, H), self.P)
        return x

# Simulate the OU process with changing kappa
change_point = 100  # Index at which kappa changes
new_kappa = kappa/2  # New value of kappa
ou_changing = OU_simulation_changing_kappa(change_point, new_kappa, dt, kappa,␣
 ↪theta, St, sigma, n=200)
process_changing = ou_changing.generate_ou_process()

# Reset the initial guesses and EKF
x = ou.initial_guess()
rekf = ResettingExtendedKalmanFilter(F, Q, R)  # Using Resetting EKF
```

```
for i in range(len(process_changing)):
    x = rekf.predict(x)
    if i == 0:
        z = process_changing[i]
    else:
        z = process_changing[i] - process_changing[i-1]
    x = rekf.update(x, z, dt, process_changing[i-1])
    kappa_estimates_changing.append(x[0, 0])
    theta_estimates_changing.append(x[1, 0])

# Apply EKF on the new process
kappa_estimates_changing = []
theta_estimates_changing = []
for i in range(len(process_changing)):
    x = aekf.predict(x)
    if i == 0:
        z = process_changing[i]
    else:
        z = process_changing[i] - process_changing[i-1]
    x = aekf.update(x, z, dt, process_changing[i-1])
    kappa_estimates_changing.append(x[0, 0])
    theta_estimates_changing.append(x[1, 0])

# Plotting the results
plt.figure(figsize=(12,5))
plt.subplot(2,1,1)
plt.plot(np.arange(len(process_changing)), kappa_estimates_changing,␣
 ↪label='Estimated kappa')
plt.axhline(y=kappa, color='r', linestyle='-', label='True kappa')
plt.axhline(y=new_kappa, color='g', linestyle='-', label='New kappa')  # New␣
 ↪line for new_kappa
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.subplot(2,1,2)
plt.plot(np.arange(len(process_changing)), theta_estimates_changing,␣
 ↪label='Estimated theta')
plt.axhline(y=theta, color='r', linestyle='-', label='True theta')
plt.axvline(x=change_point, color='k', linestyle='--', label='Change Point')
plt.legend()
plt.show()
```



It seems that resetting EKF works a lot more harder for current cituation.

By comparing the above three possible speeding up method, we need to be aware that speeding up is actually a speed and precision trade-off problem.

3 Topic: Kalman filter's Application on Financial Problems — Index Tracking Portfolios

Problem 3.1. *It is common in portfolio management to build so-called (index) tracking portfolios. Let us assume we are observing the return of the S&P 500 benchmark index, $r_{b,t}$. Now, let us pick a subset of 50 stocks from the constituents of this index. We will use these stocks to build a tracking portfolio for the index. For example, this could be the 50 companies in the index with the largest market cap. We denote the returns of these 50 stock by $\mathbf{r}_t \in \mathbb{R}^{50}$. The goal of finding a tracking portfolio is to find a dynamic trading strategy of the 50 stocks such that $\boldsymbol{\beta}_t^\top \mathbf{r}_t \approx r_{b,t}$, where $\boldsymbol{\beta}_t$ denotes the holdings (relative weights) of the tracking portfolio.*

(a) In this part, we assume that the covariance matrix of returns of the stocks in the $\mathrm{S\&P500}$, $\boldsymbol{\Sigma}$, is given and constant through time. Find the portfolio of these 50 stocks that minimizes the tracking error to $r_{b,t}$, i.e. find the solution to

$$\boldsymbol{\beta}_t^* = \mathrm{argmin}_{\boldsymbol{\beta}_t} \sqrt{\mathbb{V}\left[r_{b,t} - \boldsymbol{\beta}_t^\top \mathbf{r}_t\right]}.$$

What specific property does $\boldsymbol{\beta}_t^*$ have here?

Solution:

Formulation 1:

Suppose $\boldsymbol{S}$ and $\boldsymbol{\mu}_t = \mathbb{E}[\boldsymbol{r}_t]$ are the covariance matrix and the mean of the returns of the 50 stocks we picked, respectively. Additionally, let $\boldsymbol{e} = (1, \ldots, 1) \in \mathbb{R}^{50}$ denote the vector of all 1's. Since $r_{b,t}$ is deterministic, $\mathbb{V}\left[r_{b,t} - \boldsymbol{\beta}_t^\top \mathbf{r}_t\right] = \mathbb{V}\left[\boldsymbol{\beta}_t^\top \mathbf{r}_t\right] = \mathbb{E}\left[\boldsymbol{\beta}_t^\top \mathbf{r}_t \mathbf{r}_t^\top \boldsymbol{\beta}_t\right] = \boldsymbol{\beta}_t^\top \boldsymbol{S} \boldsymbol{\beta}_t$ and $\boldsymbol{\beta}_t^\top \mathbf{r}_t \approx r_{b,t}$, we have

$$\boldsymbol{\beta}_t^* = \mathrm{argmin}_{\boldsymbol{\beta}_t} \sqrt{\boldsymbol{\beta}_t^\top \boldsymbol{S} \boldsymbol{\beta}_t} = \mathrm{argmin}_{\boldsymbol{\beta}_t} \frac{1}{2} \boldsymbol{\beta}_t^\top \boldsymbol{S} \boldsymbol{\beta}_t \quad \text{given} \quad \boldsymbol{\beta}_t^\top \mathbb{E}[\mathbf{r}_t] = \boldsymbol{\beta}_t^\top \boldsymbol{\mu}_t = r_{b,t}, \quad \boldsymbol{\beta}_t^\top \boldsymbol{e} = 1$$

Applying the Lagrange multipliers, we have Lagrangian

$$L \equiv L(\boldsymbol{\beta}_t, \lambda, \gamma) := \frac{1}{2} \boldsymbol{\beta}_t^\top \boldsymbol{S} \boldsymbol{\beta}_t + \lambda \left(1 - \boldsymbol{\beta}_t^\top \boldsymbol{e}\right) + \gamma \left(r_{b,t} - \boldsymbol{\beta}_t^\top \boldsymbol{\mu}_t\right)$$

The first-order necessary condition then yields

$$FOC : \frac{\partial L}{\partial \boldsymbol{\beta}_t} = \boldsymbol{S} \boldsymbol{\beta}_t - \lambda \boldsymbol{e} - \gamma \boldsymbol{\mu}_t = \mathbf{0} \quad \Longrightarrow \quad \boldsymbol{\beta}_t = \boldsymbol{S}^{-1}(\lambda \boldsymbol{e} + \gamma \boldsymbol{\mu}_t) \tag{6}$$

Since $\boldsymbol{\beta}_t^\top \boldsymbol{\mu}_t = r_{b,t}$ and $\boldsymbol{\beta}_t^\top \boldsymbol{e} = 1$, we obtain

$$\begin{aligned} e^\top \boldsymbol{S}^{-1}(\lambda e + \gamma \boldsymbol{\mu}_t) = 1, \\ \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1}(\lambda e + \gamma \boldsymbol{\mu}_t) = r_{b,t} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \boldsymbol{e}^\top \boldsymbol{S}^{-1} e \lambda + \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t \gamma = 1, \\ \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{e} \lambda + \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t \gamma = r_{b,t} \end{aligned}$$

Now, for simplicity, we set $A = \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{e}, B = \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t = \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{e}$, and $C = \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t$ and obtain

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} \lambda \\ \gamma \end{bmatrix} = \begin{bmatrix} 1 \\ r_{b,t} \end{bmatrix} \Longrightarrow \begin{bmatrix} \lambda^* \\ \gamma^* \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ r_{b,t} \end{bmatrix} = \frac{1}{AC - B^2} \begin{bmatrix} C & -B \\ -B & A \end{bmatrix} \begin{bmatrix} 1 \\ r_{b,t} \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} C - B r_{b,t} \\ -B + A r_{b,t} \end{bmatrix} \tag{7}$$

where $\Delta = AC - B^2$. Then, plug in Equation (6), we get

$$\boldsymbol{\beta}_t^* = \lambda^* \boldsymbol{S}^{-1} \boldsymbol{e} + \gamma^* \boldsymbol{S}^{-1} \boldsymbol{\mu}_t$$

This is clearly the minimum as $\frac{\partial^2 L}{\partial \boldsymbol{\beta}_t^2} = \boldsymbol{S}$ is positive definite.

We call $\boldsymbol{\beta}_t^*$ a mean-variance efficient portfolio/allocation with respect to $r_{b,t}$.

Properties: By construction, the expected return of the optimal portfolio is

$$r_{b,t} = \boldsymbol{\beta}_t^{*\top} \boldsymbol{\mu}_t$$

Some algebra shows that the variance of the optimal portfolio is

$$\begin{aligned} \sigma_{b,t}^{*2} &= \boldsymbol{\beta}_t^{*\top} \boldsymbol{S} \boldsymbol{\beta}_t^* \\ &= \left(\lambda^* \boldsymbol{S}^{-1} \boldsymbol{e} + \gamma^* \boldsymbol{S}^{-1} \boldsymbol{\mu}_t\right)^\top \boldsymbol{\Sigma} \left(\lambda^* \boldsymbol{S}^{-1} \boldsymbol{e} + \gamma^* \boldsymbol{S}^{-1} \boldsymbol{\mu}_t\right) \\ &= \lambda^{*2} \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{e} + 2\lambda^* \gamma^* \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t + \gamma^{*2} \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t \\ &= \lambda^* \left(\lambda^* \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{e} + \gamma^* \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t\right) + \gamma^* \left(\lambda^* \boldsymbol{e}^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t + \gamma^* \boldsymbol{\mu}_t^\top \boldsymbol{S}^{-1} \boldsymbol{\mu}_t\right) \\ &= \lambda^* \left(\lambda^* A + \gamma^* B\right) + \gamma^* \left(\lambda^* B + \gamma^* C\right) \\ \textit{Equation (7)} \Longrightarrow &= \lambda^* \cdot 1 + \gamma^* \cdot r_{b,t} \\ \textit{Equation (7)} \Longrightarrow &= \frac{1}{\Delta}\left(C - B r_{b,t}\right) + \frac{1}{\Delta}\left(-B + A r_{b,t}\right) r_{b,t} \\ &= \frac{A r_{b,t}^2 - 2B r_{b,t} + C}{\Delta} \end{aligned}$$

Hence, we have $\sigma_{b,t}^{*2}$ as a quadratic function of $r_{b,t}$:

$$\sigma_{b,t}^{*2} = \frac{A r_{b,t}^2 - 2B r_{b,t} + C}{\Delta}$$

If we let $r_{b,t}$ vary, the efficient portfolios $\boldsymbol{\beta}_t^*$ form a a hyperbola in the $\left(\sigma^*\left(r_{b,t}\right), r_{b,t}\right)$ plane and a parabola in the $\left(\sigma^{*2}\left(r_{b,t}\right), r_{b,t}\right)$-plane, and we call it the efficient frontier.

Formulation 2:

Since minimizing the given loss function amounts to minimizing the MSE which falls under the context of linear regression, we can view this index tracking problem as a regression problem. Since the covaraince matrix is assumed to be constant over time, we can specify a

rolling window, say 30 days, and treat pairs $\{(r_{b,t-i}, \mathbf{r}_{t-i}))\}$, $i = 0, 1, \ldots, 29$ as samples. Let's denote $\mathbf{y}_t = (r_{b,t}, r_{b,t-1}, \ldots, r_{b,t-29}) \in \mathbb{R}^{30}$ as the regressand, and $\mathbf{R}_t = \begin{bmatrix} \mathbf{r}_t^\top \\ \vdots \\ \mathbf{r}_{t-29}^\top \end{bmatrix}$ as regressors. Then, for time t, we can assume

$$\mathbf{y}_t = \mathbf{R}_t \boldsymbol{\beta}_t + \boldsymbol{\epsilon}_t \quad \text{where} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_t)$$

By classical linear regression, we have

$$\boldsymbol{\beta}_t^* = \left(\mathbf{R}_t^\top \mathbf{R}_t\right)^{-1} \mathbf{R}_t^\top \mathbf{y}_t$$

By the Gauss-Markov theorem, $\boldsymbol{\beta}_t^*$ is the best unbiased linear predictor under the assumptions of classical linear regression.

(b) In this part, we no longer assume that covariances amongst stocks are time invariant. Propose a solution to minimizing the tracking error using the Kalman filter.

Solution: Leveraging the powerful Kalman filter, we can update our view on $\boldsymbol{\beta}_t$ from every single new observations on $r_{b,t}$.

State Equation & Observations Equation:

Set $\mathbf{x}_t = \boldsymbol{\beta}_t \in \mathbb{R}^{50}$, $\mathbf{F}_t = \mathbf{I}_{50} \in \mathbb{R}^{50\times 50}$, $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$, $\mathbf{Q}_t = \mathbf{0}$, $\mathbf{y}_t = r_{b,t} \in \mathbb{R}$, $\mathbf{H}_t = \mathbf{r}_t^\top \in \mathbb{R}^{1\times 50}$, $\mathbf{v}_t = v \sim \mathcal{N}(0, \mathbf{R}_t)$ and $\mathbf{R}_t = 0.3$.

Initial State:

Naturally, we set the initial state variable $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0)$ where $\boldsymbol{\mu}_0 = (1, \ldots, 1) \in \mathbb{R}^{50}$ and $\mathbf{P}_0 = \mathbf{I}_{50} * 0.1$.

Therefore, by our construction, we have the linear state space model:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_t \\ \mathbf{y}_t &= \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \end{aligned} \quad \text{where} \quad \mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{P}_0), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t), \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$$

Additionally, the initial state $\mathbf{x}_0$ and the noise terms $\mathbf{w}_t, \mathbf{v}_t$ are all assumed to be mutually independent.

(c) Download daily market data and create an example that illustrates your methodology. How does the tracking portfolio of the Kalman filter perform?

Solution: By comparing errors to $r_{b,t}$, the Kalman filter performs a lot better than the rolling window regression scheme.

For detailed codes, please check the attached Jupyter notebook.

Hint: For (c), compare your Kalman filter to a solution based on (a) where the covariance matrix is estimated on rolling windows. Can you match the performance of the Kalman filter with the simpler methodology in (a) by appropriately choosing the length of the rolling window?

References

[1] Kolm, Petter N. and Ritter, Gordon, On the Bayesian Interpretation of Black-Litterman (October 16, 2016). European Journal of Operational Research, Volume 258, Issue 2, 16 April 2017, Pages 564-572, Available at SSRN: https://ssrn.com/abstract=2853158.

Q4_Index_Tracking_Portfolio

October 28, 2023

[28]:
```
import numpy as np
import matplotlib.pyplot as plt
from pykalman import KalmanFilter
from numpy.linalg import inv
import scipy.stats as spst
import yfinance as yf
from pandas_datareader import data as pdr
import pandas as pd
```

1 Topic: Kalman filter's Application on Financial Problems — Index Tracking Portfolios

1.1 Get top 50 companies in S&P 500 Index

[25]:
```
# Read and print the stock tickers that make up S&P500
tickers = pd.read_html('https://en.wikipedia.org/wiki/
 ↪List_of_S%26P_500_companies')[0]['Symbol']
tickers.values
```

[25]:
```
array(['MMM', 'AOS', 'ABT', 'ABBV', 'ACN', 'ADM', 'ADBE', 'ADP', 'AES',
       'AFL', 'A', 'ABNB', 'APD', 'AKAM', 'ALK', 'ALB', 'ARE', 'ALGN',
       'ALLE', 'LNT', 'ALL', 'GOOGL', 'GOOG', 'MO', 'AMZN', 'AMCR', 'AMD',
       'AEE', 'AAL', 'AEP', 'AXP', 'AIG', 'AMT', 'AWK', 'AMP', 'AME',
       'AMGN', 'APH', 'ADI', 'ANSS', 'AON', 'APA', 'AAPL', 'AMAT', 'APTV',
       'ACGL', 'ANET', 'AJG', 'AIZ', 'T', 'ATO', 'ADSK', 'AZO', 'AVB',
       'AVY', 'AXON', 'BKR', 'BALL', 'BAC', 'BBWI', 'BAX', 'BDX', 'WRB',
       'BRK.B', 'BBY', 'BIO', 'TECH', 'BIIB', 'BLK', 'BX', 'BK', 'BA',
       'BKNG', 'BWA', 'BXP', 'BSX', 'BMY', 'AVGO', 'BR', 'BRO', 'BF.B',
       'BG', 'CHRW', 'CDNS', 'CZR', 'CPT', 'CPB', 'COF', 'CAH', 'KMX',
       'CCL', 'CARR', 'CTLT', 'CAT', 'CBOE', 'CBRE', 'CDW', 'CE', 'COR',
       'CNC', 'CNP', 'CDAY', 'CF', 'CRL', 'SCHW', 'CHTR', 'CVX', 'CMG',
       'CB', 'CHD', 'CI', 'CINF', 'CTAS', 'CSCO', 'C', 'CFG', 'CLX',
       'CME', 'CMS', 'KO', 'CTSH', 'CL', 'CMCSA', 'CMA', 'CAG', 'COP',
       'ED', 'STZ', 'CEG', 'COO', 'CPRT', 'GLW', 'CTVA', 'CSGP', 'COST',
       'CTRA', 'CCI', 'CSX', 'CMI', 'CVS', 'DHI', 'DHR', 'DRI', 'DVA',
       'DE', 'DAL', 'XRAY', 'DVN', 'DXCM', 'FANG', 'DLR', 'DFS', 'DIS',
```

```
'DG', 'DLTR', 'D', 'DPZ', 'DOV', 'DOW', 'DTE', 'DUK', 'DD', 'EMN',
'ETN', 'EBAY', 'ECL', 'EIX', 'EW', 'EA', 'ELV', 'LLY', 'EMR',
'ENPH', 'ETR', 'EOG', 'EPAM', 'EQT', 'EFX', 'EQIX', 'EQR', 'ESS',
'EL', 'ETSY', 'EG', 'EVRG', 'ES', 'EXC', 'EXPE', 'EXPD', 'EXR',
'XOM', 'FFIV', 'FDS', 'FICO', 'FAST', 'FRT', 'FDX', 'FITB', 'FSLR',
'FE', 'FIS', 'FI', 'FLT', 'FMC', 'F', 'FTNT', 'FTV', 'FOXA', 'FOX',
'BEN', 'FCX', 'GRMN', 'IT', 'GEHC', 'GEN', 'GNRC', 'GD', 'GE',
'GIS', 'GM', 'GPC', 'GILD', 'GL', 'GPN', 'GS', 'HAL', 'HIG', 'HAS',
'HCA', 'PEAK', 'HSIC', 'HSY', 'HES', 'HPE', 'HLT', 'HOLX', 'HD',
'HON', 'HRL', 'HST', 'HWM', 'HPQ', 'HUBB', 'HUM', 'HBAN', 'HII',
'IBM', 'IEX', 'IDXX', 'ITW', 'ILMN', 'INCY', 'IR', 'PODD', 'INTC',
'ICE', 'IFF', 'IP', 'IPG', 'INTU', 'ISRG', 'IVZ', 'INVH', 'IQV',
'IRM', 'JBHT', 'JKHY', 'J', 'JNJ', 'JCI', 'JPM', 'JNPR', 'K',
'KVUE', 'KDP', 'KEY', 'KEYS', 'KMB', 'KIM', 'KMI', 'KLAC', 'KHC',
'KR', 'LHX', 'LH', 'LRCX', 'LW', 'LVS', 'LDOS', 'LEN', 'LIN',
'LYV', 'LKQ', 'LMT', 'L', 'LOW', 'LULU', 'LYB', 'MTB', 'MRO',
'MPC', 'MKTX', 'MAR', 'MMC', 'MLM', 'MAS', 'MA', 'MTCH', 'MKC',
'MCD', 'MCK', 'MDT', 'MRK', 'META', 'MET', 'MTD', 'MGM', 'MCHP',
'MU', 'MSFT', 'MAA', 'MRNA', 'MHK', 'MOH', 'TAP', 'MDLZ', 'MPWR',
'MNST', 'MCO', 'MS', 'MOS', 'MSI', 'MSCI', 'NDAQ', 'NTAP', 'NFLX',
'NEM', 'NWSA', 'NWS', 'NEE', 'NKE', 'NI', 'NDSN', 'NSC', 'NTRS',
'NOC', 'NCLH', 'NRG', 'NUE', 'NVDA', 'NVR', 'NXPI', 'ORLY', 'OXY',
'ODFL', 'OMC', 'ON', 'OKE', 'ORCL', 'OTIS', 'PCAR', 'PKG', 'PANW',
'PARA', 'PH', 'PAYX', 'PAYC', 'PYPL', 'PNR', 'PEP', 'PFE', 'PCG',
'PM', 'PSX', 'PNW', 'PXD', 'PNC', 'POOL', 'PPG', 'PPL', 'PFG',
'PG', 'PGR', 'PLD', 'PRU', 'PEG', 'PTC', 'PSA', 'PHM', 'QRVO',
'PWR', 'QCOM', 'DGX', 'RL', 'RJF', 'RTX', 'O', 'REG', 'REGN', 'RF',
'RSG', 'RMD', 'RVTY', 'RHI', 'ROK', 'ROL', 'ROP', 'ROST', 'RCL',
'SPGI', 'CRM', 'SBAC', 'SLB', 'STX', 'SEE', 'SRE', 'NOW', 'SHW',
'SPG', 'SWKS', 'SJM', 'SNA', 'SEDG', 'SO', 'LUV', 'SWK', 'SBUX',
'STT', 'STLD', 'STE', 'SYK', 'SYF', 'SNPS', 'SYY', 'TMUS', 'TROW',
'TTWO', 'TPR', 'TRGP', 'TGT', 'TEL', 'TDY', 'TFX', 'TER', 'TSLA',
'TXN', 'TXT', 'TMO', 'TJX', 'TSCO', 'TT', 'TDG', 'TRV', 'TRMB',
'TFC', 'TYL', 'TSN', 'USB', 'UDR', 'ULTA', 'UNP', 'UAL', 'UPS',
'URI', 'UNH', 'UHS', 'VLO', 'VTR', 'VLTO', 'VRSN', 'VRSK', 'VZ',
'VRTX', 'VFC', 'VTRS', 'VICI', 'V', 'VMC', 'WAB', 'WBA', 'WMT',
'WBD', 'WM', 'WAT', 'WEC', 'WFC', 'WELL', 'WST', 'WDC', 'WRK',
'WY', 'WHR', 'WMB', 'WTW', 'GWW', 'WYNN', 'XEL', 'XYL', 'YUM',
'ZBRA', 'ZBH', 'ZION', 'ZTS'], dtype=object)
```

[54]:
```
d = []

for ticker in tickers:
    if ticker == 'BRK.B':
        ticker = 'BRK-B'
    if ticker == 'BF.B':
        ticker = 'BF-B'
```

```
    info = yf.Ticker(ticker).fast_info['marketCap']
    d +=[ (ticker, info) ]
```

[55]:
```
d.sort(key=lambda item: item[1])
top50_tickers = [item[0] for item in d[-50:]]
top50_tickers
```

[55]:
```
['PM',
 'DHR',
 'AMGN',
 'VZ',
 'WFC',
 'COP',
 'DIS',
 'NKE',
 'INTC',
 'AMD',
 'ABT',
 'TMUS',
 'CMCSA',
 'TMO',
 'PFE',
 'NFLX',
 'LIN',
 'ACN',
 'MCD',
 'CRM',
 'BAC',
 'CSCO',
 'PEP',
 'ADBE',
 'KO',
 'COST',
 'ABBV',
 'MRK',
 'CVX',
 'ORCL',
 'HD',
 'MA',
 'AVGO',
 'PG',
 'JNJ',
 'JPM',
 'XOM',
 'WMT',
 'V',
 'UNH',
```

```
 'LLY',
 'TSLA',
 'BRK-B',
 'META',
 'NVDA',
 'AMZN',
 'GOOGL',
 'GOOG',
 'MSFT',
 'AAPL']
```

1.2 Get top 50 companies' returns

[70]:
```python
df_top50_returns = pd.DataFrame()
start_date = '2013-1-1'
end_date = '2019-12-31'

for ticker in top50_tickers:
    data = yf.download(ticker, start_date, end_date)
    returns = (data['Close'] - data['Open']) / data['Open']
    df_top50_returns[ticker] = returns

df_top50_returns
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

[70]:

```
                  PM       DHR      AMGN        VZ       WFC       COP  \
Date
2013-01-02  0.024085  0.009129  0.020490 -0.004945  0.000571  0.001014
2013-01-03 -0.012422  0.011681 -0.010499 -0.006763 -0.006289  0.000507
2013-01-04  0.006046  0.000862  0.004402  0.005904  0.010118  0.009108
2013-01-07  0.000928 -0.028595  0.000452  0.009943 -0.004010 -0.007714
2013-01-08 -0.001508 -0.003628 -0.003279 -0.021122  0.000865 -0.010510
…                …         …         …         …         …         …
2019-12-23  0.005024 -0.003058 -0.003894 -0.009837  0.008811  0.014607
2019-12-24  0.001046  0.003271 -0.002018 -0.002929 -0.002410 -0.001390
2019-12-26 -0.002435 -0.000327 -0.003378  0.000163  0.001850 -0.001383
2019-12-27  0.003371 -0.002089 -0.002931  0.002280 -0.006815 -0.003374
2019-12-30 -0.007987 -0.000654 -0.004681 -0.007137 -0.010522 -0.007069

                 DIS       NKE      INTC       AMD  …       LLY      TSLA  \
Date                                                …
2013-01-02  0.005905 -0.010687  0.010875 -0.007843  … -0.008610  0.010286
2013-01-03  0.003134  0.008085 -0.003738 -0.011905  …  0.004039 -0.011654
2013-01-04  0.015765  0.007238 -0.009363  0.031872  …  0.009792 -0.011494
```

```
2013-01-07 -0.015263  0.003220 -0.001410  0.022989  …  0.003508 -0.013218
2013-01-08 -0.002359 -0.006070 -0.006594 -0.018382  …  0.008946 -0.023768
…               …       …       …       …  …       …       …
2019-12-23 -0.008430 -0.001896  0.002878  0.019740  … -0.009366  0.018068
2019-12-24  0.004911  0.003105  0.002700  0.009545  …  0.000305  0.016469
2019-12-26  0.002063  0.003687  0.006901 -0.007661  …  0.001069  0.007081
2019-12-27 -0.002054  0.005644  0.002838 -0.014301  … -0.000380 -0.010621
2019-12-30 -0.013585 -0.007288 -0.006168 -0.013437  … -0.001673 -0.032860

               BRK-B      META      NVDA      AMZN     GOOGL      GOOG  \
Date
2013-01-02  0.020587  0.020408  0.012739  0.004803  0.005324  0.005324
2013-01-03  0.005477 -0.003945  0.000786  0.004703 -0.001738 -0.001738
2013-01-04  0.001173  0.026776  0.031373  0.006095  0.011833  0.011833
2013-01-07 -0.003731  0.025444 -0.028158  0.020877 -0.000952 -0.000952
2013-01-08  0.004605 -0.015249 -0.024219 -0.002584 -0.003045 -0.003045
…               …       …       …       …       …       …
2019-12-23 -0.004987 -0.002516 -0.007068  0.002651 -0.005961 -0.005185
2019-12-24  0.001021 -0.005720  0.001763 -0.002564 -0.004281 -0.003663
2019-12-26  0.002346  0.010799  0.001801  0.037623  0.011823  0.010571
2019-12-27 -0.003174 -0.002732 -0.012219 -0.006968 -0.006862 -0.008144
2019-12-30 -0.001680 -0.016598 -0.015551 -0.014466 -0.012603 -0.010267

                MSFT      AAPL
Date
2013-01-02  0.013578 -0.008649
2013-01-03 -0.013753 -0.010550
2013-01-04 -0.019435 -0.018567
2013-01-07 -0.002988  0.003640
2013-01-08 -0.007477 -0.007369
…               …       …
2019-12-23 -0.004490  0.012369
2019-12-24 -0.000635 -0.001475
2019-12-26  0.007045  0.017871
2019-12-27 -0.003073 -0.004534
2019-12-30 -0.008806  0.007117

[1761 rows x 50 columns]
```

1.3 Get S&P500 index's returns

[71]:
```
df_sp500_returns = pd.DataFrame()
sp500_data = yf.download('SPY', start_date, end_date)
sp500_returns = (sp500_data['Close'] - sp500_data['Open']) / sp500_data['Open']
df_sp500_returns['SPY'] = sp500_returns
df_sp500_returns
```

```
[**********100%%**********]  1 of 1 completed
```

[71]:
```
                 SPY
Date
2013-01-02  0.006547
2013-01-03 -0.001781
2013-01-04  0.002740
2013-01-07  0.000823
2013-01-08 -0.001098
…                …
2019-12-23 -0.001151
2019-12-24 -0.000747
2019-12-26  0.004011
2019-12-27 -0.002718
2019-12-30 -0.005790

[1761 rows x 1 columns]
```

1.3.1 Rolling window Regression

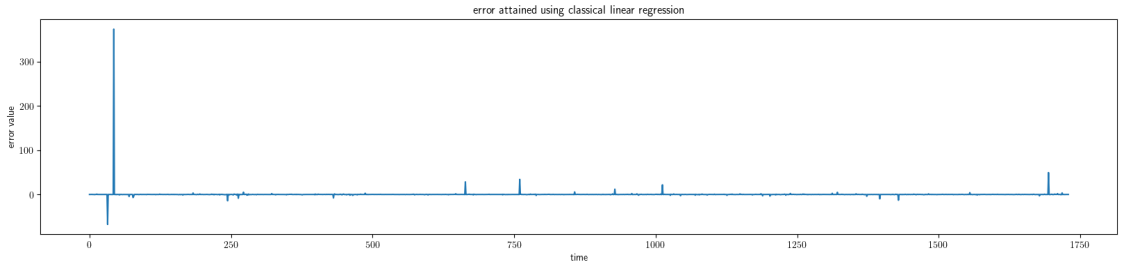
[201]:
```
# Specify the window size
window_size = 30

x_sequence_rolling = []

for t in range(window_size + 1, N):
    R = df_top50_returns.iloc[-t:-t + window_size].values

    # Compute the OLS Estimator
    x_t = inv(R.T@R) @ R.T @ df_sp500_returns.iloc[-t:-t + window_size].values.
 ↪reshape(-1, 1)
    x_sequence_rolling += [x_t]
```

Compute the error $\mathbf{r}_t^\top \beta_t^{OLS} - r_{b,t}$:

[202]:
```
errors_rolling = [df_top50_returns.iloc[-window_size-t-1].
 ↪values@x_sequence_rolling[t] - df_sp500_returns.iloc[-window_size-t-1].
 ↪values for t in range(len(x_sequence_rolling))] # start with -window_size-1
```

[207]:
```
fig = plt.figure(figsize=(20, 4))
plt.plot(np.arange(len(errors_rolling)), errors_rolling)
plt.xlabel('time')
plt.ylabel('error value')
plt.title('error attained using classical linear regression')
```

[207]:
```
Text(0.5, 1.0, 'error attained using classical linear regression')
```



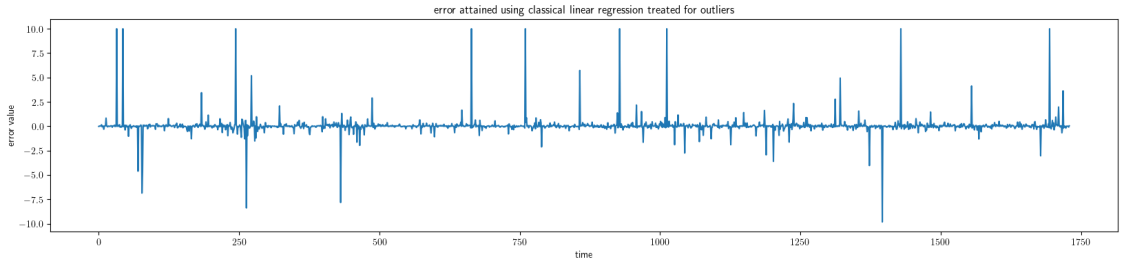
Delete outliers

[208]:
```
errors_rolling1 = [error if error < 1e1 and error > -1e1  else np.array([10])␣
 ↪for error in errors_rolling]
```

Have a look

[209]:
```
fig = plt.figure(figsize=(20, 4))
plt.plot(np.arange(len(errors_rolling1)), errors_rolling1)
plt.xlabel('time')
plt.ylabel('error value')
plt.title('error attained using classical linear regression treated for␣
 ↪outliers')
```

[209]:
```
Text(0.5, 1.0, 'error attained using classical linear regression treated for
outliers')
```



1.3.2 Kalman Filter

Leveraging the powerful Kalman filter, we can update our view on β_t from every single new observations on $r_{b,t}$.

State Equation & Observations Equation:

Set $\mathbf{x}_t = \beta_t \in \mathbb{R}^{50}$, $\mathbf{F}_t = \mathbf{I}_{50} \in \mathbb{R}^{50\times 50}$, $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$, $\mathbf{Q}_t = 0$, $\mathbf{y}_t = r_{b,t} \in \mathbb{R}$, $\mathbf{H}_t = \mathbf{r}_t^\top \in \mathbb{R}^{1\times 50}$, $\mathbf{v}_t = v \sim \mathcal{N}(0, \mathbf{R}_t)$ and $\mathbf{R}_t = 0.3$.

Initial State:

Naturally, we set the initial state variable $\mathbf{x}_0 \sim \mathcal{N}(\mu_0, \mathbf{P}_0)$ where $\mu_0 = (1, \dots, 1) \in \mathbb{R}^{50}$ and $\mathbf{P}_0 = \mathbf{I}_{50} * 0.1$.

Therefore, by our construction, we have the linear state space model:

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_t \\ \mathbf{y}_t &= \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t \end{aligned} \quad \text{where} \quad \mathbf{x}_0 \sim \mathcal{N}(\mu_0, \mathbf{P}_0), \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t), \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$$

Additionally, the initial state $\mathbf{x}_0$ and the noise terms $\mathbf{w}_t, \mathbf{v}_t$ are all assumed to be mutually independent.

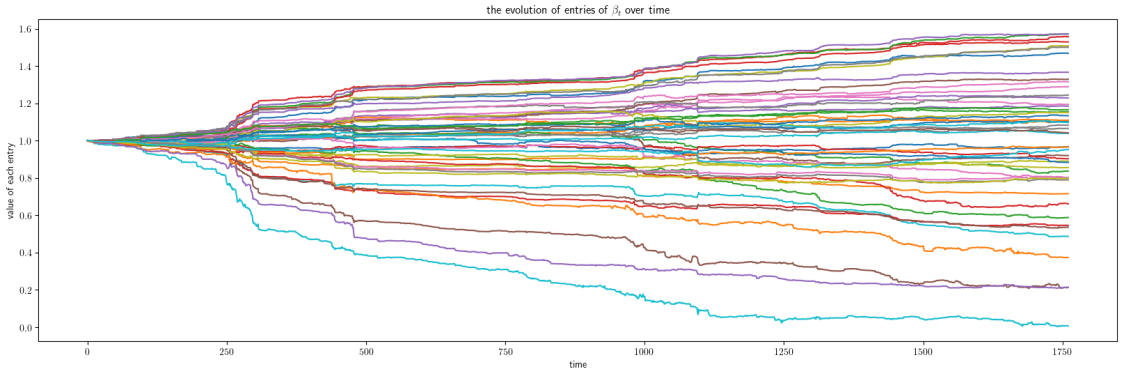
[178]:
```
N = df_sp500_returns.shape[0]
```

[179]:
```
# initial Guess
x_00 = np.array([1]*50).reshape(-1, 1)
P_00 = np.eye(50) * 0.1

# latent state variables sequence
x_sequence = [x_00]
P_sequence = [P_00]

# perturbations
Q_t = np.eye(50) * 0
R_t = 0.3

for t in range(1, N):
    F_t = np.eye(50)
    H_t = df_top50_returns.iloc[-t].values.reshape(1, -1)
    I = np.eye(50)

    # predict step
    x_t_tminus1 = F_t@x_sequence[t-1]
    P_t_tminus1 = F_t@P_sequence[t-1]@F_t.T + Q_t
    # print(x_t_tminus1.shape, P_t_tminus1.shape, H_t.shape)

    # update step
    S_t = H_t@P_t_tminus1@H_t.T + R_t
    K_t = P_t_tminus1@H_t.T@inv(S_t)

    x_tt = x_t_tminus1 + K_t@(df_sp500_returns.iloc[-t].values.reshape(1, 1) -␣
 ↪H_t@x_t_tminus1)
    P_tt = (I - H_t@K_t)@P_t_tminus1

    x_sequence += [x_tt]
    P_sequence += [P_tt]
```

Plot the entries of the latent state variable β_t:

[196]:
```
plt.rcParams['text.usetex'] = True
fig = plt.figure(figsize=(20,6))
for i in range(50):
    plt.plot(np.arange(N), x_seq.T[0][i])
plt.xlabel('time')
plt.ylabel('value of each entry')
plt.title('the evolution of entries of '+ r'$\beta_t$' + " over time")
```

[196]:
```
Text(0.5, 1.0, 'the evolution of entries of $\\beta_t$ over time')
```



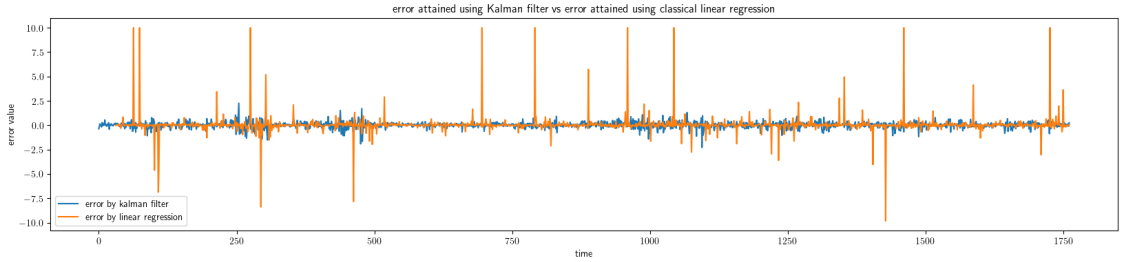
Compute the error $\mathbf{r}_t^\top \beta_t - r_{b,t}$:

[198]:
```
error_kalman = [df_top50_returns.iloc[-t-1].values@beta - df_sp500_returns.
↪iloc[-t-1].values for t, beta in enumerate(x_seq)]
```

1.4 Plot the error attained using Kalman filter and that attained using classical linear regression

[210]:
```
fig = plt.figure(figsize=(20, 4))
plt.plot(np.arange(len(error_kalman)), error_kalman, label='error by kalman␣
↪filter')
plt.plot(np.arange(window_size + 1, len(errors_rolling1)+window_size + 1),␣
↪errors_rolling1, label='error by linear regression')
plt.xlabel('time')
plt.ylabel('error value')
plt.title('error attained using Kalman filter vs error attained using classical␣
↪linear regression')
plt.legend()
```

[210]:
```
<matplotlib.legend.Legend at 0x17187aed0>
```



1.4.1 We noticed that Kalman filter performs significantly better!

[ ]:

2 ICA

Given that the principally-independent component analysis method which essentially was the truncated rank-K SVD of a matrix $\mathbf{X}$ followed by an ICA rotation of the left singular components:

$$\mathbf{X} \simeq \mathbf{USV}' = \mathbf{U}_I \mathbf{S}_I \mathbf{V}'_I, \tag{26}$$

where $\mathbf{U}_I = \mathbf{UA}_I$ with $\mathbf{A}_I = \arg\max_{\mathbf{A},\mathbf{A}'\mathbf{A}=I_K} \mid k_\ell(\mathbf{UA}) \mid$, $k_\ell(\mathbf{G})$ being any centered cumulant of order $\ell \geq 3$ which for all practical purposes can be considered a non-linear (activation) function applied to each of the entries of $\mathbf{G}$. Furthermore the matrix $\mathbf{V}_I$ was defined as $\mathbf{V}'_I := \mathbf{D}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV}'$ where $\mathbf{D}$ was chosen so that $\mathbf{V}_I$ has unimodular columns.

5 (a) **Show that D is a diagonal matrix.**

Given $\mathbf{U}_I = \mathbf{UA}_I$, $\mathbf{V}'_I := \mathbf{D}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV}'$ and $\mathbf{S}_I = \mathbf{SD}$, we have

$$\begin{aligned}
\mathbf{X} &\simeq \mathbf{USV}' \\
&= \mathbf{U}_I \mathbf{S}_I \mathbf{V}'_I \\
&= \mathbf{UA}_I\mathbf{SDD}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV}' \\
&= \mathbf{UA}_I\mathbf{I}_I\mathbf{A}_I\mathbf{SV}' \\
&= \mathbf{UA}_I\mathbf{A}_I\mathbf{SV}'.
\end{aligned}$$

In order to be consistent with the SVD, i.e., $\mathbf{X} \simeq \mathbf{USV}'$, we must have

$$\mathbf{A}_I\mathbf{A}_I = \mathbf{I}_I.$$

Hence, $\mathbf{A}_I$ must be a diagonal matrix with its diagonal entries $a_{ii} = \pm 1$ randomly.

Thus, for $\mathbf{V}'_I := \mathbf{D}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV}'$, we have known that $\mathbf{S}$, $\mathbf{S}^{-1}$ and $\mathbf{A}_I$ are now diagonal and we have

$$\mathbf{S}^{-1}\mathbf{A}_I\mathbf{S} = \mathbf{A}_I.$$

Thus, we have

$$\mathbf{V}'_I := \mathbf{D}^{-1}\mathbf{A}_I\mathbf{V}'.$$

Moreover, given that $\mathbf{V}'$ is orthonormal and $\mathbf{V}'_I$ is a matrix with unimodular columns, we can conclude that $\mathbf{D}^{-1}$ is another diagonal matrix in the same form as $\mathbf{A}_I$, i.e., a diagonal matrix with its diagonal entries $d_{ii} = \pm 1$ randomly, as required.

5 (b) **Show that $\mathbf{S}_I = \mathbf{SD}$ is diagonal such that $\mathbf{Tr}\left(\mathbf{S}_I^2\right) = \mathbf{Tr}\left(\mathbf{S}^2\right)$.**

As mentioned in (a), we have shown that $\mathbf{D}^{-1}$ is a diagonal matrix with its diagonal entries $d_{ii} = \pm 1$ randomly.

Given that the SVD, $\mathbf{X} \simeq \mathbf{USV}'$, $\mathbf{S}$ is also diagonal.

Thus, $\mathbf{S}_I = \mathbf{SD}$ since the product of diagonal matrices is also a diagonal matrix.

Denote the diagonal entries of $\mathbf{S}_I$, $\mathbf{S}$ and $\mathbf{D}$ are $s_{I,ii}$, s_{ii} and d_{ii} respectively. Since $d_{ii} = \pm 1$ randomly, we have

$$s_{I,ii} = \pm s_{ii}.$$

We can directly see that

$$s^2_{I,ii} = s^2_{ii}.$$

Hence, we have

$$\mathrm{Tr}(\mathbf{S}_I^2) = \mathrm{Tr}(\mathbf{S}^2),$$

as required.

5 (c) **Show that the method can be derived as the limit, $\lambda^2 \to 0$, of the optimization**

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{P},\mathbf{Q}:\mathbf{P'P}=\mathbf{diag}(\mathbf{Q'Q})=\mathbf{I}_k} \|\mathbf{X} - \mathbf{PRQ'}\|_F - \lambda^2 |k_\ell(\mathbf{P})|.$$

Given the optimization problem as

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{P},\mathbf{Q}:\mathbf{P'P}=\mathrm{diag}(\mathbf{Q'Q})=\mathbf{I}_k} \|\mathbf{X} - \mathbf{PRQ'}\|_F - \lambda^2 |k_\ell(\mathbf{P})|, \tag{27}$$

given $\mathbf{U}_I = \mathbf{UA}_I$ with $\mathbf{A}_I = \arg\max_{\mathbf{A},\mathbf{A'A}=I_K} \mid k_\ell(\mathbf{UA}) \mid$, $\mathbf{V}'_I := \mathbf{D}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV'}$, $\mathbf{S}_I = \mathbf{SD}$, and $\mathbf{P} = \mathbf{UA}$, we can rewrite this Lagrangian optimization problem as

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{A}_I,\mathbf{D}} \left\|\mathbf{X} - \mathbf{UA}_I\mathbf{SDD}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV'}\right\|_F, \tag{28}$$

$$\text{s.t.}\mathbf{A}_I = \arg \max_{\mathbf{A}_I} |k_\ell(\mathbf{UA})|, \text{ and } \mathbf{D} \text{ is diagonal.} \tag{29}$$

Since $\mathbf{D}$ is diagonal as shown in (a), this optimization problem is equivalent to

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{A_I}} \left\|\mathbf{X} - \mathbf{UA}_I\mathbf{SDD}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV'}\right\|_F, \tag{30}$$

$$\text{s.t. } \mathbf{A}_I = \arg \max_{\mathbf{A}_I} |k_\ell(\mathbf{UA})|. \tag{31}$$

Therefore, as previous proof, we can see that the only uncertain matrix $\mathbf{A}_I$ in $\mathbf{UA}_I\mathbf{SDD}^{-1}\mathbf{S}^{-1}\mathbf{A}_I\mathbf{SV'}$ is obtained by the constrain of the Lagrangian optimization problem, which is equivalent to say that we can derive the $\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I$ directly from the Lagrangian optimization problem if the constrain is fulfilled.

Hence, we can conclude that the method can be derived as the mentioned optimization problem, as required.

5 (d) **Show that an alternative objective function achieving the same result is**

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{P},\mathbf{Q}:\mathbf{P'P}=\mathbf{diag}(\mathbf{Q'Q})=\mathbf{I}_k} \|\mathbf{X} - \mathbf{PRQ'}\|_F - \lambda^2 J(\mathbf{P})$$

where $J[\mathbf{x}] := H[\mathbf{x_{gauss}}] - H[\mathbf{x}]$ is the negentropy and $J(\mathbf{P})$ is the sum of the negentropies of all the columns of P.

Following the same thought in (c), we can rewrite the given optimization problem in a Lagrangian optimization problem

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{A},\mathbf{D}} \left\|\mathbf{X} - \mathbf{U} \arg \max_{\mathbf{A},\mathbf{A'A}=I_K} \mid k_\ell(\mathbf{UA}) \mid \mathbf{SDD}^{-1}\mathbf{S}^{-1} \arg \max_{\mathbf{A},\mathbf{A'A}=I_K} \mid k_\ell(\mathbf{UA}) \mid \mathbf{SV'}\right\|_F, \tag{32}$$

$$\text{s.t. } J[\mathbf{UA}] = H[\mathrm{Column}_{\mathbf{UA}_{\mathrm{gauss}}}] - H[\mathrm{Column}_{\mathbf{UA}}] \geq 0, \text{ and } \mathbf{D} \text{ is diagonal.} \tag{33}$$

Similar to above, we can furthur simplify this optimization problem as

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg \min_{\mathbf{A}} \left\|\mathbf{X} - \mathbf{U} \arg \max_{\mathbf{A},\mathbf{A'A}=I_K} \mid k_\ell(\mathbf{UA}) \mid \mathbf{SDD}^{-1}\mathbf{S}^{-1} \arg \max_{\mathbf{A},\mathbf{A'A}=I_K} \mid k_\ell(\mathbf{UA}) \mid \mathbf{SV'}\right\|_F, \tag{34}$$

$$\text{s.t. } J[\mathbf{UA}] = H[\mathrm{Column}_{\mathbf{UA}_{\mathrm{gauss}}}] - H[\mathrm{Column}_{\mathbf{UA}}] \geq 0. \tag{35}$$

Where, the constrain

$$J[\mathbf{UA}] = H[\mathrm{Column}_{\mathbf{UA}_{\mathrm{gauss}}}] - H[\mathrm{Column}_{\mathbf{UA}}] \geq 0$$

is equivalent to

$$\mathbb{E}[\ln(\text{Column}_{\mathbf{UA}})] \geq \mathbb{E}[\ln(\text{Column}_{\mathbf{UA}_{\text{gauss}}})] \tag{36}$$

$$\Leftrightarrow \mathbb{E}[\ln(\text{Column}_{\mathbf{UA}})] = \sup\left\{\mathbb{E}[\ln(\text{Column}_{\mathbf{UA}_{\text{gauss}}})]\right\}. \tag{37}$$

by the definition and properties of negentropy.

Thus, from the constrain, we can have

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \mathbb{E}[\ln(\text{Column}_{\mathbf{UA}_{\text{gauss}}})] \tag{38}$$

$$\Leftrightarrow \mathbf{A}^* = \arg\max_{\mathbf{A}} \mathbb{E}[\ln(\text{Column}_{\mathbf{UA}})] \tag{39}$$

$$\Leftrightarrow \mathbf{A}^* = \arg\max_{\mathbf{A}} J(UA) \tag{40}$$

$$\Leftrightarrow \mathbf{A}^* = \arg\max_{\mathbf{A}} J(P). \tag{41}$$

Thus, the Lagrangian optimization problem can finally be written as

$$\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I = \arg\min_{\mathbf{A}} \left\| \mathbf{X} - \mathbf{U} \arg\max_{\mathbf{A}, \mathbf{A}'\mathbf{A}=I_K} \mid k_\ell\left(\mathbf{UA}\right) \mid \mathbf{SDD}^{-1}\mathbf{S}^{-1} \arg\max_{\mathbf{A}, \mathbf{A}'\mathbf{A}=I_K} \mid k_\ell\left(\mathbf{UA}\right) \mid \mathbf{SV}' \right\|_F, \tag{42}$$

$$\text{s.t. } \mathbf{A}^* = \arg\max_{\mathbf{A}} J(P). \tag{43}$$

Similar to (c), we can see that the only uncertain matrix $\mathbf{A}$ in $\mathbf{X} - \mathbf{U}\arg\max_{\mathbf{A},\mathbf{A}'\mathbf{A}=I_K} \mid k_\ell\left(\mathbf{UA}\right) \mid \mathbf{SDD}^{-1}\mathbf{S}^{-1}\arg\max_{\mathbf{A},\mathbf{A}'\mathbf{A}=I_K} \mid k_\ell\left(\mathbf{UA}\right) \mid \mathbf{SV}'$ is obtained by the constrain of the Lagrangian optimization problem, which is equivalent to say that we can derive the $\mathbf{U}_I, \mathbf{S}_I, \mathbf{V}_I$ directly from the Lagrangian optimization problem if the constrain is fulfilled.

Hence, we can conclude that the method can be derived as the mentioned optimization problem, as required.