Ultra-High-Frequency Multivariate Covariance Estimation of Noisy and Asynchronous Asset Returns

And its Application to Chinese Stock Market

DUAN, YUNAN
Nanjing University

May 31, 2017

Abstract

In order to find a positive-semidefinite estimtor of multivariate covariance matrices of noisy and aynchronous asset returns, we adopt and compare two methodologies: Kalman smoother and expectation maximization (KEM) algorithm and Gibbs sampler algorithm in Bayesian framework (Bayesian Method), where in both cases we model microstructure noise as measurement errors and asynchronous trading effects as missing observations in an otherwise synchronous series. We then compare the performance of these two estimators along with another benchmark estimating approach using extensive simulation study. The empirical application to a high-frequency dataset of Chinese stock market confirms our simulation study.

Acknowledgement

Here I would like to thank my supervisor, Dr. Qu, Hui, for her generous help and constant encouragement through the making of my undergraduate thesis. I would also like to thank my parents, who have always been my greatest supporters, both financially and spiritually.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The importance of multivariate covariance matrix of asset returns in risk management, asset allocation and option pricing makes it a subject of active research. Merton (1980)[22] and its following work showed that available intra-day price information can be exploited in the construction of covariance measures. But there are still three concerns when dealing with multivariate high-frequency price processes. First, asset prices in general are subject to the influence of market microstructure noise, which makes the statistical inference of price properties more complicated. Second, the so called asynchronous trading effect is consistent with high-frequency data and strongly affects the estimation of covariance matrix. It was first proposed by Epps (1979)[27], who discovered that the realized covariance computed on a artificially regularized grid has an increasing bias as the sampling frequency increases. In the literature, Palandri (2006)[2] used so called multiscale subsampling method to improve the convergency properties; Zhang (2011)[11] discussed microstructure noise and proposed a Two-Scale Realized Covariance (TSRC) estimator; Mancino and Sanfelici (2011)[12] applied the multivariate Fourier method to tackle the asynchronous trading problem. Third, we still need to guarantee the positive semidefiniteness (psd) of our estimator. Barndorff-Neilsen et al. (2011) [17]proposed a Multivariate Realized Kernel estimator to ensure our desired property. However, this scheme could result in discarding information for liquid assets.

Driven by the need of a multivariate covariance estimator that is both positive semidefinite and robust to microstructure noise and asynchronicity, we adopt and compare two methodologies. They are both based on the assumption that microstructure noise can be modeled as a measurement error on the true price process, which leads us to a state space model with the transition equation describing the dynamics of the true price process and the observation equation modeling the observed price process that is contaminated by the microstructure noise. They are also based on the idea

of viewing asynchronous time series as synchronous series with missing observations. The combination of these two assumptions leads to our first methodology, proposed by Corsi, Peluso and Audrino (2015)[6]: Kalman filter recursion and expectation maximization (EM) algorithm (we term this method as KEM in this paper). In expectation step, we reconstruct the true and synchronized price processes using Kalman filter recursion; in maximization step, we searched for parameter values (in this case the covariance matrices) which maximize the expected likelihood of the reconstructed series. By making use of all trade information of all assets, this KEM estimator has an advantage of exploiting all information contained in the price processes, while guaranteeing the estimator is psd. On the other hand, Peluso, Corsi and Mira (2015)[26] viewed the latent price series and the covariance matrix as parameters in the Bayesian framework and then used Gibbs sampler and Markov Chain Monte Carlo algorithm to obtain our estimation. In Gibbs sampler, we choose an Inverse Wishart distribution as the prior distribution and since covariance matrix is drawn from it, this method preserves its positive semidefiniteness natrually. Like KEM, our Bayesian estimator is robust to microstructure noise and fully incorporate the price information contained in the available asset processes. It will also be shown that since MCMC tends to perform poorly in the nonstationary Gaussian Bayesian Model, the speed of convergency and the accuracy of our Bayesian estimator is not as good as KEM estimator.

In section 2 we will propose the state space model and two estimating methodologies. Section 3 presents the simulation study which compares the KEM estimator with the Bayesian estimator, along with a benchmark estimator proposed by Hayashi and Yoshida (2005)[28], in a broad range of initial settings. Section 4 is the empirical study of our two estimators applied to model and forecast Chinese stock market. Section 5 concludes.

Chapter 2

Model and Methodology

We start by introducing a model considering both asynchronicity and microstructure noise:

$$Y_t = X_t + \sqrt{R}dB_t \tag{2.1}$$
$$dX_t = \mu_t + \sqrt{Q}dW_t \tag{2.2}$$

where Y_t is a d-dimensional observed log-price process, X_t is a d-dimensional true log-price process, μ is a vector of predictable locally bounded elements, W_t is the d-dimensional vector of independent Brownian motions, B_t is a d-dimensional Brownian motion, $B_t \perp W_t$, Q is the constant diffusion coefficient matrix, R is the constant variance of the microstructure noise term.

This state space model may look too simplistic to model this complicated problem for two reasons. First, here we do not consider the jumps in the price process; however, the presence of jumps can be frequent in practice. Our methodology is to first remove jumps using a simple filter and then apply models to the resulting filtered series, because our methodology focuses on the continuous part of the quadratic variation o the latent price process. Second, We recognize that the volatility Q = Q_t is a time-varying process, but in simulation where the constant volatility matrix is allowed our estimation is very accurate. It is supported by Xiu (2010) [4] for the univariate case and Liu and Tang (2012) and Shephard and Xiu(2012) for the multivariate case, all of which show that the QMLE of the volatility of a missepcified model with constant volatility remains consistent and optimal in terms of its rate of convergency under fairly general assumptions.

We discretize this model on an ultra-high-frequency grid with constant diffusion coefficient and zero drift. The our discrete time model is:

$$Y_t = X_t + \eta_t, \quad \eta_t \sim N(0, R) \tag{2.3}$$
$$X_t = X_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, Q) \tag{2.4}$$

where X_t and Y_t are defined as in (2.1) and (2.2), Y_t is partitioned in its observed and missing components $[Y_t^o, Y_t^m]$ and η_t and ϵ_t are assumed to be uncorrelated and mutually independent errors. We also assume that the matrix R to be diagonal, which means that the microstructure noises are uncorrelated across assets.

The discretized model is a simple linear state space model, known as a local level model, consisting of the observation equation (2.3) and the state equation (2.4). The model can also be viewed as a particular case of a dynamic linear model, proposed by West and Harrison (1997)[13]. This model is also called local level model, or random walk plus noise model or steady forecasting model. It is characterized by the general form [A,C,R,Q], respectively the observation matrix, transition matrix, observation error variance matrix and transition error variance matrix. Our model is a special case of the DLM with matrices [I_d,I_d,R,Q].

2.1 Kalman-EM Estimation

Following Corsi, Peluso and Audrino (2015)[6], the general state space model (2.1) and (2.2) accounts separately for uncertainties defined by the measurement error η_t and the noise ϵ_t. If one knows the values for the parameters Q and R, then the conventional Kalman smoothing and forecasting estimators can be calculated as conditional expectations and will have minimum mean square error. If the parameters Q and R are not specified in advance, they are estimated by maximum likelihood using the EM(expectation maximization) algorithm described in Dempster et al. (1977)[1].

Here in our problem, we have already partitioned Y in its observed and missing components $[Y^o, Y^m]$. We then define the hidden variables $\tilde{X} = [X, Y^m]$. Our goal in the EM algorithm is to maximize, with respect to Q, R, $\tilde{X}$, Y^o, the joint log likelihood:

$$\begin{aligned} logL = logP(\tilde{X}, Q, R|Y^o) \propto -\frac{T}{2}ln|Q| - \frac{1}{2}\sum_{t=1}^{T}(X_t - X_{t-1})'Q^{-1}(X_t - X_{t-1}) \\ -\frac{T}{2}ln|R| - \frac{1}{2}\sum_{t=1}^{T}(Y_t - X_t)'R^{-1}(Y_t - X_t) \end{aligned} \tag{2.5}$$

Then we consider applying the EM algorithm conditionally on Y^o, i.e., define the estimated parameters at the $(k+1)$ step as $\tilde{X}_{k+1}$, Q_{k+1}, R_{k+1} which maximize

$$E_r(logL|Y^o) \tag{2.6}$$

where E_r denotes the conditional expectation containing the k step values $\tilde{X}_k$, Q_k, R_k. The EM algorithm iterates between such two steps with respect to $\tilde{X}$ and Q, R, holding the other fixed. In the language of mathematics, we iterate between the following two steps in EM algorithm:

$$\begin{aligned} \text{E step: } & X_{k+1}, Y_{k+1} = arg\ max_{X,Y} E[logP(Q, R|X_k, Y_k)] \\ \text{M step: } & Q_{k+1}, R_{k+1} = arg\ max_{Q_k,R_k} E[logP(X, Y|Q_k, R_k)] \end{aligned}$$

Specifically, in the E step, the distribution of $\tilde{X}$ that maximizes (2.5), holding Q and R fixed, is exactly the conditional distribution of $\tilde{X}$, i.e., $P(\tilde{X}|Y^o, Q, R)$, which is the standard output of the Kalman filter. Thus we simply perform the Kalman filtering and smoothing recursion in the E step, which is reported in the Appendix A (Shumway and Stoffer, 1982[21]). In the maximization step, given the estimated distribution of $\tilde{X}$ in the E step, we continue to maximize (2.5) using multivariate regression approach, which gives us the following estimated values of Q and R at iteration (k+1):

$$\begin{aligned} Q_{k+1} = (T-1)^{-1}[\sum_{t=2}^{T}(V_t^s + X_t^s X_t^{s'}) - (\sum_{t=2}^{T}(VL_t^s + X_t^s X_t^{s'})) \\ (\sum_{t=2}^{T}(VL_{t-1}^s + X_{t-1}^s X_{t-1}^{s'}))^{-1}(\sum_{t=2}^{T}(VL_t^s + X_t^s X_t^{s'}))'] \end{aligned} \tag{2.7}$$

$$R_{k+1} = T^{-1}\sum_{t=1}^{T} D_t \begin{bmatrix} (Y_t^o - X_t^{0,s})(Y_t^o - X_t^{0,s})' + V_t^{o,s} & 0 \\ 0 & R_t^m(k) \end{bmatrix} D_t' \tag{2.8}$$

where X_t^s is the smoothed log-price process and $X_t^{0,s}$ are the components of X_t^s corresponding to Y_t^o; V_t^s is the variance of the smoothing error, $V_t^{o,s}$ is the submatrix of V_t^s corresponding to Y_t^o, and VL_t^s is the one-lag autocovariance of the smoothing error (For expressions of these quantities please refer to Appendix A);R_t^m is the submatrix of R corresponding to Y_t^m and D_t is a permutation matrix.

For convergency purpose, we keep track of the value of the log likelihood function at each iteration using the form derived by Gupta and Mehra (1974)[15]:

$$\begin{aligned} logL = \sum_t [-\frac{1}{2}ln(|V_t^{o,p} + R_t^o|) - \frac{1}{2}(Y_t^o - X_{t-1}^{o,f})'(V_t^{o,p} + R_t^o)^{-1} \\ (Y_t^o - X_{t-1}^{o,f})] \end{aligned} \tag{2.9}$$

Procedures for EM algorithm in this problem are as follows:

1. Calculate the hidden variables $\tilde{X}$ using Kalman filter and smoother recursion reported in Appendix A given the value of initial parameters Q and R;

2. Use the estimated hidden variables in 1 and equation (2.7) and (2.8) to calculate Q and R;
3. Repeat 1 and 2 above until convergency, i.e., the log likelihood function is stable.

Dempster et al. (1977)[1] shows that this procedure yields non-decreasing likelihoods. Although it seems like that we are maximizing the wrong likelihood (the expected likelihood of the joint date instead of that of the observed data), EM algorithm indeed guarantees to increase the correct likelihood. EM also has computational benefits: it only does simple matrix multiplications at each step, and it can be easily extended to the case of missing data since missing observations and the state space variables can be jointly treated as hidden variables.

2.2 Bayesian Estimation

Let's change our perspective in viewing this model. Recall that in the E step of KEM estimation, we are to estimate $\tilde{X}$ assuming the prior values of Q and R. This assumption also falls into the framework of Bayesian inference methodology. Furthermore, an algorithm called Gibbs sampler is used here to generate samples of $\tilde{X}$, Q and R from the conditional pdfs, where Q is the estimated covariance matrix. Theoretically, Gibbs sampler is presented as follows (Hogg, Mckean and Craig 2013)[20]:

Theorem 2.2.1 *Let m be a positive integer, and let X_0 be given as an initial value. Then for $i = 1, 2, 3, ..., m$, we generate random variables by the following algorithm:*

$$\begin{aligned} &1.\ \text{Generate } Y_i|X_{i-1} \sim f(y|x) \\ &2.\ \text{Generate } X_i|Y_i \sim f(x|y) \end{aligned}$$

Then $Y_i \to Y \sim f_Y(y)$ in distribution, $X_i \to X \sim f_X(x)$ in distribution, as $i \to \infty$.

The proof of this theorem can be found in Lehmann and Casella (1998)[5].

Following this procedure, Peluso, Corsi and Mira (2015)[26] construct the Bayesian estimation methodology using so called augmented Gibbs sampler, which samples the data twice by considering both the missing observations and the latent process as additional parameters. Therefore, this algorithm samples X, R, Q, Y^m from their full conditional distribution. As for the prior distributions, we sample Q and R from the Inverse Wishart distribution, which naturally preserves the positive semi-definiteness of the

estimated Q and R (Eaton 2007, chapter 8)[14]:

$$p(Q|R,Y,X) \propto IW(SS_Q,T), \quad SS_Q = \sum_{t=1}^{T}(X_t - X_{t-1})(X_t - X_{t-1})' \quad (2.10)$$

$$p(R|Q,Y,X) \propto IW(SS_R,T), \quad SS_R = \sum_{t=1}^{T}(Y_t - X_t)(Y_t - X_t)' \quad (2.11)$$

For the missing observations, we sample them from the conditional distribution:

$$p(Y^m|Y^o,X,R,Q) \propto N(X_{i,t} + R_{i.}R_{-i}^{-1}(Y_{-1,t} - X_{-i,t}), \omega_i^2 - R_{i.}R_{-i}^{-1}R_{i.}') \quad (2.12)$$

where $Y_{-i,s:t}$ is the matrix of log-prices for assets j, $\forall j \neq i$ and from time s to t; $X_{-i,s:t}$ is similar. R_{-i} is obtained from R by dropping the row and column corresponding t asset i, and $R_{i.}$ is the i-th row of R without its i-th element.

Finally, we draw log latent price process from its conditional distribution using FFBS (Forward Filtering Backward Simulation) algorithm, a Kalman smoother in which the smoothing recursions are replaced by simulations of the latent process:

$$p(X|Y,Q,R) = \prod_{t=1}^{T} p(X_t|X_{t+1:T},Y) \quad (2.13)$$

where the last factor in the product is $p(X_T|Y)$, that is, the filtering distribution of X_T $N(X_t^f, V_t^f)$ with X_t^f the filtered latent log-price and V_t^f its covariance matrix. For $t = T-1, T-2, ..., 1$, we continue to draw X_t from $p(X_t|X_{t+1:T},Y) \propto N(X_t^f + V_t^f(V_t^p)^{-1}(X_{t+1} - X_t^p), V_t^f(V_t^p)^{-1}V_t^f)$, where X_t^p is the predicted latent log-price.

In summary, the augmented Gibbs sampler iterates among the following four steps:

1. Draw the covariance matrix Q from an Inverse Wishart distribution: $IW(SS_Q,T)$, with $SS_Q = \sum_{t=1}^{T}(X_t - X_{t-1})(X_t - X_{t-1})'$
2. Draw the covariance matrix R from an Inverse Wishart distribution: $IW(SS_R,T)$, with $SS_R = \sum_{t=1}^{T}(Y_t - X_t)(Y_t - X_t)'$
3. Impute, for $i = 1, 2, ..., d$ and $t \in t^{miss}$, the missing observations Y^m by drawing from $N(X_{i,t} + R_{i.}R_{-i}^{-1}(Y_{-1,t} - X_{-i,t}), \omega_i^2 - R_{i.}R_{-i}^{-1}R_{i.}')$
4. Extract X from its full conditional, $\prod_{t=1}^{T} N(X_t^f + V_t^f(V_t^p)^{-1}(X_{t+1} - X_t^p), V_t^f(V_t^p)^{-1}V_t^f)$,using FFBS algorithm.

Note that here we choose to perform Bayesian statistical inference using the Monte Carlo simulation, and Gibbs sampler has the Markovian property. Thus, here we actually perform Markov Chain Monte Carlo (MCMC)

simulation. One of the notorious properties of MCMC is that it usually exhibits slow convergence speed. The first reason is that the different points of the latent process X are strongly dependent on each other. Second, the latent process and the covariance matirx Q are also strongly dependent, especially in large sample settings as ours. This is known in the literature as RobertsÍCStramer critique (Roberts and Stramer, 2001) and is formalized by noting that:

$$\mathbb{P}lim_{\Delta t \to 0} \Sigma_{t=1}^{T} (X_t - X_{t-1})'(X_t - X_{t-1}) \; = \; Q \tag{2.14}$$

This asymptotic relationship between Q and X causes, in the limit, the Markov Chain to be reducible, that is unable to escape from the current value.

This problem also requires the discussion of the starting value. Although different starting values do not affect the validity of convergence once the chain reaches its stationarity, i.e., the converging property of this chain is independent of the choice of starting values, here we use the HY estimator as the starting value to speed up the process. One can find useful discussions in Lehmann and Casella (1998)[5].

Chapter 3

Simulation Study

In this section we compare the performance of our two proposed estimators, along with the pairwise Hayashi and Yoshida (2005)[28] estimator (HY). HY estimator is the cross-product of all returns with at least a partial overlapping. HY is robust to the asynchronicity but not to the microstructure noise. This property will be examined fully in our following results.

We mentioned earlier in this paper that the KEM and Bayesian estimator are theoretically psd. Here we still need to guarantee that HY estimator is psd when we use it. We could achieve this goal by adopting a projection procedure which is able to project HY into space of psd matrices without dramatically changing it. This procedure can be found in Hautsch, Kyj and Oomen (2012)[16], and here we briefly present it: let $\lambda_{(i)}$ denote i-th eigenvalue of covariance matrix M, then all the eigenvalues below λ_M are converted to

$$\lambda^*_{(i)} = \frac{\sum_{i=k+1}^{d} max(0, \lambda_{(i)})}{d-k}, \quad i = k+1, ..., d \tag{3.1}$$

where $\lambda_M = (1 - \frac{\lambda_{(1)}}{d})(1 + \sqrt{\frac{d}{N}})^2$ with $\lambda_{(1)}$ being the highest eigenvalue, d is the number of assets, N is the sample size and $k = \sum_{i=1}^{d} I(\lambda_{(i)} \geq \lambda_M)$ with I being the indicator function.

3.1 Data Generating Process

The data generating process (DGP) is a stochastic volatility model proposed by Heston (1993)[24]. For i=1,2,...,d and t=1,2,...,T,

$$dX_{i,t} = \sigma_{i,t} dW_{i,t} \tag{3.2}$$

$$d\sigma^2_{i,t} = k_i(\tilde{\sigma}^2_i - \sigma^2_{i,t}) + s_i \sigma_{i,t} dB_{i,t} \tag{3.3}$$

where $E(dW_{i,t}, dW_{j,t)} = \rho_{i,j}dt$, $E(dW_{i,t}, dB_{j,t}) = \delta_{i,j}\pi_i dt$, $X_{i,t}$is the log latent price of asset i at time t, $W_{i,t}$ and $B_{i,t}$ are Brownian motions with correlation structure defined above, k_i and s_i are, respectively, the speed of the mean reversion and the volatility of the variance process $\sigma^2_{i,t}$, $\tilde{\sigma}^2_i$ is the asymptotic mean of $\sigma^2_{i,t}$.

Based on the log true process X_t we just generated, the observed log price process Y_t is as below:

$$Y_t = X_t + T_t \tag{3.4}$$

where $T_t \sim t_d(x; R, v) \propto (1 + \frac{1}{v}x'\Omega^{-1}x)^{-\frac{v+d}{2}}$, a d-dimentional scaled student's t-distribution with scale matrix R and v degrees of freedom. Finally, to reproduce the asynchronicity of intraday log prices, we randomly cancel observations in Y_t.

Here we apply Euler-Maruyama dicretization scheme [3] to equation (3.2) and (3.3) to generate the data. For a general form of a stochastic differential equation

$$\begin{aligned} dX(t) = f(X(t))dt + g(X(t))dW(t), \\ X(0) = X_0, \\ 0 \leq t \leq T \end{aligned} \tag{3.5}$$

where W denote a Brownian motion, f and g are given functions. Let $\Delta t = T/L$(for some positive integer L) and $\tau_j = j\Delta t$ (j=0,1,...,L), the Euler-Maruyama formula gives us(see Appendix B for details):

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W(\tau_j) - W(\tau_{j-1})), \quad j = 1, 2, ..., L \tag{3.6}$$

We then apply the deduction from equation (3.5) to (3.6) to equation (3.2) and (3.3):

$$X_{i,j} = X_{i,j-1} + \sigma_{i,j-1}(W(\tau_j) - W(\tau_{j-1})) \tag{3.7}$$

$$\sigma^2_{i,j} = \sigma^2_{i,j-1} + k_i(\tilde{\sigma}^2_i - \sigma^2_{i,j-1}) + s_i\sigma_{i,j-1}(B(\tau_j) - B(\tau_{j-1})) \tag{3.8}$$

In MATLAB, we generate our desired dataset by iterating between equation (3.7) and (3.8) and following data-generating procedures. Table 3.1 shows an example of the generated data process:

Another data generating process is a multifactor extension of previous model with two volatility factors having different persistence. In addition, a deterministic intraday volatility component is added to the stochastic process in order to reproduce the observed U-shape intraday pattern. This model, known as Double Heston model, is proposed by Christoffersen, Heston and Jacobs (2009)[18], and it is able to improve the quality of fit on

Table 3.1: The generated log price process used in our simulation

4.6082	0	3.0921	0	5.6189	3.5745	0	...
3.2035	3.5497	2.4426	0	3.7247	1.4533	0	...
4.6429	4.4845	0	0	4.5522	0	5.0176	...
4.9801	5.4604	0	0	6.7232	6.7306	3.4442	...
6.0398	3.3566	4.9029	0	5.0194	0	3.3632	...
...	...	...	...	...	...	...	...

empirical data compared to the original Heston model. We would like to point out that dataset generated by this model, although theoretically mimic the pattern of price process more vividly, does not change our simulation results reported in the next section.

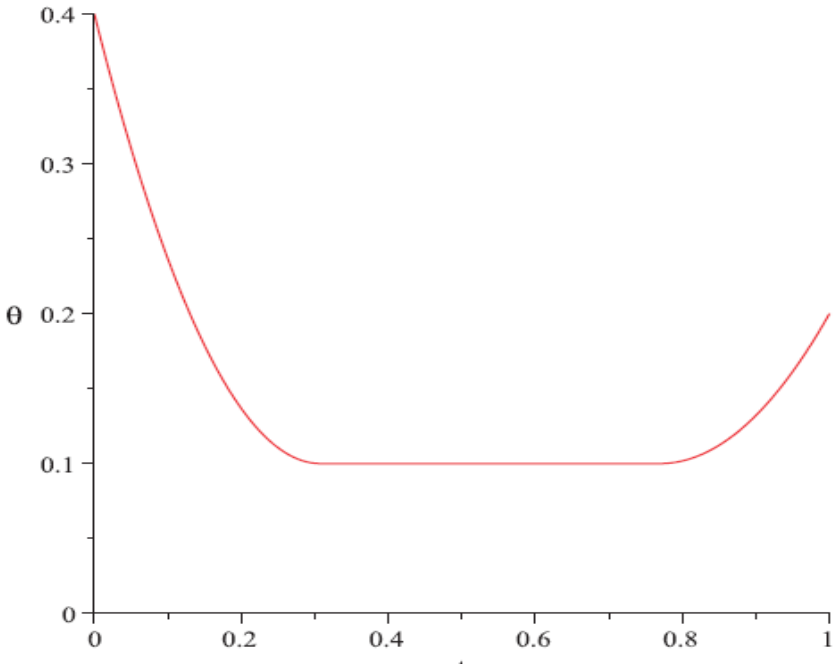

Figure 3.1: Volatility factor deterministic intraday trend

3.2 Simulation Settings

We simulate the covariance matrix of 10 assets for M=500 simulated sample paths. All simulations are initialized from $P_0 = log([100, 40, 60, 80, 40, 20,$

$90, 30, 50, 60]$). Here and in the next section of empirical study, we reconstruct the synchronized latent price process at the frequency of one second, i.e., sample size $T = 4 * 60 * 60 = 14400$. Parameters are set to be $\tilde{\sigma}^2, k, s = 0.01, 2, 0.1$ (Khaled and Samia 2010)[9]. The estimated covariance matrix are computed at the daily horizon and compared with the realized covariance matrix of the latent process for that day. Recall that Y_t denote the vector of true latent price process, then the realized covariance (RV) is defined (Hansen and Lunde 2006)[19] as:

$$RV = \Sigma_{i=2}^{T}(Y_i - Y_{i-1}) * (Y_i - Y_{i-1})' \tag{3.9}$$

We choose the Frobenius norm and the Stein norm as our measure of difference between the estimated and realized covariance matrix, as they preserve consistency between orderings in a multivariate volatility model (Laurent et al. 2013[25]). The lower the Frobenius distance or the Stein distance, the better our estimator performs. The Frobenius norm is defined as:

$$Frob = \sqrt{\sum_{i=1}^{d}\sum_{i=1}^{d}(\hat{\sigma_{ij}} - \sigma_{ij})^2} \tag{3.10}$$

where $\hat{\sigma_{ij}}$ is the ijth element of the estimated Q and σ_{ij} is the ijth element of the RV of the true price process. The Stein norm is defined as:

$$stein = tr(\hat{\sigma}^{-1}\sigma) - ln(det(\hat{\sigma}^{-1}\sigma)) - d \tag{3.11}$$

where $\hat{\sigma}$ is the estimated Q and σ is the realized covariance of the true price process, d is the number of assets in our simulation, function tr stands for the trace of a matrix, and ln is the natural logarithm. We further define the mean and standard deviation of the these tow distances:

$$mean_{Frob} = \frac{1}{T}\sum_{j=1}^{T} Frob_j \tag{3.12}$$

$$SD_{Frob} = (\frac{1}{T-1}\sum_{j=1}^{T}(Frob_j - mean_{Frob})^2)^{\frac{1}{2}} \tag{3.13}$$

$$mean_{Stein} = \frac{1}{T}\sum_{j=1}^{T} stein_j \tag{3.14}$$

$$SD_{Stein} = (\frac{1}{T-1}\sum_{j=1}^{T}(stein_j - mean_{Stein})^2)^{\frac{1}{2}} \tag{3.15}$$

To compare the performance of our estimators in a broad range of cases where market microstructure noise varies from moderate to severe and

Table 3.2: Ten-dimensional Q matrices used to generate data in our simulations

$$Q = \begin{pmatrix} 0.1165 & 0.0109 & 0.0100 & 0.0094 & 0.0090 & 0.0078 & 0.0104 & 0.0071 & 0.0069 & 0.0130 \\ 0.0109 & 0.0570 & 0.0086 & 0.0083 & 0.0075 & 0.0071 & 0.0095 & 0.0067 & 0.0062 & 0.0129 \\ 0.0100 & 0.0086 & 0.0814 & 0.0103 & 0.0075 & 0.0072 & 0.0110 & 0.0062 & 0.0097 & 0.0093 \\ 0.0094 & 0.0083 & 0.0103 & 0.0722 & 0.0076 & 0.0066 & 0.0101 & 0.0061 & 0.0076 & 0.0093 \\ 0.0090 & 0.0075 & 0.0075 & 0.0076 & 0.0561 & 0.0118 & 0.0076 & 0.0059 & 0.0071 & 0.0085 \\ 0.0078 & 0.0071 & 0.0072 & 0.0066 & 0.0118 & 0.0398 & 0.0069 & 0.0055 & 0.0065 & 0.0075 \\ 0.0104 & 0.0095 & 0.0110 & 0.0101 & 0.0076 & 0.0069 & 0.0719 & 0.0062 & 0.0081 & 0.0103 \\ 0.0071 & 0.0067 & 0.0062 & 0.0061 & 0.0059 & 0.0055 & 0.0062 & 0.0342 & 0.0046 & 0.0069 \\ 0.0069 & 0.0062 & 0.0097 & 0.0076 & 0.0071 & 0.0065 & 0.0081 & 0.0046 & 0.0681 & 0.0070 \\ 0.0130 & 0.0129 & 0.0093 & 0.0093 & 0.0085 & 0.0075 & 0.0103 & 0.0069 & 0.0070 & 0.0540 \end{pmatrix}$$

Table 3.3: Ten-dimensional diagonal R matrices used to generate data in our simulations

$$R = diag(0.0505, 0.0222, 0.2011, 0.0937, 0.1425, 0.0822, 0.0606, 0.1040, 0.1719, 0.0072)$$

asynchronicity, i.e. missing probabilities, ranges from low to high, we set up six different simulation scenarios(Original missing probabilities $p = [1/2, 1/3, 1/2, 1/4, 1/4, 1/3, 1/5, 1/4, 1/3, 1/4]$, and matrices Q and R are as in Table 2.):

1. standard: missing probabilities p and noise matrix R;
2. high noise: missing probabilities p and noise matrix $R + 0.35$;
3. high missings: missing probabilities $p + 0 : 35$ and noise matrix R;
4. high missings, high noise: missing probabilities $p + 0.35$ and noise matrix $R + 0.35$;
5. dispersed missings: more dispersed missing probabilities $w = [0, 0.5, 0.8, 0.9, 0.25, 0, 0.5, 0.8, 0.9, 0.25]$ and noise matrix R;
6. dispersed missings, high noise: missing probabilities w and noise matrix $R + 0.35$.

We classify these scenarios based on three indicators: average noise-to-signal, average missing probabilities and the standard deviation of the missing observations. The average noise-to-signal is the ratio between the diagonal value of R, the covariance matrix of the microstructure error, and the diagonal value of Q, the covariance matrix of the latent price process, which measures how much influence the noise on the observed prices. It ranges from 0.78 in the Standard scenario to 2.58 in the High-noise scenario. The average missing probability is the amount of time with no price observed over the total amount of time, which indicates the extent of liquidity. It goes from 0.32 in the Standard scenario to 0.67 in the high-missing scenario. The

Table 3.4: Simulation Settings

Setting	Avg.noise-to-signal	Avg.missing prob.	SD missings
Standard	0.78	0.32	0.11
High noise	2.58	0.32	0.11
High missings	0.78	0.67	0.11
High missings, high noise	2.58	0.67	0.11
Dispersed missings	0.78	0.49	0.35
Dispersed missings, high noise	2.58	0.49	0.35

Standard deviation of missing observations is the standard deviation of the average missing probabilities, which tells us the difference of the profiles of assets' liquidity. It starts from 0.11 at Standard scenarios to 0.35 in dispersed missings.

Simulation settings, including variance matrices Q and R used in DGP, and different simulation scenarios, are summarized in the Table 3.2, 3.3 and 3.4.

3.3 Simulation Results and Analysis

First we would like to point out that simulation methods used in the two different estimations are different. In KEM, we iterate between Kalman filter/smoother and maximizing the expected likelihood until the log-likelihood $LogL(Q, R)$ converges. In Bayesian framework, due to the high instability and low convergency speed of Markov Chain Monte Carlo (MCMC) simulation method, we generate 500 matrices of prices and we run the Gibbs sampler for each generated sample of prices for 5000 steps, after 5000 initial iterations of burn-in. Table 3.5 and 3.6 describes the simulation results under different scenarios.

From Table 3.5, we can see clearly, in the standard setting featuring a moderate level of microstructure noise and a high and homogeneous level of liquidity across the assets, the rank of performance among these three estimators: KEM and Bayesian estimator clearly outperform the benchmark HY estimator, which results from the fact that these two methods are able to pull all information from liquid assets to more accurately estimate the covariance matrix; we also discovered that KEM has a slight advantage over Bayesian in terms of the accuracy of estimation, which in our opinion is due to the instability of MCMC method. Another important characteristic that KEM outperforms Bayesian is that KEM is much faster, which we will illustrate later in this section.

In the high noise setting, where the noise-to-signal ratio is raised from the value 0.78 of the standard setting to 2.58 and this level of noise contamination

Table 3.5: Expected values and standard deviations of the Frobenius distances between the estimated and realized covariance matrix in the simulation settings summarized in Table 4

	KEM	Bayesian	HY
1. Standard			
Mean	0.0534	0.0538	0.0711
SD	0.0056	0.0041	0.0057
2. High noise			
Mean	0.0537	0.0542	0.1124
SD	0.0054	0.0028	0.0045
3. High missings			
Mean	0.0547	0.0557	0.0748
SD	0.0024	0.0019	0.0025
4. High missings,high noise			
Mean	0.0621	0.0629	0.1191
SD	0.0056	0.0044	0.0014
5. Dispersed missings			
Mean	0.0658	0.0667	0.0815
SD	0.0023	0.0046	0.0037
6. Dispersed missings,high noise			
Mean	0.0681	0.0688	0.1242
SD	0.0081	0.0019	0.0085

Note: The competing methodologies are the newly introduced KEM approach and Bayesian estimating method, and the pairwise covariance terms estimated by Hayashi and Yoshida (2005). Frobenius distance is defined as $Frob = \sqrt{\sum_{i=1}^{d}\sum_{i=1}^{d}(\hat{\sigma_{ij}} - \sigma_{ij})^2}$.

Table 3.6: Expected values and standard deviations of the Stein distances between the estimated and realized covariance matrix in the simulation settings summarized in Table 5

	KEM	Bayesian	HY
1. Standard			
Mean	0.0484	0.0493	0.0724
SD	0.0041	0.0046	0.0077
2. High noise			
Mean	0.0510	0.0519	0.1154
SD	0.0028	0.0054	0.0054
3. High missings			
Mean	0.0562	0.0573	0.0854
SD	0.0016	0.0015	0.0023
4. High missings,high noise			
Mean	0.0592	0.0609	0.1241
SD	0.0056	0.0049	0.0053
5. Dispersed missings			
Mean	0.0656	0.0674	0.0892
SD	0.0037	0.0066	0.0036
6. Dispersed missings,high noise			
Mean	0.0693	0.0713	0.1462
SD	0.0042	0.0079	0.0085

Note: The competing methodologies are the newly introduced KEM approach and Bayesian estimating method, and the pairwise covariance terms estimated by Hayashi and Yoshida (2005). Stein distance is defined as $stein = tr(\hat{\sigma}^{-1}\sigma) - ln(det(\hat{\sigma}^{-1}\sigma)) - d$.

is quite extreme for standard empirical data, HY performs even worse than it performs in other settings since it is not robust to microstructure noise. On the contrary, KEM and Bayesian estimators is less impacted by the contamination.

In the high missing setting, we raise the average missing probability from 0.32 to 0.67, more than doubling it. Within this setting we can stress test the ability of our proposed approaches to reconstruct the latent efficient price process of the 10 stocks from a much smaller number of observed prices. Still KEM and Bayesian estimator outperform HY, even though HY estimator is designated to deal with non-synchronous trading issues. It seems that our KEM and Bayesian estimator has an advantage in reconstructing the true latent price process from missing observations.

In the next setting where we combine both high noise and high missing, KEM and Bayesian estimator still perform steadily and better than HY, which means that these two are robust to both microstructure noise and asychronicity contemporaneously. The performance of HY is stil severely affected by the existence of noise.

In the Dispersed setting, we construct a scenario where the missing probabilities are more dispersed, i.e. we increase the standard deviation of the distribution of the missing probabilities across the 10 assets. Within this setting we consider two levels of noise-to-signal ratios: moderate and high. We term these two settings dispersed missing and dispersed missing high noise, respectively. KEM an Bayesian estimator are still able to obtain better performance than HY. HY estimator performs fine in the dispersed setting but not in the dispersed and high noise setting due to lack of robustness to microstructure noise.

From table 3.6, where we use the Stein distance as measurement of estimating accuracy between our estimated covariance matrices and the realized covariance, we could obtain similar results as that of the Frobenius distance: KEM and Bayesian estimator both show better property in estimating accuracy than HY estimator, and KEM performs slightly better than Bayesian.

Summarizing, in our extensive monte carlo simulation across different noise and asynchronicity settings, KEM and Bayesian estimators are all superior than HY estimator, i.e., these two approaches could obtain more accurate covariance matrix estimations. This is due to their high robustness to market microstructure noise and asynchronicity, and their ability to successfully obtain information contained in the multivariate variables.

Now let's take a look at the speed our estimators use to perform estimation. From table 3.7 we can conclude that KEM and HY estimator are much more efficient than the Bayesian estimator. The computational cost

Table 3.7: CPU time in second per iteration for fixed number of assets d=10 and different number of sample size T.

	KEM	Bayesian	HY
$24 * 60 * 60 = 86400$	3.1	10.7	2.6
$4 * 60 * 60 = 14400$	1.1	2.9	0.8
$60 * 60 = 3600$	0.6	1.7	0.4

in seconds is a function of the sample size T and the number of assets d. The algorithm CPU time is $O(T\sqrt{\frac{d(d+1)}{2}})$: there is a linear computational cost in the sample size T, and a square root cost in $\sqrt{\frac{d(d+1)}{2}}$, the number of parameters in the covariance matrix.

3.4 Reconstruction of latent price process

An important result of our approach is the signal extraction of the latent efficient price process X for each asset. By the KEM and Bayesian estimation procedure we can filter out the microstructure noise and reconstruct the latent dynamics of the efficient price by exploiting the correlations of one asset with the dynamics of all the other assets. This signal extraction is particularly useful for the less liquid series that have fewer observations because they can benefit more from the information contained in the dynamics of the more liquid assets. Our reconstruction approach is in fact the Kalman filter-smoother algorithm, which are both utilized in KEM and Bayesian approach (FFBS algorithm), thus the degree of accuracy of these two approaches in extracting latent price process is approximately the same.

As an illustration, in Figure 3.2 we use dataset generated in our simulation and plot the reconstructed latent price process together with the observed tick prices. One can clearly see the degree of fitting between the plot (latent price process) and scatter (observed prices) is high.

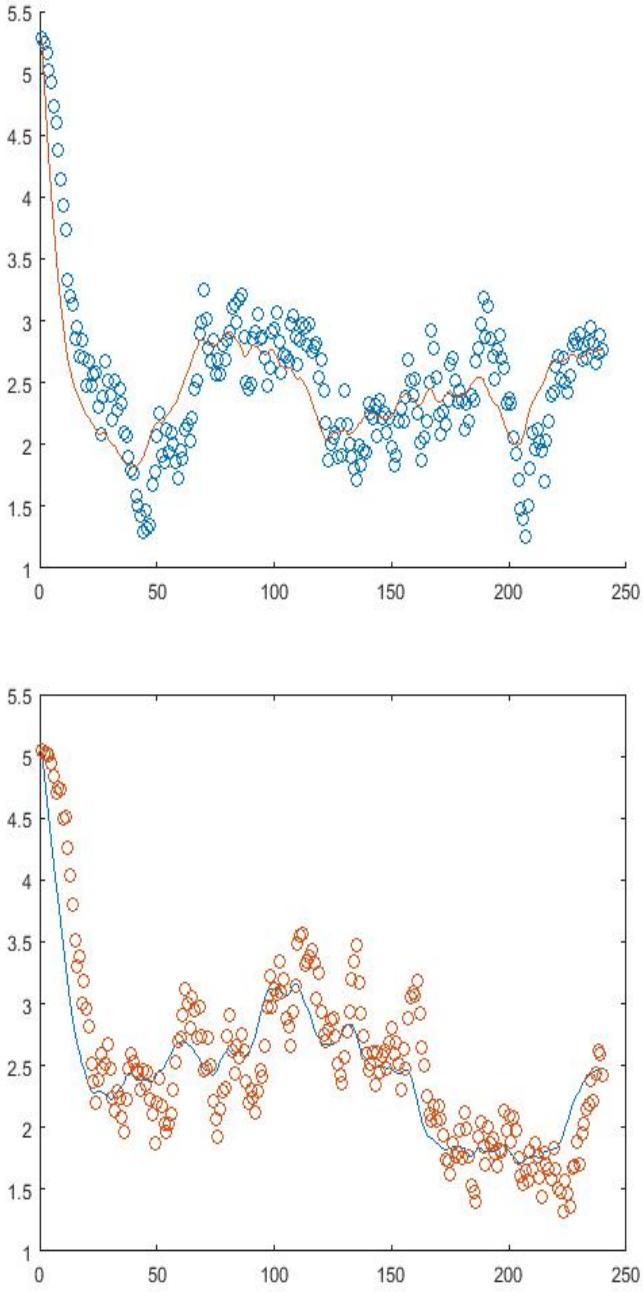

Figure 3.2: Reconstructed latent log-price process with KEM (above) and Bayesian (below) and observed log prices generated in our simulation for T=240

Chapter 4

Empirical Study

4.1 Dataset and Gaussianity Tests

In our empirical study, we use a dataset obtained from Wind database, consisting of tick-by-tick data for 30 Chinese stocks from January 11th 2016 to March 17th 2017. Our stocks consist of both liquid assets and less liquid ones, such as CITIC Securities, China Minsheng Bank, Sinopec, Southern Airlines and China Merchants Bank. In the table 4.1 we report a $10 * 10$ price matrix as an example of our empirical dataset, whose elements are prices corresponds to each column and row, with columns being the sub-sample of 10 stocks in our dataset and rows being the first 10 seconds on January 11th 2016.

First we remove jump observations using a simpler filter that has following steps: assume $Y = (Y_{i,1}, ..., Y_{i,T})$ to be the T observed log prices of asset i in a day

1. Sort the prices and get the ordered sample $(Y_{i,(1)}, ..., Y_{i,T})$;
2. Compute the threshold d as the sample standard deviation of $(Y_{i,\lfloor 0.25T \rfloor}$, ..., $Y_{i,\lfloor 0.75T \rceil})$, where $\lfloor . \rfloor$ is the floor function;
3. If $|Y_{i,t} - Y_{i,t-1}| > 2d$ and $|Y_{i,t} - Y_{i,t+1}| > 2d$, then $Y_{i,t}$ is removed from the series.

In table 4.2 we report the descriptive statistics for 10 stocks in our dataset. To view different missing probabilities in assets with different liquidity, we choose two stocks, China Minsheng Bank and Northern Rare Earth, as representatives of liquid and illiquid stocks and we depict their log-scale time series in Figure 4.1. The average missing probability of them ranges from 0.5 to 0.8, with mean and standard deviation being 0.69 and 0.13. Thus high missing is the most close simulation scenario to our empirical study.

Table 4.1: An example of our empirical data: trading information in the first 10 seconds on January 11th 2016

	600016	600028	600029	600030	600036	600050	600100	600104	600109	600111
1	0	0	11.36	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	11.40	0
4	0	0	0	0	0	0	0	0	0	11.98
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	10.84	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	12.16	0	0
9	0	0	0	0	0	0	11.75	0	0	0
10	0	0	0	0	0	0	11.76	0	0	0

The first row of this table is the code of each 10 stocks on Chinese stock market. The first column represents the beginning 10 seconds on January 11th 2016. One can see from this table the existence and randomness of missing observations.

Table 4.2: Descriptive statistics of 10 Chinese stocks' tick-by-tick log prices for the period January 11th 2016 to March 17th 2016, downloaded from Wind

Name	Stock Code	Mean	SD	nobs($*10^4$)	MissProb
China Minsheng Bank	600016	10.82	0.0065	1.2290	0.5242
Sinopec	600028	11.46	0.0034	1.3532	0.5863
Southern Airlines	600029	12.24	0.0086	1.3328	0.5735
CITIC Securities	600030	12.06	0.0047	1.0462	0.6453
China Merchants Bank	600036	11.18	0.0053	1.2743	0.4734
China Unicom	600050	12.09	0.0064	0.9174	0.6832
Tongfang Co.	600100	12.29	0.0038	1.3472	0.7932
SAIC Motor	600104	11.99	0.0061	1.2643	0.7532
Sinolink Securities	600109	11.85	0.0073	1.1363	0.8363
Northern Rare Earth	600111	14.59	0.0079	1.1524	0.7932

Note: The columns reports for each stock: stock name, stock number, average mean log price per day, average standard deviation of log price every day, average number of observations per day and average percentage of missing observations per day

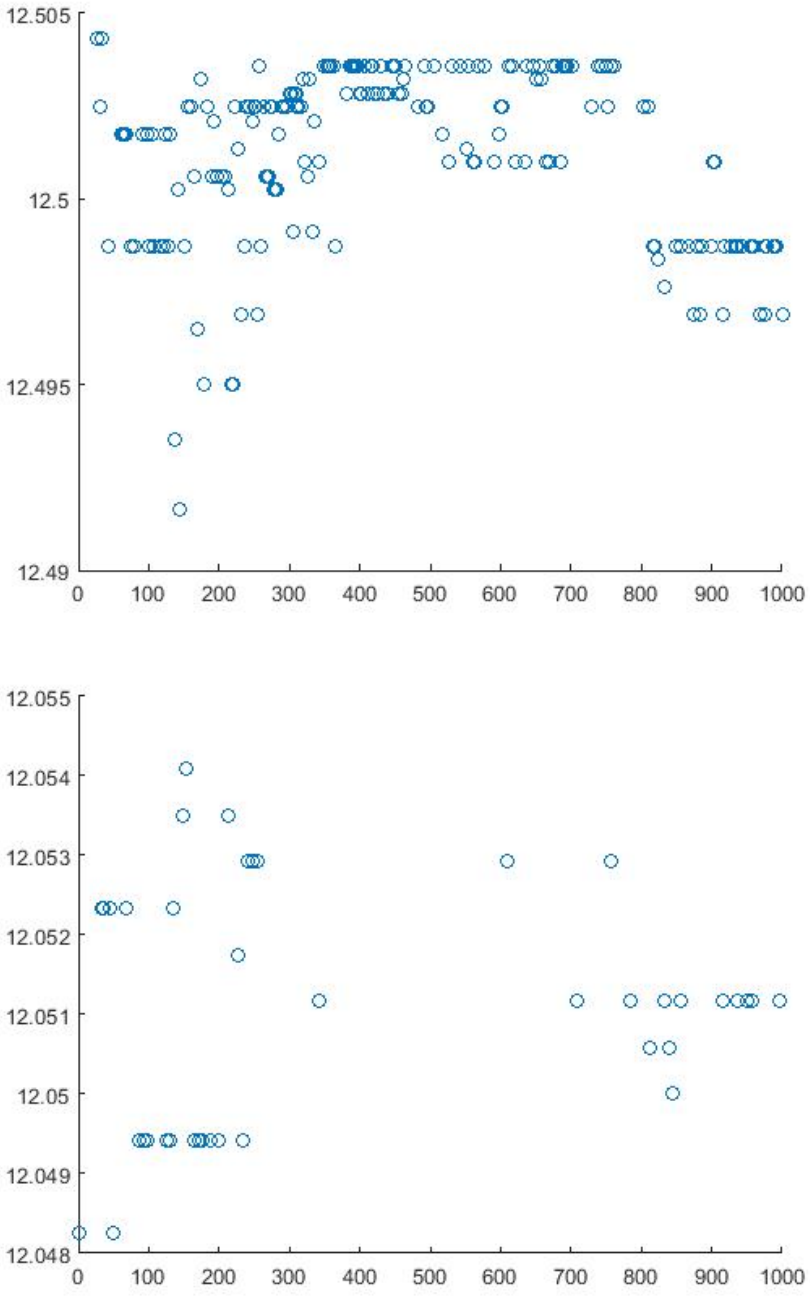

Figure 4.1: Above:China Minsheng Bank, First 1000 seconds of January 11th 2016 log prices; Below: Northern Rare Earth Company, First 1000 seconds of January 11th 2016 log prices

Table 4.3: Empirical study results: moment test and χ^2 test

Method	removal of jump	Mean dist	Cov dist	p-val of χ^2 test
KEM	after	0.1174	0.9763	0.7041
	before	0.2654	1.3534	0.6332
Bayesian	after	0.3536	3.7463	0.5325
	before	0.5832	4.5262	0.4011
HY	after	0.6732	9.6453	0.3824
	before	1.7453	10.4333	0.3117

Note: The competing methodologies are the newly introduced KEM approach and Bayesian estimating method, and the pairwise covariance terms estimated by Hayashi and Yoshida (2005). HY is projected to impose positiveness of the estimated covariance matrices using method introduced earlier. In this table, the first column reports the competing methodologies. The second and third column reports the test based on moments comparison. And the last column reports the p-value for the χ^2 test.

In order to show the performance of estimators using empirical dataset, we perform two tests: Moment test and χ^2 test. Andersen et al. (2007)[29] argue that if the daily returns are standardized by the mean and corresponding intraday realized volatility, then the resulting standardized distribution is indeed Gaussian. In a multivariate case, it is analogous that the daily return vector could be rescaled to multivariate standard Gaussian distribution by the daily realized covariance matrix. Hence, following this theory, we propose two tests: first we compute the standardized open-to-close daily return vectors using mean vectors and estimated covariance matrices. Then,

1. In the moment test, we compare the sample and hypothesized moments of the standardized returns. In particular, we compute the mean of the standardized returns over the days and the Euclidean distance between it and the null vector. In the same way, we compute the covariance matrix of the standardized returns and the Frobenius distance between this matrix and the identity matrix. Theoretically, more accurate estimation method will produce lower distances.
2. In the χ^2 test, we compute the probability of observing extreme data, given the null hypothesis of multivariate standard Gaussian distribution of standardized returns. Under the null, the vector of daily returns is an observation of the multivariate standard Gaussian distribution, or, equivalently, its squared Euclidean norm is an observation from a χ^2 distribution with d degrees of freedom, denoted χ^2_d. Then the area under the χ^2_d density on the right of the observed squared Euclidean norm is the p-value, and we expect the best methodology to show a

higher p-value, suggesting more confidence in the null hypothesis.

It is clearly showed in the table 4.3 that in empirical study, KEM is still the best performing estimator among three while the Bayesian estimator outperforms the HY estimator. Early in this section, we mentioned that our empirical dataset fits the high missing scenario in our simulation. Thus, it is understandable that our KEM and Bayesian estimator, which are robust to market microstructure noise, show superior performance over the HY estimator. In the meantime the result also showed that these three estimators perform uniformly worse with the presence of jumps. It is consistent with the theory that jumps in time series data could induce deviations from standardized Gaussian returns.

4.2 Portfolio Allocation Study

Here in this section, we perform a comparison study on the accuracy of estimated covariance matrices when they are applied to a portfolio allocation problem, using the methodology proposed by Engle and Colacito (2006)[23]. In our context we consider an investor who solves the conventional mean-variance portfolio allocation problem to allocate funds to different stocks in our dataset, i.e., his or her goal is to minimize the expected variance of the portfolio with some optimal weights of stocks. This optimization problem can be summarized as:

$$\begin{aligned} min_{\omega_t}\ \sigma_t^2 = min_{\omega_t}\ \omega_t' Q_t \omega_t, \\ subject\ to\ \omega_t' i = 1 \end{aligned} \tag{4.1}$$

where i denotes a vector of ones. Here we need the value of covariance matrix Q_t to solve this problem; to fulfill our goal to compare the performance of different covariance estimators $\hat{Q}_{t,j}$, we use these estimations to solve for each optimization problem:

$$\begin{aligned} min_{\omega_t}\ \sigma_t^2 = min_{\omega_t}\ \omega_t' \hat{Q}_{t,j} \omega_t, \\ subject\ to\ \omega_t' i = 1 \end{aligned} \tag{4.2}$$

where j denotes the jth method for estimating covariance.

To make pairwise comparison of the performance of three competing estimators, we can test differences in the following way: for each time t and estimating approach j, the above optimization problem is solved and the weights of each stock across the portfolio is determined. Then we perform a widely used model accuracy test proposed by Diebold and Mariano (2006)[7] on the differences of variances on two portfolios $u_t = \sigma_{t,j_1}^2 - \sigma_{t,j_2}^2$, the null hypothesis of which is that the mean of the differences

Table 4.4: Pairwise Diebold-Mariano test on the mean of portfolio differences

j_1	j_2	Mean
KEM	HY	-0.0405
Bayesian	HY	-0.0254
KEM	Baysian	-0.0075

Note: Results of pairwise tests on portfolio variances constructed from 30 Chinese stocks for the null hypothesis of equal predictive ability between: the competing methodologies are the newly introduced KEM approach and Bayesian estimating method, and the pairwise covariance terms estimated by Hayashi and Yoshida (2005). HY is projected to impose positiveness of the estimated covariance matrices using method introduced earlier. Negative values are in favor of the approaches in the first column.

u_t is zero. In results, positive mean shows that method j_2 is more accurate. In the table 6, we report that mean of portfolio differences for each pair of competing approaches using dataset from Gaussianity tests consisting of 30 Chinese stocks from January 11th 2016 to March 17th 2017.

Results in table 4.4 show consistency with our simulation study and Gaussianity tests: KEM estimator is the most accurate in terms of predicting covariance in all three estimators, while Bayesian estimator is more accurate than HY estimator. Here HY is projected to impose positiveness of the estimated covariance matrices using the Eigenvalue clipping projection method mentioned in the simulation study section.

We could also evaluate the economic benefit of the proposed covariance matrices, using the methodology proposed by West, Edison and Cho (1993)[10] and Fleming, Kirby and Ostdiek (2003)[8], which studies the different utility levels obtained by different covariance estimators when we applied them to portfolio allocation problem. Here we consider the same question as before, i.e., a risk-averse investor who wants to use a conditional mean-variance optimization rule to allocate funds across the stocks in our datatset. The following is the mathematical model of this problem:

Let R_t^f be the risk-free rate, $\mu_t^i = E_t[R_{t+1}^i]$ the expected value at time t of the stock i return at time t+1, $\mu_t^p = E_t[R_{t+1}^p]$ the expected return of the portfolio. The investor is interested in solving the optimization problem:

$$min_{\omega_t} \omega_t' Q_t \omega_t$$

subject to

$$\omega_t' \mu_t + (1 - \omega_t' \mathbf{1}_d) R_t^f = \mu_p$$

Table 4.5: Annualized fees (expressed in basis points) that an investor following a volatility timing strategy would be willing to pay to employ the KEM estimated covariance in place of the alternative estimators. The portfolio weights are obtained minimizing the conditional variance of a portfolio containing the 30 stocks in our dataset and three-month China Treasury bonds

Expected return	Method	$\lambda = 2$	$\lambda = 7$	$\lambda = 10$
0.06	Bayesian	0.1	1.4	3.7
	HY	0.3	1.7	3.5
0.10	Bayesian	1.5	3.6	7.5
	HY	2.3	9.3	11.5
0.15	Bayesian	2.6	4.4	8.4
	HY	3.6	7.3	17.9

where ω_t is the vector of portfolio weights, Q_t is the covariance matrix estimated at time t with tick prices observed in day t-1, and $\mathbf{1}_d$ is a d-dimensional vector of ones. To solve this optimization problem, we assume there is no predicability in the expected returns of the single stocks at the daily level and we follow a volatility timing strategy where the weights vary only with Q_t. Thus, we have $\mu_t^i = \mu_t = R_t^f + 0.1$, i.e., we fix the expected stock returns equal to the risk-free rate plus a constant spread in a mean-variance framework. The optimal weights that minimize the portfolio variance is:

$$\omega_t = \frac{(\mu_p - R_t^f) Q_t^{-1} (\mu_t - R_t^f) \mathbf{1}_d}{(\mu_t - R_t^f) \mathbf{1}_d Q_t^{-1} \mathbf{1}_d' (\mu_t - R_t^f)'}$$

Here we introduce a measure which computes the economic differences between the covariance estimators, proposed by Engle and Colacito (2006); Audrino and Trojani (2006); Corsi and Andrino (2012). This measure is interpreted as the fee that the investor would be willing to pay in order to switch from one covariance to the other. It is called AU measure, defined as:

$$AU = \frac{\lambda}{2} \frac{1}{T} \sum_{t=1}^{T} (R_t^p - \bar{R}^p)^2$$

where

$$R_t^p = R_t^f + \omega_{t-1}' (R_t - R_t^f \mathbf{1}_d)$$

is the return of the portfolio constructed at time t-1, $\bar{R}^p$ is the sample mean of portfolio returns, and λ is a coefficient of risk aversion.

Table 4.6: Annualized fees (expressed in basis points) that an investor following a volatility timing strategy would be willing to pay to employ the Bayesian estimated covariance in place of the alternative estimators. The portfolio weights are obtained minimizing the conditional variance of a portfolio containing the 30 stocks in our dataset and three-month China Treasury bonds

Expected return	Method	$\lambda = 2$	$\lambda = 7$	$\lambda = 10$
0.06	KEM	0.0	-0.1	-0.8
	HY	0.1	0.9	1.5
0.10	KEM	-0.5	-1.5	-2.9
	HY	1.4	2.3	5.5
0.15	KEM	-1.6	-2.4	-5.8
	HY	2.6	6.3	10.9

Our table 4.5 and 4.6 reports the values of AU measure for different estimating methodologies under different portfolio expected returns:0.06, 0.10 and 0.15, and risk aversion coefficients: λ equals to 2,7 and 10, using the empirical dataset of 30 stocks in Chinese stock market. The table shows a trend in investor's investment decisions when he applies the volatility timing strategy: it is conclusive that investors are willing to pay a positive fee to switch to KEM or Bayesian estimator from HY esimator, which is consistent with our previous simulation and empirical results that KEM and Bayesian estimators always perform better than the benchmark HY estimator. Furthermore, we can also discover that investors are willing to switch to KEM estimator from Bayesian estimator, even if it comes with a positive annual fee; on the opposite direction, investors are more reluctant to switch to Bayesian estimator from KEM, which is clearly showed by the negative value of AU measure. There is another trend worth noting: the fee investors are willing to pay is increasing as the expected portfolio returns μ_p and risk aversion coefficient λ are increasing, which is a sign of growing economic benefits.

Chapter 5

Conclusion

Our research adopts two methodologies, Kalman-EM (KEM) and Bayesian estimation, for estimating multivariate covariance matrices, and compares their performance in predicting/forecasting with a benchmark estimator, Hayashi and Yoshida (HY). Unlike the benchmark one, these two estimators are robust to both market microstructure noise and asynchronicity trading effect, and could successfully extract all trading information contained in price process when estimating covariance matrices. The success results from the assumption that we could treat market microstructure as a measurement error and asychronicity as missing observations in otherwise synchronous time series, and their adoption of a state space model which provides an appropriate setting to account for these two model errors. In terms of specific methods, KEM algorithm iterates between two steps: 1) E step: reconstruct the smoothed and synchronized series of the latent price processes; 2) M step: use the fictitious complete datatset to maximize the likelihood, obtaining estimates for covariance matrices. Bayesian estimation method cast this problem in the usual Bayesian framework, viewing latent price processes and covariance matrices as parameters in conditional distributions, which leads to the application of Gibbs sampler: we iterate alternatively between four sampling steps for many times until this Markov Chain is stable, then the samples of the covariance matrix is our estimation. Another important matter is that these two estimators are positive semi-definite (psd) by construction, which could be essential in practice.

To show the superiority of these two estimators over HY, we perform Monte Carlo simulation analysis in a broad range of simulation settings including different levels of market microstructure noise and missing observation distributions. We also perform an empirical study analysis, including Gaussianity tests and portfolio allocation analysis, using dataset consisting of 30 Chinese stocks from January 11th 2016 to March 17th 2017. These studies, both simulation and empirical, show consistently that KEM and

Bayesian estimator outperform HY, and KEM also has its advantage over Bayesian in both estimating accuracy and computing speed. Thus, the KEM estimator is highly suited for possibly high-dimensional portfolio applications, while Bayesian estimator is a good method when applying to smaller sample. As we have mentioned in our introduction, these psd estimators have important applications in portfolio selection and risk management.

Bibliography

1. Dempster A, Laird N, and Rudin D. Maximum likelihood estimation from incomplete data. *Journal of the Royal Statistical Society*, 39:1–38, 1977.
2. Palandri A. Consistent realized covariance for aynchronous observations contaminated by the market microstructure noise. *Working paper. University of Copenhagen*.
3. Oksendal B. *Stochastic differential equations: an introduction with applications*. New York: Springer, 6th edition, 2003.
4. Xiu D. Quasi-maximum likelihood estimation of volatility with high frequency data. *Journal of Econometrics*, 159:235–250, 2010.
5. Lehmann E.L. and Casella G. *Theory of Point Estimation*. New York: Springer-Verlag, 2th edition, 1998.
6. Corsi F, Peluso S, and Audrino F. missing in asynchronicity: a kalman-em approach for multivariate realized covariance estimation. *Journal of Applied Econometrics*, 30:377–397, 2015.
7. Diebold F and Mariano R. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13:253–263, 1995.
8. Fleming J, Kirby C, and Ostdeik B. The economic value of volatility timing using realized volatility. *Journal of Financial Econometrics*, 67:473–509, 2003.
9. Khaled K and Samia M. Estimation of the parameters of the stochastic differential equations black-scholes model share price of gold. *Journal of Mathematics and Statistics*, 6(4):421–424, 2010.
10. West K, Edison H, and Cho D. he shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well. *Management Science*, 55:1914–1932, 2009.
11. Zhang L. Estimaing covariation: Epps effect, microstructure noise. *Journal of Econometrics*, 160(1):33–47, 2011.

12. Mancino M and Sanfelici S. Estimating covariance via fourier method in the presence of asynchronous trading and microstructure noise. *Journal of Financial Econometrics*, 9(2):367–408, 2011.

13. West M and Harrison P. *Bayesian Forecasting and Dynamic Models*. Springer: New York, 1997.

14. Eaton M.L. *Multivariate Statistics: a vector space approach*. Institute of Mathematical Statisitcs, 2007.

15. Gupta N and Mehra R. Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *IEEE Transaction on Automatic Control*, AC-19:774–783, 1974.

16. Hautsch N, Kyj L, and Oomen R. A blocking and regularization approach to high-dimensiional realized covariance estimation. *Journal of Applied Econometrics*, 27:625–645, 2012.

17. Barndorff-Nielsen OE, Hansen PR, Lunde A, and Shephard N. Multivariate realized kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. *Journal of Econometrics*, 162(2):149–169, 2011.

18. Christoffersen P, Heston S, and Jacobs K. Autility-based comparison of some models od exchange rate volatility. *Journal of International Economics*, 35:23–45, 1993.

19. Hansen P and Lunde A. Realized covariance and market microstructure noise. *Journal of Business and Economic Statistics*, 24(2):127–161, 2006.

20. Hogg R, McKean J, and Craig A. *Introduction to Mathematical Statistics*. Pearson, 7th edition, 2013.

21. Shumway R and Stoffer D. An approach to time series smoothing and forecasting using the em algorithm. *Journal of Time Series Analysis*, 3:253–264, 1982.

22. Merton RC. On estimating the expected return on the market: an exploratory investigation. *Journal of Financial Econometrics*, 8:323–361, 1980.

23. Engel RF and Colacito R. Testing and valuing dynamic correlations for asset allocation. *Journal of Business and Economic Statistics*, 24(2):238–253, 2006.

24. Heston S. A closed-form solution for options with stochastic volatility with applicaitons to bond and currency options. *Review of Financial Studies*, 6:327–343, 1993.

25. Laurent S, Rombouts J, and Violante F. On loss funcitons and ranking forecasting performances of multivariate volatility models. *Journal of Econometrics*, 173(1):1–10, 2013.

26. Peluso S, Corsi F, and Mira A. A bayesian high-frequency estimator of the multivariate covariance of noisy and asynchronous returns. *Journal of Financial Econometrics*, 13(3):665–697, 2015.

27. Epps T. Comovements in stock prices in the very short run. *Journal of American Statistical Association*, 74:291–296, 1979.

28. Hayashi T and Yoshida N. On covariance estimation of non-synchronous observed diffusion processes. *Bernoulli*, 11:359–379, 2005.

29. Andersen T.G., Bollerslev T, and Dobrev D. No-arbitrage semi-martingale restrictions for continuous-timevolatility models subject to leverage effects, jumps and i.i.d.noise: Theory andtestable distributionalimplications. *Journal of Financial Econometrics*, 138:125–180, 2007.

Appendix A: Kalman Filter and Smoothing Recursions

The Kalman filter recursion formulas are, for $t = 1, ..., T$:

$$\begin{aligned} X_t^p &= X_{t-1}^f \\ V_t^p &= V_{t-1}^f + Q \\ K_t &= V_t^p Z_t (Z_t V_t^p Z_t + R)^{-1} \\ X_t^f &= X_t^p + K_t (Y_t - Z_t X_t^p) \\ V_t^f &= V_t^p - K_t Z_t V_t^p \end{aligned} \tag{5.1}$$

where X_t^p is the predicted value of the log-price latent process, V_t^p is the variance of the prediction error, K_t is the filtering correction term, X_t^f is the filtered value of the log-price latent process, V_t^f is the variance of the filtering error and Z_t is a diagonal matrix with ith diagonal element equal to 1 if $Y_{i,t}$ is observed or equal to 0 otherwise, $i == 1, ..., d$.

The smoothing recursions are, for $t = T - 1, ..., 1$:

$$\begin{aligned} J_t &= V_t^f (V_{t+1}^p)^{-1} \\ X_t^s &= X_t^f + J_t (X_{t+1}^s - X_t^f) \\ V_t^s &= V_t^f + J_t (V_{t+1}^s - V_{t+1}^p) J_t' \\ VL_{t+1}^s &= V_{t+1}^f J_t' + J_{t+1} (VL_{t+2}^s - V_{t+1}^f) J_t', \quad t < T - 1 \\ VL_T^s &= (I - K_n) V_{n-1}^f \end{aligned} \tag{5.2}$$

where J_t is the smoothing correction term, X_t^s is the smoothed log-price latent process, V_t^s is the variance of the smoothing error and VL_t^s is the one-lag autocovariance of the smoothing error.

Appendix B: Proof of Euler-Maruyama Formula

For a general form of a stochastic differential equation

$$\begin{aligned} dX(t) = f(X(t))dt + g(X(t))dW(t), \\ X(0) = X_0, \\ 0 \leq t \leq T \end{aligned} \tag{5.3}$$

where W denote a Brownian motion, f and g are given functions. A solution is a stochastic process, which can be interpreted as integral equation:

$$\begin{aligned} X(t) = X_0 + \int_0^t f(X(s))ds + \int_0^t g(X(s))dW(s), \\ 0 \leq t \leq T \end{aligned} \tag{5.4}$$

Let $\Delta t = T/L$ (for some positive integer L) and $\tau_j = j\Delta t$ $(j = 0, 1, ..., L)$. Our numerical approximation to $X(\tau_j)$ will be denoted X_j. The Euler-Maruyama method takes the form as equation (19):

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W(\tau_j) - W(\tau_{j-1})), \quad j = 1, 2, ..., L \tag{5.5}$$

Now we show how to obtain equation (5.5): setting $t = \tau_j$ and $t = \tau_{j-1}$ in (5.4), we get:

$$X(\tau_j) = X_0 + \int_0^(\tau_j) f(X(s))ds + \int_0^{\tau_j} g(X(s))dW(s) \tag{5.6}$$

$$X(\tau_{j-1}) = X_0 + \int_0^(\tau_{j-1}) f(X(s))ds + \int_0^{\tau_{j-1}} g(X(s))dW(s) \tag{5.7}$$

We subtract equation (5.6) and (5.7):

$$X(\tau_j) = X(\tau_{j-1}) + \int_{\tau_{j-1}}^(\tau_j) f(X(s))ds + \int_{\tau_{j-1}}^{\tau_j} g(X(s))dW(s) \tag{5.8}$$

For the first integral in (5.8) we can use the conventional deterministic approximation and for the second we use Ito formula:

$$\int_{\tau_{j-1}}^(\tau_j) f(X(s))ds \approx (\tau_{j-1} - \tau_j) f(X_{j-1}) = \Delta t f(X_{j-1}) \tag{5.9}$$

$$\int_{\tau_{j-1}}^{\tau_j} g(X(s))dW(s) \approx g(X_{j-1})(W(\tau_j) - W(\tau_{j-1})) \tag{5.10}$$

Replacing (5.9) and (5.10) in (5.8) completes our proof for the Euler-Maruyama formula.

Appendix C: Matlab Code of KEM Algorithm

```
function [x_filt,x_smooth,Q,R,Q_old,c_conv,l,i,q,r]
=KEM(y,Q_init,R_init,maxiter,eps,showplot)

% KEM algorithm
% INPUTS:  y         = dxT data matrix with NaN of d subjects for T
%                      observations
%          Q_init    = initial value for innovation error variance
%          R_init    = initial value for observation error variance
%          maxiter   = max n of iterations allowed before convergence
%          eps       = min tolerance change in log-likelihood for
%                      convergence
%          showplot  = plots are (not) shown if (0) 1
% OUTPUTS: x_filt    = filtered data at last iteration
%          x_smooth  = smoothed data at last iteration
%          Q         = estimated innovation error variance
%          R         = estimated observation error variance
%          Q_old     = estimated innovation error variance before the last
%                      iteration
%          c_conv    = equal to 1 if convergence has been reached
%          l         = log-likelihood path
%          i         = num of steps done until convergence (or maxiter)
%          q         = vectorized innovation error variance path
%          r         = vectorized observation error variance path

% preliminaries
[d,T]=size(y);
a=isnan(y)==0;
c_conv=0;

% memory allocation
l=NaN*ones(1,maxiter);
deltalog=NaN*ones(1,maxiter);
q=NaN*ones(d*(d+1)/2,maxiter);
r=NaN*ones(d*(d+1)/2,maxiter);

% initialization
l_old=-10e10;
Q=Q_init; R=R_init; I=eye(d); Phi=I;
```

```
for i=1:maxiter
    tic
    disp(['iteration ', int2str(i)])

    % Kalman filtering and smoothing
    y(a==0)=NaN;
    clear x_filt x_smooth V_smooth Vt_smooth loglik
    [x_filt,x_smooth,V_smooth,Vt_smooth,loglik]=Ksmoother_miss_fast(y,Q,R
    y(a==0)=0;

    % compute incomplete data log-likelihood
    l(i)=loglik;

    % E step
    S=zeros(d); S10=zeros(d); eps_smooth=zeros(d);
    for t=1:T
        D=diag(a(:,t));
        S=S+x_smooth(:,t)*x_smooth(:,t)'+V_smooth(:,:,t);
        if t>1,S10=S10+x_smooth(:,t)*x_smooth(:,t-1)'+Vt_smooth(:,:,t);en
        eps_smooth=eps_smooth+D*(y(:,t)-x_smooth(:,t))*(y(:,t)...
        -x_smooth(:,t))'*D'+D*V_smooth(:,:,t)*D'+(I-D)*R*(I-D)';
    end
    S00=S-x_smooth(:,T)*x_smooth(:,T)'-V_smooth(:,:,T);
    S11=S-x_smooth(:,1)*x_smooth(:,1)'-V_smooth(:,:,1);

    %M step
    Q_old=Q; R_old=R; Phi_old=Phi; %#ok<NASGU>
    Q=(S11-S10/S00*S10')/(T-1);
    Q=tril(Q)+tril(Q,-1)';
    q(:,i)=vech2(Q);
    R=eps_smooth/T;
    R=tril(R)+tril(R,-1)';
    r(:,i)=vech2(R);
    Phi=S10/S00;

    % break for convergence
    deltalog(i)=abs(l(i)-l_old)/abs(l_old);
    if deltalog(i)<eps,
        str=int2str(i);
        disp(['break for convergence at iteration ', str])
        c_conv=1;
```

```
        break
    end
    l_old=l(i);
    toc
end

function [x_filt,x_smooth,V_smooth,Vt_smooth]=Ksmoother_miss_fast(y,Q,R)

% Kalman filter and smoother for the dynamic linear model
% y_t = x_t     + eta_t, eta_t~N(0,R)
% x_t = x_{t-1} + eps_t, eps_t~N(0,Q)

% INPUTS:  y         = dxT matrix of T observations for d subjects
%          Q         = dxd observational error matrix
%          R         = dxd innovation error matrix
% OUTPUTS: x_pred    = predicted value of the latent process
%          x_smooth  = smoothed value of the latent process
%          V_pred    = variance of the prediction error
%          V_smooth  = variance of the smoothing error
%          Vt_smooth = one-lag autocovariance of the smoothing error
%          D         = instrumental matrix expressing missings
%          V_filt    = variance of the filtering error

% preliminaries and memory allocation
[d,T]=size(y);
a=isnan(y)==0;
I=eye(d);
x_pred=zeros(d,T);x_filt=zeros(d,T);x_smooth=zeros(d,T);
V_filt=zeros(d,d,T);V_pred=zeros(d,d,T);V_smooth=zeros(d,d,T);
Vt_smooth=zeros(d,d,T);J=zeros(d,d,T);
loglik=0;

% filtering initialization
y(a==0)=0;
for j=1:d,x_filt(j,1)=y(j,find(y(j,:),1,'first'));end
V_filt(:,:,1)=zeros(d);
x_pred(:,1)=x_filt(:,1);
V_pred(:,:,1)=V_filt(:,:,1);

for t=2:T,
```

```
    % "Obervations" matrix
    D=diag(a(:,t));

    % Kalman filter
    x_pred(:,t)=x_filt(:,t-1);
    V_pred(:,:,t)=V_filt(:,:,t-1)+Q;
    K=V_pred(:,:,t)*D'/(D*V_pred(:,:,t)*D'+R);
    x_filt(:,t)=x_pred(:,t)+K*(y(:,t)-D*x_pred(:,t));
    V_filt(:,:,t)=V_pred(:,:,t)-K*D*V_pred(:,:,t);
    V_temp=V_pred(a(:,t),a(:,t));
    R_temp=R(a(:,t),a(:,t));
    if ne(sum(a(:,t)),0)
        if ne(det(V_temp+R_temp),0)
        loglik=loglik+(-1/2*log(det(V_temp+R_temp))-1/2*((y(a(:,t),t)...
        -x_filt(a(:,t),t-1))'/(V_temp+R_temp)*(y(a(:,t),t)...
        -x_filt(a(:,t),t-1))));
        end
    end
end

% smoothing initialization
x_smooth(:,T)=x_filt(:,T);
V_smooth(:,:,T)=V_filt(:,:,T);
Vt_smooth(:,:,T)=(I-K*D)*V_filt(:,:,T-1);
J(:,:,T-1)=V_filt(:,:,T-1)/V_pred(:,:,T);
x_smooth(:,T-1)=x_filt(:,T-1)+J(:,:,T-1)*(x_smooth(:,T)-x_filt(:,T-1));
V_smooth(:,:,T-1)=V_filt(:,:,T-1)+J(:,:,T-1)*(V_smooth(:,:,T)...
                    -V_pred(:,:,T))*J(:,:,T-1)';

% Rauch recursions
for t=T-1:-1:2,
    J(:,:,t-1)=V_filt(:,:,t-1)/V_pred(:,:,t);
    x_smooth(:,t-1)=x_filt(:,t-1)+J(:,:,t-1)*(x_smooth(:,t)-x_filt(:,t-1
    V_smooth(:,:,t-1)=V_filt(:,:,t-1)+J(:,:,t-1)...
                        *(V_smooth(:,:,t)-V_pred(:,:,t))*J(:,:,t-1)';
    Vt_smooth(:,:,t)=V_filt(:,:,t)*J(:,:,t-1)'+J(:,:,t)*(Vt_smooth(:,:,t+
    -V_filt(:,:,t))*J(:,:,t-1)';
end
```

Appendix D: Matlab Code of Gibbs Sampler

```
function [y1,P_y]=f_gibbs(P,nsim)
% preliminaries and memory allocation
[T,d]      = size(P);                    % dimensions of price matrix
y1         = NaN*ones(d*(d+1)/2,nsim);   % sampled Var elements
a          = isnan(P)==0;                % data indicator
tol        = 1e-04;

% missings initialization
P_y=P; n_miss=sum(a==0);
if sum(n_miss)~=0
    lag = 10*T./(T-n_miss); lag=round(lag);
    for j = 1:d
        P_y(1,j)=P(find(a(:,j)==1,1,'first'),j);
        P_y(1:lag(j),j)=nanmean(P_y(1:lag(j),j));
        for i = (1+lag(j)):T,P_y(i,j) =...
            nanmean(P_y((i-lag(j)):min(i+lag(j),T),j)); end
    end
end
P_y(a==1)=P(a==1);

% covariances initialization
% Calculate Realized Covariance Matrix RCQ %
RCQ=zeros(d);
for i=2:T
    m=P(:,i)-P(:,i-1);
    RCQ=RCQ+m*m';
end
sigma_new = RCQ;
omega_new = diag(max(diag(diff(P_y)'*diff(P_y)-sigma_new),tol));

% vectorization of covariance parameters
y1(:,1)=vech2(sigma_new);

% projection to positive matrix with min Frobenius
sigma_new=frobproj(sigma_new,0.001);

%%%%%%%%% GIBBS SAMPLER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for t=2:nsim,
    disp(t)
```

```
    tic

    % sampling latent process
    x_sim     = FFBS_fast(P_y',sigma_new/T,omega_new/T)';
    SS_sigma  = diff(x_sim)'*diff(x_sim);
    SS_omega  = (P_y-x_sim)'*(P_y-x_sim);
    SS_sigma  = tril(SS_sigma)+tril(SS_sigma,-1)';
    SS_omega  = tril(SS_omega)+tril(SS_omega,-1)';

    % sampling sigma
    sigma_new = iwishrnd(SS_sigma,T)*T;
    y1(:,t)   = vech2(sigma_new);

    % sampling omega
    omega_new = iwishrnd(SS_omega,T)*T;
    y3        = vech2(omega_new); disp(y3')

    % sampling missing observations
    if sum(n_miss)~=0,
        s=y3(1:d)/T;
        for i=1:d,
            ind = setdiff(1:d,i); a_i = a(:,i)==0;
            Omega22=omega_new(ind,ind)/T;
            Omega12=omega_new(i,ind)/T;
            P_y(a_i,i)=normrnd(x_sim(a_i,i)+
            (Omega12/Omega22*(P_y(a_i,ind)-...
             x_sim(a_i,ind))')',sqrt(s(i)-Omega12/Omega22*Omega12'));
        end
        P_y=real(P_y);
    end

    % plots
    for j=1:3,subplot(3,1,j),plot(y1(j,1:t));end;
    pause(0.0001)

    toc
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x_sim,x_filt,loglik]=FFBS_fast(y,Q,R)
% preliminaries and memory allocation
```

```
[d,T]   = size(y);
loglik  = 0;
tol     = 1.0e-18;
x_filt  = zeros(d,T);
x_sim   = zeros(d,T);
V_filt  = zeros(d,d,T);

% filtering initialization
V0              = eye(d)*0.001;
K               = V0/(V0+R);
V_filt(:,:,1)   = V0-K*V0;
x_filt(:,1)     = y(:,1);

% Kalman filter
for t=2:T,
    V_pred          = V_filt(:,:,t-1)+Q;
    K               = V_pred/(V_pred+R);
    x_filt(:,t)     = x_filt(:,t-1)+K*(y(:,t)-x_filt(:,t-1));
    V_filt(:,:,t)   = V_pred-K*V_pred;
end

% simulation initialization
V_filt(:,:,T)  = tril(V_filt(:,:,T))+tril(V_filt(:,:,T),-1)';
V_filt(:,:,T)  = frobproj(V_filt(:,:,T),tol);
x_sim(:,T)     = mvnrnd(x_filt(:,T),V_filt(:,:,T));

% backward simulation
for t=T-1:-1:1,
    h           = x_filt(:,t)+V_filt(:,:,t)/(V_filt(:,:,t)...
                  +Q)*(x_sim(:,t+1)-x_filt(:,t));
    H           = V_filt(:,:,t)-V_filt(:,:,t)/(V_filt(:,:,t)...
                  +Q)*V_filt(:,:,t);
    H           = tril(H)+tril(H,-1)'; H = frobproj(H,tol);
    x_sim(:,t)  = mvnrnd(h,H);
    loglik      = loglik+sum(log(mvnpdf(x_sim(:,t),h,H)));
end

% Projection onto the space of positive or semi-positive definite
% matrices minimizing the Frobenius distance.
% INPUTS: x=initial matrix
%         tol=values to assign to negative eigenvalues. If tol=0, y will
```

```
%            singular
% OUTPUT: y=resulting matrix

function y=frobproj(x,tol)

if min(eig(x))<=0,
    [A,B]=eig(x);
    B=diag(real(B));
    B(B<=0)=tol;
    y=A*diag(B)*A';
    y=tril(y)+tril(y,-1)';
else
    y=x;
end
```