Bayesian Inference for Statistics Recurrent Stochastic Volatility

Weihang Qiu*

* Corresponding author. Email: 3262975029@qq.com

Jinan University Birmingham University Joint College, Jinan University, Guangzhou, 511436, China

Abstract. Stochastic volatility model is an important model for studying financial time series. Due to the very complex volatility of financial markets, it is particularly important to analyze the characteristics of non-linearity and long memory. Moreover, the traditional stochastic volatility model does not explain the self-dependence in statistical sense, which will affect the model's capture of volatility effect.In this paper, a statistical recurrence volatility model is established. The cyclic neural network is used to model the time series so as to draw the long dependence at the moment, thus overcoming the limitation that the traditional random model can only explain the short dependence, and the nonlinear transformation is used to replace the logarithmic transformation of the traditional random model. We combine in-depth learning with stock volatility to simulate five international stock index data sets: Swiss stock index SWX, French market index CAC, and Singapore stock market index STI. In addition, SMC and particle MCMC are used to perform Bayesian inference on this parameter state space model. Compared with the other three traditional models, the Statistics Recurrence Stock Volatility model is superior to SV, NSV and LMSV in describing the complex fluctuation effect, and provides better off-sample prediction.In order to better combine RNN and SV models, we use a special type of RNN model, namely the Statistical Cyclic Unit (SRU) structure, which can capture the complex wave effects ignored by the traditional SV model while retaining the basic components of the SV model. This combination enables the SR-SV model to achieve flexible and excellent prediction performance by using in-depth learning, and fills up the weakness of neural network in explanation.

Keywords: Stochastic Volatility, cyclic neural network, nonlinearity, long memory.

1 Introduction

Volatility plays a significant role in financial transactions, such as option pricing and stock markets. Volatility is used to calculate the value at risk (VaR) of a portfolio, which is the variance of the return. However, unlike securities prices, fluctuations cannot be directly observed. Therefore, it is usually measured as the statistical fluctuation of the earnings from the market index, and then uses various statistical methods to measure the volatility of the earnings series. Classical academic methods include ARCH (autoregressive conditional heteroscedasticity), GARCH (generalized ARCH), TGARCH (threshold GARCH), EGACH (index GARCH), etc. These models can describe heteroscedasticity well, but they cannot explain the leverage effect of financial volatility, namely asymmetry. Moreover, GARCH models are essentially a linear expression of certainty, and cannot effectively describe the random oscillation of volatility. In the SV model, additional random terms are added by methods such as MCMC. Compared

BDEDM 2024, January 12-14, Ningbo, People's Republic of China

DOI 10.4108/eai.12-1-2024.2347143

with other models, the stochastic volatility model can model volatility as a stochastic process rather than a deterministic process. This allows us to obtain an approximate distribution of volatility for each time in the series and provides confidence for the forecast. However, Myron and Bruce used R/S analysis to study the price and return rate data of 200 stocks and confirmed that most of the price variables have the characteristics of long-term dependence. Huang and others found that there is long-term memory of absolute price movements in the stock and option markets. However, in the traditional time series modeling, the SV model represents a rapid decay process in geometric series, which is slower than the autocorrelation decay of the time series and cannot capture the long dependence well [1].

In order to solve this problem, Breidt, Crato and deLima proposed the Long Memory Stochastic Volatility (LMSV) model. They first proposed the ARFIMA model [2,3] based on the fractional difference noise (FDN) model, and then combined it with the SV model. The model is widely used to simulate the strong persistence and long memory in economic time series. However, since the ARFIMA model is linear and cannot describe the nonlinear transition, Yu, Yang and Zhang have introduced a series of nonlinear SV(N-SV) models. In the original lognormal SV model, the logarithmic fluctuation obeys the AR (1) process, and the smoothing function of the allowed fluctuation obeys the AR (1) process. From the final result, the nonlinear CARCH (1,1) model is generalized for random fluctuation, that is, the Box-Cox power function is used instead of the logarithmic function. Yu, Yang and Zhang have simulated the data of currency exchange and option pricing and the empirical results show that the N-SV model is feasible, but the choice of the model is very complicated. In this paper, we will discuss the out-of-sample prediction performance of LMSV and N-SV models and compare them with the model in this paper [4,5].

In order to characterize the long memory and nonlinear wave effects, this paper chooses RNN model which is famous for solving these problems. Cyclic neural network is a kind of neural network which takes sequence data as input, recurs in the evolution direction of sequence and all nodes are connected in chain. This is a powerful nonparametric tool and a key feature is the use of a feedback connection that allows the output of the network to be fed back into the network as an input to the next time step. This feedback loop enables the network to simulate the dependencies between different elements in the input sequence, while the artificial neural network can contain non-linearity and all variables. However, Makridakis, Spiliotis and Assimakopoulos note that the accuracy of machine learning models is generally lower than statistical models widely studied in financial time series literature (such as ARMA and ARIMA). Therefore, the use of traditional RNN is neglected, among which LongShort-TermMemorynetworks (LSTM) is a common recurrent neural network. LSTM solves the problem of long-term dependence in RNN. However, the internal work of lstm may be difficult to explain, and the performance of lstm is highly dependent on the choice of superparameters, such as the number of hidden units, learning rate and number of layers, which makes it challenging to find the optimal configuration.

Therefore, we need to develop volatility models based on RNN. These models not only have the ability of RNN to produce accurate forecasts, but also capture the volatility effect as well as AR(1) process in SV model, so that the models have interpretable and meaningful in-sample analysis.

In order to better combine the SV model with the RNN model, we use the Statistical Cyclic Unit (SRU) structure, which is a cyclic neural network (RNN) structure, introduced by Lei and Zhang in a paper in 2017. It is designed to be efficient and fast by minimizing the number of operations required to update the status. It has proven to perform well on a range of tasks, including language modeling, machine translation and speech recognition.

This combination enables the SR-SV model to not only enhance the strong interpretability of the fluctuation effect, but also add a strong prediction ability. The SR-SV model belongs to the parametric state space model, and its Bayesian inference can use the latest sequential Monte Carlo (SMC) and particle MCMC literature.

2 The traditional three models

2.1 SV model

Set is a series of revenue. Then the sv model isa $= \{at, t = 1, \ \ldots, \ T\}$

$$z_t = \mu + \phi(z_{t-1} - \mu) + \epsilon_t^z \ , \epsilon_t^z \sim N(0, \sigma^2), t = 2, \ldots, T,$$

$$z_1 \sim N\left(\mu, \frac{\sigma^2}{1-\varphi^2}\right) \tag{1}$$

$$y_t = e^{\frac{1}{2}z_t}\epsilon_t^y, \epsilon_t^y \sim N(0,1), t = 1, 2, \ldots, T. \tag{2}$$

The stochastic volatility (SV) model presented consists of two equations that describe the evolution of two variables: z_t and y_t. The variable z_t represents the log volatility process. It follows an autoregressive AR(1) model, which means its value at time t depends on its previous value, z_{t-1}, and is influenced by a mean parameter μ. The persistence parameter φ controls the influence of the previous value on the current value, ensuring the process remains stationary. The equation also includes a random error term ϵ_t^z, which captures the unpredictable component of z_t. The variable y_t represents the observed data, and it is related to z_t through an exponential transformation. The value of y_t at time t is determined by the exponential of half the value of z_t, multiplied by a random error term ϵ_t^y. The error term ϵ_t^y follows a standard normal distribution. The model assumes that the persistence parameter φ lies between -1 and 1 to maintain stationarity of both the z and y processes. This range ensures that the influence of past values does not grow too large or diminish too quickly. It assumes that the log volatility process z_t exhibits persistence, meaning that past values strongly influence current values. This persistence parameter φ is often close to 1 in financial data, indicating a strong auto-dependence. Overall, the SV model provides a framework for modeling the evolution of z_t and the relationship between z_t and y_t. It assumes an autoregressive structure for z_t and incorporates randomness through error terms. However, the SV model's assumption of an AR(1) process may not capture more complex dynamics in the log volatility process, such as long-term memory or nonlinear auto-dependence. Therefore, it is suggested to combine the SV model's structure with techniques from recurrent neural network (RNN) time series modeling to create a more flexible prior distribution for z_t. This flexible prior would allow for capturing more intricate dynamics beyond the limitations of the SV model.

2.2 NSV model

Yu,Yang,and Zhang proposed a class of nonlinear SV(N-SV) models, the core of which is that the use of Box-Cox-like changes to z_t can reduce the unobservable error and the correlation of prediction variables to a certain extent. We introduce a parameter δ to measure the degree of nonlinearity, and then estimate the parameter through the data itself to determine the form of data transformation that should be taken, which can cause a stable transformation of variance for continuous dependent variable unsatisfiednormal distributionThe variance is no longer dependent on the mean. Using a transformation, equation becomes

$$z_t = \mu + \phi(z_{t-1} - \mu) + \epsilon_t^z, \epsilon_t^z \sim N(0, \sigma^2), t = 2, \ldots, T,$$

$$z_1 \sim N\left(\mu, \frac{\sigma^2}{1-\varphi^2}\right) \quad (3)$$

$$y_t = (1 + \delta z_t)^{1/(2\delta)} \epsilon_t^y, \epsilon_t^y \sim N(0,1), t = 1, 2, \ldots, T, \quad (4)$$

According to model As, therefore, SV model NSV is a special case. $\delta \to 0$, $(1 + \delta z_t)^{1/(2\delta)} \to e^{\frac{1}{2}z_t}$.

2.3 LMSV model

Breidt, Crato and deLima used ARFIMA(p,d,q) process on z_t to capture the long memory self-dependence exhibited in financial time series. Their LMSV model was written：

$$(1 - B)^d \Phi(B) z_t = \Theta(B)\eta_t, \eta_t \sim N(0, \sigma_\eta^2), \quad (5)$$

$$y_t = \sigma_t \epsilon_t, \sigma_t = \kappa, \epsilon_t \sim N(0,1),$$

$$t = 1, 2, \ldots, T, \quad (6)$$

where $\Phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \varphi_p B^p, \Phi(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ and B is the backshift operator, that is, $B^s X_t t := X_{t-s}$, and the fractional integral parameter d is within (0.5, 0.5).

3 SRU model

Modeling time series data includes analyzing and predicting patterns and trends over time. The classic ARIMA (autoregressive integrated moving average) model uses a combination of autoregressive, differential and moving average terms to model the data. The rnn may incorporate information from previous time steps into the input sequence by using hidden states, each of which is recursively updated in the RNN as the network processes the sequence as a function of the previous hidden state and the input of the current time step. This cyclic update mechanism allows the RNN to retain memory of previously seen inputs. By learning to update hidden states based on input sequences, RNN can implicitly model the dynamics of data and capture time dependencies that are difficult to capture using other machine learning models.

Our goal is to find the density of z_t, so we have to model the density for a given sequence x_t . A priori p (z_t |x_t, $D_{1:t-1}$), where {D_t=(x_t,z_t), t=1,2,...} is the time series data of a period of time series has dependency, so a group of hidden units ht needs to feed itself by using the lag value h_{t-1} of its previous time step by using RNN. While RNN allows hidden units to connect

to their previous time step value, enabling the network to describe memory. The mathematical expression of the traditional RNN model (Elman1990) is

$$h_t = \Psi(w_x x_t + w_h h_{t-1} + b),\ \eta_t = \beta_0 + \beta_1 h_t,$$
$$z_t|\,\eta_t \sim p(z_t|\,\eta_t). \tag{7}$$

The model parameters include w_x, w_h, β_0, β_1;

$\Psi(x)$ is inartificial neural network On the neurons of the brainfunction, responsible for mapping the inputs of neurons to the outputs, can be increasedNeural network model The nonlinear, common sigmoid function, tanh(z) function and ReLU function.

The network consists of interconnected layers of nodes (neurons) that transform input data into useful representations and ultimately produce the desired outputs. We can find the mapping between input and output through training. $P(z_t|\,\eta_t)$ depends on the mapping we find. If z_t is continuous, then $p(z_t|\,\eta_t)$ is a Gaussian density of mean η_t. We believe that the neural network initially did not have any memory, i.e.h_1 was 0. Equation (7) suggests that the hidden state at time t is the output of a composite function

$$h_t = f(x_t, f(x_{t-1}, \ldots, f(x_1, h_0))), \text{where}$$
$$f(x_t, h_{t-1}) := \Psi(w_x x_t + w_h h_{t-1} + b), \tag{8}$$

When the gradient propagates in multiple layers of the depth neural network, the gradient becomes too small and the gradient disappears, which makes it difficult for the network to learn meaningful features. This problem is especially evident in the cyclic neural network, where gradients must propagate through many time steps. On the other hand, when the gradient becomes too large, there will be an explosive gradient problem, resulting in unstable training and numerical overflow error. This can happen in deep networks with large weights, where gradients grow exponentially as they propagate through the network.

In this paper, the weight is larger, the gradient of ht relative to the model parameters may explode or disappear, which makes the learning efficiency of the traditional RNN model in long time series very low. So we will use the statistical cyclic unit (SRU) to overcome the above problem, the SRU structure can be written as

$$r_t = \Psi(w_h h_{t-1} + b_r), \tag{9a}$$
$$\varphi_t = (w_r r_t + w_x x_t + b_\varphi), \tag{9b}$$
$$h_t^{(j)} = \alpha_j h_{t-1}^{(j)} + (1 - \alpha_j)\varphi_t, j = 1, \ldots, m;$$
$$h_t = (h_t^{(1)}, \ldots, h_t^{(m)})^T, \tag{9c}$$

Where $\alpha=(\alpha_1,\ldots,\alpha_m)\in(0,1)$ is a vector of average weights, rt and T help to calculate the hidden state ht, and the rest are model parameters. We simply record the function structure in (9a)-(9c) as h_t = SRU (x_t, h_{t-1}).SRU is a cyclic neural network with a cyclic layer and a gating mechanism that allows it to selectively forget or remember information from previous time steps. While the moving average structure involves taking a sliding window of an input sequence and calculating an average of the values in that window. This creates a smooth version of the input sequence that can be used for various tasks such as anomaly detection or signal processing. By combining the moving average structure with the SRU elements, we can create

a circular neural network that utilizes both techniques. The input sequence is first smoothed using a moving average structure and then the SRU element is input to capture the time dependency between the smoothed values. In general, this approach is very useful for capturing both short-term and long-term patterns in the input sequence.

4 SRSV model

The combination of SV and SRU models provides a flexible modeling approach that captures a wide range of dynamic patterns in the data. In this technique, the SV model can capture the overall volatility structure of the data, and the SRU structure is used to capture the long-term memory and nonlinear self-dependence that are not considered in the basic SV model, which can more accurately represent the underlying data generation process.

The mathematical expression of that combine model is:

$$r_t = \Psi(w_h h_{t-1} + b_r), t = 2, \ldots, T, (10a)$$

$$\varphi_t = \Psi(w_r r_t + w_\eta \eta_{t-1} - 1 + w_z z_{t-1} + b_\varphi), t = 2, \ldots, T, (10b)$$

$$h_t = \alpha h_{t-1} + (1 - \alpha)\varphi_t, t = 2, \ldots, T, h_1 = 0, (10c)$$

$$\eta_t = \beta_0 + \beta_1 h_t + \epsilon_t^\eta, \epsilon_t^\eta \overset{iid}{\sim} N(0, \sigma^2), t = 1, 2, \ldots, T, (10d)$$

$$z_t = \eta_t + \phi z_{t-1}, t = 1, \ldots, T, (10e)$$

$$y_t = e^{\frac{1}{2}z_t} \epsilon_t^y, \epsilon_t^y \overset{iid}{\sim} N(0, 1), t = 1, 2, \ldots, T, (10f)$$

In this technique, the initial value of the fluctuation process is set to z_0, which is the logarithm of the data variance, and the initial value of the SRU hidden state h_1 is set to 0 according to the previous discussion. The SR-SV model combines the advantages of the SRU and SV models. Firstly, the logarithmic volatility at time t in (10e) can be written as

$$z_t = \beta_0 + \beta_1 SRU(\eta_{t-1}, z_{t-1}, h_{t-1}) + \phi z_{t-1} + \epsilon_t^\eta . (11)$$

Therefore, the parameter β_1 better represents the influence in the potential fluctuation process Z, and makes up for the shortcoming that the AR(1) process can only capture the short-term linear influence.

In the traditional stochastic volatility (SV) model, the process volatility is modeled as a stable stochastic process. However, in many practical applications, the volatility of the data can exhibit erratic behavior, such as sudden peaks or changes in volatility over time. In this case equation (11) needs to be converted to

$$z_t = \beta_0 + \beta_1 \mathfrak{N}(\eta_{t-1}, w_z z_{t-1}, h_{t-1}) + \phi z_{t-1} + \epsilon_t^\eta , (12)$$

In the context of stochastic volatility (SV) model, the volatility process Z is usually modeled as a function of its previous value $z_{t-1}\}$. One way to model this dependence is through the weight parameter w_z, which determines how strongly the previous value of Z affects the current value of Z .. So here w_z represents the volatility, the value of w_z will affect the dependency between z_t, and if w_z is missing, z_t only depends linearly on z_{t-1}. The function N(x) repre-

sents a non-linear transformation of a given input x and is generally used to introduce non-linear dependencies between variables in the model. In the SV model, the function N(x) can be used to model the nonlinear dependence between the fluctuation process z and other variables in the model.

Secondly, SRU model has several super parameters, including hidden size, exit rate and scale parameter α. Specifying alpha values in advance may limit the model's flexibility to capture complex time dependencies in the data. In some cases, it may be difficult to know a priori which α values are optimal for a given data set. Also, summary statistics obtained by setting the scale. alpha. to a value specified in advance may be difficult to interpret and may not be able to clearly understand the basic patterns in the data.

In order to solve these problems, we can learn the parameter α in the SRU model from the data, usually through a process called back propagation. Back propagation is an optimization algorithm that uses gradient descent iteration to update the model parameters based on the error between the predicted output and the real output. Alpha controls the weight assigned to the previous hidden state when calculating a new hidden state. The higher the value of α, the higher the weight of the previous hidden state, and the lower the value of α, the lower the weight of the previous hidden state. By adjusting the value of α, the SRU model can learn different time dependencies in the captured data.

Third, neural networks are highly flexible models with many parameters that are easily over-fitted, which occurs when the model becomes too complex and begins to fit the noise in the data rather than the underlying pattern. Therefore, we inject noise to regularize the SR-SV model to prevent over-fitting. The noise injection is accomplished by allowing the previous fluctuation state z_{t-1} and the previous noise signal. eta. T-1 as inputs to the SRU layer in the network. The idea behind the noise injection is to add random disturbances to the input data, which helps to prevent the network from memorizing the training data and over-fitting.

Finally, in the SR-SV model, the parameter β_0 acts as a scaling factor for the variance of the observed variable y_t. More specifically, the parameter τ can be defined as $\tau = e^{\beta 0/2}$,y_t can be expressed as: $y_t = \tau\, e^{\frac{1}{2}z_t}\, \epsilon_t^y$,Where z_t is hidden and represents the logarithm of the volatility process.

Intuitively, τ controls the overall scale of the volatility process, which determines the uncertainty and change level of the observed data. By adjusting τ, the model can explain different levels of volatility and capture changes in data over time. Please note that in the SR-SV model, β_0 is not estimated directly, but is inferred from other parameters of the model during the training process. The value of τ can be used to explain the results of the model and to predict future observations.

5 Bayesian inference

Bayesian estimation and inference of SR-SV model involves estimating the posterior distribution of model parameters given the observed data. This usually uses SMC or Markov Chain Monte Carlo (MCMC) methods. For Bayesian estimation and inference of SR-SV model, the choice between SMC and MCMC methods depends on the specific requirements and con-

straints of the analysis, and may involve a trade-off between efficiency and accuracy. In general, SMC method may be more suitable for exploratory analysis and large-scale simulation, while MCMC method may be more suitable for rigorous reasoning and decision-making [6].

The likelihood function of the SR-SV model is:$p(y_{1:T}|\theta) = \int p\left(y_{1:\ t}\middle|z_{1:T},\theta\right)(z_{1:T}|\theta)dz_{1:T}$ Where p($y_{1:\ t}|z_{1:T}$, θ) is the likelihood of the observed value given the potential state and parameter, and p($z_{1:T}|\theta$) is the joint distribution of the potential states given the parameter.

$p(y_{1:\ t}) = \int_{\Theta} p(y_{1:\ t}|\theta)p(\theta)d\theta$is the marginal likelihood.$\theta$is the vector of 11 parameters of the SR-SV model, the SR-SV model is a state-space model that consists of a measurement equation and a state transition equation. In the measurement equation, the observed value at time t, denoted as y_t, is assumed to follow a normal distribution with mean zero and variance e^{z_t} The term e^{z_t} represents the measurement error at time t.and the state transition equation is

$$z_t|z_{1:t-1}, h_t \sim N(\phi z_{t-1} + \beta_0 + \beta_1 h_t, \sigma^2), t \geq 2,$$
$$z_1 \sim N(\beta_0, \sigma^2). (13)$$

However, SMC method is generally more effective and scalable than MCMC method for state space models of nonlinear and non-Gaussian models. The SMC method also provides a method for estimating the likelihood function, which is usually difficult to handle for nonlinear state space models and can be used for model selection and comparison. Once the posterior distribution is estimated using SMC, we can use it to make predictions and extrapolate model parameters.

5.1 SMC calculation of 5.1SRU-SV model

The density-adjusted sequential Monte Carlo (D-T-SMC) sampler is a variant of the sequential Monte Carlo (SMC) method. It is specially designed for Bayesian inference that the likelihood function is difficult to calculate, but can be sampled from a priori and a posteriori distributions. It uses a series of intermediate distributions, which are harmonic versions of posterior distributions, which makes it more efficient in exploring posterior distributions. The D-T-SMC method first samples M weighted particles $\{W_0^j, \theta_0^j\}_{j=1}^M$ from the sample distribution $\pi_0(\theta)$, such as a priori p(θ). Then, the particle traverses a series of k intermediate distributions $\pi_k(\theta)$ and the final distribution is the posterior distribution π (θ). The goal is to estimate the posterior distribution by combining the weighted particles from each intermediate distribution.

$$\pi_k(\theta) := \pi_k(\theta|y_{1:\ t}) \propto \hat{p}(y_{1:\ t}|\theta, u)^{\gamma_k} p(\theta), (14)$$

Where U is the auxiliary variable and γk is the normalization constant with integral of $\pi_k(\theta)$ being 1.

The DT-SMC method performs three main steps for each intermediate distribution:Weighting: the particle set $\{W_{k-1}^j, \theta_{k-1}^j\}_{j=1}^M$ in the previous intermediate distribution $\pi_{k-1}(\theta)$ is re-weighted to approximate the target distribution $\pi_k(\theta)$.Resampling: The resampling step is performed to avoid particle degeneracy, which occurs when some particles have very low weights and others have very high weights. In the DT-SMC method, a resampling step is per-

formed when the effective sample size (ESS) is below a preset threshold. In the resampling step, substitutions are made from the current set of particles to sample m new sets of particles with a probability proportional to their weights.

Markov Mobility: The Markov Mobility step updates particles using a Markov kernel that aims to move the particles to a target distribution. Markov kernels can be chosen in different ways depending on the problem at hand. For example, the Pseudo-marginal Metropolis-Hastings (PMMH) algorithm can be used to estimate the unbiased likelihood in a Markov moving step.

The DT-SMC method iterates through these steps until the target distribution $\pi_k(\theta)$=π(θ) is reached, and then generates a final estimate based on the weighted particle set of the last intermediate distribution.

When using the state space model for Bayesian inference, the likelihood function is usually difficult to handle, and the PMMH (Particle Marginal Metropolis-Hastings) method can be traditionally used. However, this method is computationally inefficient and the quality of the proposal distribution may be poor, which may lead to low acceptance rates and slow convergence of the Markov chain. The PMMH method may not be applicable to our model with high-dimensional parameter space or model with high likelihood nonlinearity or multi-modes [7].

In order to solve this problem, Deligiannidis, Doucet and Pitt proposed a new method called Pseudo-edge Markov Chain Monte Carlo Algorithm (CPM). This method uses the Met-Hastings step and the pseudo-edge step, and introduces a set of auxiliary random variables to estimate the likelihood ratio in the MH acceptance probability. Specifically, the CPM method introduces a small perturbation to generate a new likelihood estimate, which is proportional to the difference between the current estimate and the previous estimate, which enables the algorithm to converge to the true likelihood value more quickly. This introduces the correlation between the auxiliary variables and the proposed parameters, which helps to reduce the variance of the likelihood ratio estimator and improve the mixing of Markov chains.

In addition, the DT-SMC method provides marginal likelihood estimation as a by-product of the algorithm. This is because the method involves calculating the normalized constant of the posterior distribution, which is proportional to the marginal likelihood. Marginal likelihood is the probability of observing data under a given model, integrating all possible values of model parameters. It can be used to compare the relative goodness of fit of different models, taking into account the complexity of the models and assessing the overall quality of the reasoning performed by the algorithm.

5.2 Model selection of marginal likelihood

According to the above discussion, the DT-SMC method provides an unbiased estimate of the marginal likelihood as a by-product of the particle filtering process. Suppose we have two models $M_1$and $M_2$with parameters θ_1 and θ_2, respectively. The marginal likelihood of the two models is:

$$p(y|M_1) = \int p(y|\theta_1, M_1)p(\theta_1|M_1)d\theta_1$$

$$p(y|M_1) = \int p(y|\theta_2, M_2)p(\theta_2|M_2)d\theta_2$$

Where p(y|θ, m) is the likelihood function of the model and p(θ|M) is the prior distribution of the parameters. The ratio of the marginal likelihood of the two models is called the Bayesian factor: B(M_1,M_2)=p(y|M_1)/p(y|M_2)

If B(M_1,M_2)>1, the M_1model is better than the M_2 model. The larger the Bayesian factor, the stronger the evidence supporting M_1 [8].

6 Performance assessment

Table 1. Prior distribution of the parameters.

SR-SV		SV		N-SV	
Parameter	Prior	Parameter	Prior	Parameter	Prior
$\boldsymbol{\beta 0}$	N(0,0,1)	μ	N(0,25)	μ	N(0,25)
$\frac{\boldsymbol{\phi}+\mathbf{1}}{\mathbf{2}}$	Beta(20,1.5)	$\frac{\phi+1}{2}$	Beta(20,1.5)	$\frac{\phi+1}{2}$	Beta(20,1.5)
$\boldsymbol{\sigma^2}$	IG(2.5,0.25)	σ^2	IG(2.5,0.25)	σ^2	IG(2.5,0.25)
$\boldsymbol{\beta 1}$	IG(2.5,1)			δ	N(0,0,1)
$\boldsymbol{\alpha}$	Beta(2,2)				
$\mathbf{W_h}, \mathbf{W_\phi}, \mathbf{W_\eta}$	N(0,0,1)				
$\mathbf{b_r}, \mathbf{b_\phi}$	N(0,0,1)				
$\mathbf{W_z}$	IG(2.5,1)				

We use DT-SMC sampler to perform bayesian inference on SV, N-SV and SR-SV models. Different from GARCH, LMSV model has no explicit state space representation and its likelihood function is difficult to analyze. Therefore, the frequency domain estimation method is often used to estimate the model parameters. We now motivate the prior choices in Table 2. We follow Yu, Y ang, and Zhan g and kim, shephard d, andchib to set the same a priori, i.e. Beta distribution. Table 1 shows the prior distribution of the parameters in the three models.

The three models here are SV, N-SV and SR-SV The persistence parameter φ is specific to these models. In all models, the parameter σ^2 is inverse prior, but the prior selection of other parameters is different. In the SR-SV model, the beta0 value of intercept is β_0, and the normal prior has a small square variance, because a small value of β_0 is often observed in the experimental results. The parameters β_1 and w_z in the SR-SV model have a unimodal posterior distribution under the inverse prior condition. For other SRU parameters, the normal prior with zero mean and small square variance proposed by RNN is used. The empirical results show that these parameters tend to be very small.

Table 2. Forecast scores used to measure off-sample performance.

Score	Definition	Score	Definition
PPS	$-\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}\log p(y_t\|y_{1:t-1},\hat{\theta})$	MSE_1	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}(\sigma_t-\hat{\sigma}_t)^2$
QLIKE	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}(\log(\hat{\sigma}_t^2)+\sigma_t^2\hat{\sigma}_t^{-2})$	MSE_2	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}(\sigma_t^2-\hat{\sigma}_t^2)^2$
$\mathrm{R^2LOG}$	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}\left[\log(\sigma_t^2\hat{\sigma}_t^{-2})\right]^2$	MSE_1	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}\lvert\sigma_t-\hat{\sigma}_t\rvert$
		MSE_2	$\mathrm{T}_{\mathrm{test}}^{-1}\sum_{\mathrm{D_{test}}}\lvert\sigma_t^2-\hat{\sigma}_t^2\rvert$.

Table 2 lists the forecast scores used to measure off-sample performance. In this assessment, the estimated posterior mean of the in-sample data is compared to the out-of-sample data using one-step prediction. Our goal is to see if the model can predict well future data not used in the model fit. The smaller the forecast score, the better the forecast performance for the out-of-sample data.

6.1 Simulator investigation

In this paper, a time series of T=3000 observations is generated, which is recorded as three simulation models of SIMI, SIMII and SIMIII (see Table 4). Model 1 is a GARCH (1,1) model [9], while Model 2 is an extension of GARCH(1,1), and Model 3 is a FIGARCH(1,d,1) model. The first 2,000 observations for each dataset are used for model estimation and the last 1,000 observations are used for off-sample analysis. Table 3 introduces the simulation model of the three experiments [10].

Table 3. Simulation: Data generating process.

Data	Model	Parameter
EXP I	$\sigma_t^2 = \mu + \alpha y_{t-1}^2 + \beta\sigma_{t-1}^2, t = 2, \dots, T$ $y_t = \sigma_t\epsilon_t, \epsilon_t \sim N(0,1), t = 1, \dots, T$	$\sigma_t^2 = 0.1, \mu = 0.1$ $\alpha = 0.07, \beta = 0.92$
EXP II	$h_t = \mu + \alpha\frac{(y_{t-1}^2)^{\delta} - 1}{\delta} + \beta h_{t-1}, t = 2, \dots, T$ $y_t = (1 + \delta h_t)^{\frac{1}{2\delta}}\epsilon_t, \epsilon_t \sim N(0,1), t = 1, \dots, T$	$h_1 = 0.1, \beta = 0.1$ $\alpha = 0.15, \beta = 0.82, \delta = 0.9$
EXP III	$\sigma_t^2 = \mu + [1 - \beta B - (1 - \phi B)(1 - B)^d]y_t^2 + \beta\sigma_{t-1}^2, t = 2, \dots, T$ $y_t = \sigma_t\epsilon_t, \epsilon_t \sim N(0,1), t = 1, \dots, T$	$\sigma_t^2 = 0.1, \mu = 0.01$ $\phi = 0.01, \beta = 0.5, d = 0.62$

Table 4. Posterior means of the parameters with the posterior standard deviations in brackets.

		μ	ϕ	σ^2	α	β_0	β_1	w_z	Mar. llh
EXP I	SV	2.147	0.938	0.021					-5100.7
		(0.229)	(0.005)	(0.004)					(0.136)
	SR		0.986	0.027	0.520	0.037	0.392	0.225	-5099.9
	− SV		(0.022)	(0.006)	0.169()	(0.029)	(0.227)	(0.262)	(0.308)
EXP II	SV	1.057	0.969	0.038					-4060.4
		(0.129)	(0.010)	(0.005)					(0.158)
	SR		0.798	0.056	0.541	0.037	0.432	0.503	-4057.6
	− SV		(0.105)	(0.011)	(0.146)	(0.043)	(0.199)	(0.267)	(0.298)
EXP III	SV	0.155	0.968	0.050					-3147.0
		(0.341)	(0.007)	(0.006)					(0.201)
	SR		0.894	0.045	0.609	-0.105	0.343	0.315	-3144.3
	− SV		(0.027)	(0.012)	(0.261)	(0.052)	(0.131)	(0.098)	(0.324)

Table 4 shows the posterior estimates of the SV and SR-SV model parameters, and the last column shows the marginal likelihood estimates. We summarize the estimation results and conclusions of the SV and SR-SV model fitting simulation data (SIMI, SIMII and SIMIII). From the estimation results, we can draw the following conclusions: in SIMII and SIMIII, the distance between the estimated posterior mean value of β_1 and w_z and zero exceeds two standard deviations, indicating that there is volatility effect rather than linearity in volatility dynamics, while in SIMI, the distance between $\beta_1$and zero is less than two standard deviations, indicating that only simple linearity effect can be detected in volatility dynamics of this

data set. For SIMI,SV and SR-SV models are very suitable. If the real data generation process has no other effect in volatility dynamics than short memory linear self-dependence, SR-SV model is close to SV model. For SIMII and Simiii, the additional neural network structure of the SR-SV model effectively captures the volatility effects ignored by the basic SV model.

Table 5 reports the predicted performance scores for the SV and SR-SV models with Monte Carlo standard error. The SR-SV model outperforms the SV model in all forecast scores for SIMII and SIMI3, and the SR-SV model outperforms the SV model in all scores except the PPS score for SIMI, which demonstrates the impressive off-sample forecast capability of the SR-SV model.

Table 5. Simulation: Forecast performance of the SR-SV and SV models.

		PPS	MSE1	MSE2	MAE1	MAE2	QLIKE	R2LOG	COUNT
EXP I	SV	3.510	1.010	0.995	0.980	0.920	0.994	0.993	one
		(0.001)	(0.003)	(0.003)	(0.002)	(0.004)	(0.001)	(0.004)	
	SR − SV	3.490	0.955	0.865	0.765	0.710	1.030	0.880	six
		(0.000)	(0.002)	(0.003)	(0.001)	(0.002)	(0.001)	(0.003)	
EXP II	SV	2.340	0.475	0.710	0.775	0.825	0.775	1.070	0
		(0.000)	(0.000)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
	SR − SV	2.330	0.225	0.445	0.485	0.585	0.530	0.880	seven
		(0.000)	(0.002)	(0.003)	(0.002)	(0.000)	(0.003)	(0.003)	
EXP III	SV	1.670	0.990	1.020	0.920	0.975	0.945	0.710	0
		(0.001)	(0.004)	(0.004)	(0.003)	(0.003)	(0.004)	(0.004)	
	SR − SV	1.660	0.780	0.805	0.640	0.725	0.685	0.465	seven
		(0.000)	(0.004)	(0.003)	(0.002)	(0.003)	(0.003)	(0.002)	

7 Stock forecast

This section studies and analyses stock market data for different indices, including CAC,SWX and STI. The data are derived from the realized pool and the analysis is based on a revenue reduction process, calculated using the adjusted closing price. The analysis is divided into in-sample and out-of-sample parts, of which the first 2,000 returns are used for in-sample analysis and the remaining 1,000 returns are used for out-of-sample analysis. Table 6 provides descriptive statistics on the degradation benefits of the CAC, SWX and STI datasets. Descriptive statistics include minimum, maximum, standard deviation, skewness, and kurtosis, which are calculated based on the return value of each dataset.

Table 6. Descriptive statistics for the demeaned returns of the CAC, SWX and STI datasets.

	Min	Max	Std	Skew	Kurtosis	Vn(10)	Vin(20)	Vin(30)
CAC	-7.434	9.987	1.265	0.113	10.964	3.229	2.503	2.148
						2.460	1.928	1.672
SWX	-11.619	12.153	1.185	0.306	17.0547	3.937	3.033	2.590
						2.567	2.091	1.846
STI	-7.218	6.660	1.130	-0.323	7.386	3.785	2.979	2.578
						3.021	2.456	2.168

The table 6 shows the test statistics for the long memory modified R/S test of lag Q performed

by Lo on the absolute and squared gains for each data set. This test is used to determine whether the return has a long-term memory, which means that the autocorrelation of the return persists over time. The value of Vn(q) indicates the degree of long memory, and the greater the value, the stronger the evidence of long memory. From the table 6, we can see that all data sets show evidence of long memory, as indicated by the significant asterisk. Different data sets have different Vn(q) values, SWX data set has the highest Vn(q) value and CAC data set has the lowest Vn(q) value, which indicates that SWX data set has the strongest long-memory evidence among the three data sets. The analysis shows that there is volatility clustering effect in financial data. This study uses different realized volatility measures, including realized variance (RV), double power variance (BV), median realized volatility (MedRV) and realized kernel variance (RKV), to evaluate the prediction performance of a given data set.

7.1 Sample analysis

Table 7. Applications: Posterior means of the parameters with the posterior standard deviations in brackets.

		μ	ϕ	σ^2	δ	α	β_0	β_1	w_z	Mar. llh
CAC	SV	-1.030	0.985	0.035						-2869.3
		0.236	0.004	0.006						-0.167
	$N-SV$	-0.135	0.973	0.034	-0.195					-2870.4
		0.218	0.005	0.006	0.082					0.228
	$SR-SV$		0.868	0.061		0.609	-0.122	-0.406	-0.393	-2866.8
			0.056	0.018		0.196	0.059	0.197	0.151	0.297
SWX	SV	-0.208	0.983	0.024						-2688.0
		0.317	0.004	0.004						0.182
	$N-SV$	-0.369	0.981	0.024	-0.245					-2687.0
		0.267	0.003	0.007	0.078					0.211
	$SR-SV$		0.827	0.051		0.787	-0.199	0.532	0.384	-2683.8
			0.059	0.024		0.134	0.081	0.258	0.137	0.334
STI	SV	-0.216	0.975	0.041						-2789.1
		0.234	0.007	0.010						0.229
	$N-SV$	-0.220	0.977	0.042	-0.195					-2789.3
		0.261	0.006	0.007	0.086					0.237
	$SR-SV$		0.847	0.091		0.783	-0.183	0.445	0.359	-2786.2
			0.081	0.047		0.249	0.197	0.334	0.326	0.407

According to Table 7, the analysis was conducted on the fitting of SV, N-SV, and SR-SV models to three datasets. The results indicate that the SR-SV model provides a better fit to the data compared to the SV and N-SV models, as evidenced by higher log-marginal likelihood values and Bayes factors exceeding $e^{2.3}$ in all cases.

The SR-SV model captures more general volatility effects, such as nonlinearity and long-memory auto-dependence, as indicated by the posterior means of the nonlinearity long-memory parameter (β_1), fractional integration parameter (d), and the nonlinear parameter (w_z). These parameters are all significantly different from zero, suggesting strong evidence of long-memory dependence and more general serial dependence in the volatility dynamics of the datasets.

Furthermore, the SR-SV model exhibits a moving average weight parameter (α) with modes close to 1, indicating its ability to capture short-term dependence on volatility. This feature is similar to the results observed in previous simulations.

It is important to note that the persistence parameter (φ) in the SR-SV model is consistently smaller than the persistence parameters in the SV and N-SV models. This is attributed to the significance of the parameter (w_z) in the SR-SV model, which reduces the linear effect of past volatility on current volatility. This finding contrasts with the scenario where no volatility effects are present, resulting in similar persistence parameters between the SV and SR-SV models.

Overall, the SR-SV model demonstrates a superior data fit compared to the SV and N-SV models. It effectively captures mean regression behavior, exhibits strong volatility persistence, incorporates a moving average component that captures short-term dependence on volatility, and detects more general serial dependence beyond linear relationships. The estimation results also indicate the presence of a more general fluctuation effect, supported by the long memory parameter and fractional integral parameter exceeding two standard deviations from zero.

In conclusion, the SR-SV model offers a comprehensive and suitable approach for analyzing the datasets, surpassing the SV and N-SV models in terms of data fit and capturing a broader range of dynamics in volatility. Table 8 shows a model diagnosis of filtered log volatility and residuals for various stock market indices (CAC, SWX, and STI) using different volatility models (SV, N-SV, LMSV, and SR-SV).

Table 8. Applications: Model diagnostics of the filtered log volatility and residual $\widehat{\epsilon_t}^{y}$.

		Filtered volatility				Residual $\widehat{\epsilon_t}^{y}$			
		Mean	Std	Kurtosis	Skew	Std	Kurtosis	Skew	LB − $\widehat{\epsilon_t}$
CAC	SV	1.542	1.952	24.709	4.116	0.998	2.849	-0.214	0.981
	N − SV	1.602	2.313	38.798	5.195	0.985	2.772	-0.211	0.973
	LMSV	1.431	1.684	13.965	2.936	1.005	3.632	-0.162	0.901
	SR − SV	1.342	1.572	26.615	4.229	0.994	2.852	-0.206	0.965
SWX	SV	1.353	2.108	56.192	6.245	0.971	2.830	-0.039	0.166
	N − SV	1.422	3.179	182.195	11.563	0.971	2.804	-0.027	0.232
	LMSV	1.286	1.656	22.736	3.798	1.006	4.016	-0.027	0.327
	SR − SV	1.114	1.673	51.864	5.938	0.983	2.798	-0.051	0.136
STI	SV	1.429	1.565	13.586	2.859	0.988	2.737	-0.119	1.03
	N − SV	1.496	1.881	22.185	3.722	0.987	2.705	-0.113	1.106
	LMSV	1.292	1.458	11.899	2.662	1.004	3.585	-0.054	1.109
	SR − SV	1.172	1.164	13.401	2.733	0.991	2.731	-0.134	1.107

The "Filtered Volatility" column provides the average, standard deviation, kurtosis and skewness of the filtered logarithmic volatility for each model and index. The Residual column provides the same statistics for the residuals of the corresponding model. The "lb -p Value" column shows the P value of the Ljung-Box test with 10 lags, which tests for the presence or absence of autocorrelation in the residuals.

On the whole, based on these diagnostic measures, the model seems to perform quite well without obvious signs of irregularity or inappropriateness. LMSV models generally show the lowest kurtosis and skewness values of filtered volatility and residuals, indicating a more symmetrical and peak distribution. The SR-SV model generally shows the lowest lb -p value, indicating the best fit for the data in terms of residual autocorrelation.

7.2 Off-sample analysis

Table 9. Applications: Summary statistics on the one-step-ahead out-of-sample forecast conditional variancesand residual . $\widehat{\boldsymbol{\sigma}}_{\mathbf{t}}^{\mathbf{2}}\widehat{\boldsymbol{\epsilon}}_{\mathbf{t}}\mathbf{c}$.

eight		Filtered volatility				Forecast residual$\widehat{\epsilon}_t^y$			
		Mean	Std	Kurtosis	Skew	Std	Kurtosis	Skew	LB $-\widehat{\epsilon}_t$
CAC	SV	1.083	0.590	3.047	0.729	0.998	4.016	-0.329	0.653
	N − SV	1.086	0.598	3.527	0.912	0.993	3.989	-0.337	0.660
	LMSV	1.097	0.626	3.530	0.898	1.003	3.994	-0.305	0.578
	SR − SV	0.957	0.468	3.322	0.898	1.035	3.886	-0.323	0.598
SWX	SV	0.781	0.496	13.284	2.715	0.990	4.171	0.042	0.424
	N − SV	0.776	0.524	22.141	3.665	0.989	4.116	0.027	0.413
	LMSV	0.867	0.503	9.332	2.003	0.936	4.283	0.063	0.327
	SR − SV	0.618	0.343	15.776	3.015	1.099	3.943	-0.015	0.407
STI	SV	1.027	0.663	4.463	1.111	0.933	3.874	-0.633	0.849
	N − SV	1.029	0.689	5.346	1.391	0.935	3.699	-0.629	0.801
	LMSV	1.118	0.675	6.119	1.563	0.941	3.940	-0.555	0.772
	SR − SV	0.846	0.466	4.271	1.032	0.965	3.458	-0.562	0.724

For each index, four different volatility models are considered: SV,N-SV,LMSV and SR-SV. Filtered volatility and forecast residuals for mean, standard deviation, kurtosis and skewness are reported. In addition, the LBp values of 10 lagging Ljung-Box tests are also provided in Table 9.

In general, the average filtered volatility and forecast residual error are different between different models and indices. However, the standard deviation of the prediction residual error is usually low, which indicates that the model can capture most of the changes in the data. The kurtosis and skewness of the forecast residual error are also different between different models and indexes, which indicates that the model can capture different aspects of the data volatility structure. The LBp value of the residuals is usually high, indicating that the model can capture the autocorrelation structure in the data.

The marginal likelihood estimation shows that the SR-SV model is more suitable for the in-sample data of five exponential data sets than the SV and N-SV models. The use of noise injection regularization in the SR-SV model helps to prevent known over-fitting problems, and the forecast volatility and forecast residuals that are one step ahead of the meaner SV and N-SV models appear to perform well. In all five index data sets, the SR-SV model is one step ahead of the others in predicting the mean and standard deviation of volatility. SR-SV forecast is usually more conservative in low volatility period, and the forecast interval usually has a smaller band than other models.

Table 10 provides performance indicators for different models, predicting the Standard & Poor's 500 Index (SPX) one step ahead using different implementation measures. These indicators include PPS (symbolic forecast probability), MSE1 and MSE2 (mean squared error of original and excess returns), MAE1 and MAE2 (mean absolute error of original and excess returns), QLIKE (quasi-log likelihood), R2LOG (log r squared) and Count (number of times the model has the lowest or best forecast score). The models include benchmark SV model, non-singular value (N-SV) model, local median SV(LMSV) model, generalized power volatil-

ity (GP-Vol) model and stochastic horizontal stochastic volatility (SR-SV) model.

Table 10. SPX data: One-step-ahead forecast performance of the SR-SV and benchmark models using different realized measures, with the Monte Carlo standard errors in brackets.

five		PPS	MSE_1	MSE_2	MAE_1	MAE_2	QLIKE	R^2	Count
BV	SV	1.234	0.111	1.543	0.265	0.456	0.404	0.777	0
		0.002	0.001	0.003	0.002	0.002	0.002	0.003	
	$N-SV$	1.223	0.123	1.537	0.246	0.435	0.403	0.780	0
		0.001	0.001	0.003	0.002	0.002	0.002	0.004	
	LMSV		0.138	1.547	0.292	0.497	0.443	0.922	0
	$GP-VOL$		0.194	1.746	0.339	0.609	0.599	1.205	0
	$SR-SV$	1.229	0.100	1.532	0.190	0.323	0.301	0.520	8
		0.002	0.000	0.003	0.000	0.002	0.003	0.000	
MedRV	SV		0.117	0.834	0.259	0.421	0.362	0.920	0
			0.000	0.002	0.001	0.000	0.001	0.003	
	$N-SV$		0.117	0.830	0.260	0.421	0.362	0.924	0
			0.000	0.002	0.001	0.000	0.001	0.004	
	LMSV		0.143	0.918	0.300	0.496	0.362	0.924	0
	$GP-VOL$		0.196	1.370	0.339	0.600	0.575	1.447	0
	$SR-SV$		0.104	0.826	0.238	0.391	0.311	0.762	0
			0.000	0.002	0.000	0.000	0.001	0.001	
RKV	SV		.0144	0.833	0.256	0.418	0.364	0.914	0
			0.001	0.000	0.000	0.001	0.000	0.002	
	$N-SV$		0.143	0.831	0.256	0.419	0.362	0.915	0
			0.001	0.002	0.000	0.000	0.000	0.001	
	LMSV		0.140	0.902	0.288	0.474	0.402	1.099	0
	$GP-VOL$		0.180	1.035	0.315	0.547	0.499	1.259	0
	$SR-SV$		0.143	0.821	0.238	0.386	0.346	0.807	6
			0.000	0.001	0.000	0.000	0.000	0.001	
RV	SV		0.128	1.862	0.246	0.421	0.332	0.796	0
			0.000	0.002	0.001	0.002	0.000	0.002	
	$N-SV$		0.128	1.861	0.247	0.422	0.330	0.799	0
			0.000	0.002	0.001	0.001	0.001	0.003	
	$GP-VOL$		0149	1.947	0.282	0.483	0.386	0.982	0
	LMSV		0.191	2.044	0.318	0.570	0.487	1.188	0
	$SR-SV$		0.137	1.859	0.224	0.381	0.317	0.691	six
			0.000	0.001	0.000	0.000	0.001	0.000	

In general, the SR-SV model outperforms the benchmark SV model and other models in most performance indicators. The SR-SV model has the highest PPS value, the lowest MSE1 and MSE2 values, and the lowest MAE1 and MAE2 values. In most cases, it also has the highest QLIKE and R2LOG values. The N-SV model and the LMSV model also show good performance in some cases, while the GP-Vol model has been underperforming. The results show that the proposed SR-SV model can predict the SPX index better than the benchmark SV model and other existing models. The results also highlight the importance of using appropriate implemented measures for volatility forecasting, as different measures can result in different model performance. Judging from the out-of-sample prediction performance, the SR-SV model is always better than all other models on all data sets, further proving that it does not over-fit the data. The performance of SV and N-SV models is mixed, and no one model always performs better than the other. The predictions of LMSV and GP-Vol models have been the least accurate.

8 Conclusion

In this paper, a new volatility modeling and forecasting method is proposed, which combines the statistical cycle unit structure in machine learning with the stochastic volatility model in financial econometrics. The resulting SR-SV model proved to be an efficient model for volatility modeling and forecasting, and was able to capture various volatility effects ignored by the SV benchmark model. However, the SR-SV model still has many deficiencies that need to be further explored: the SR-SV model needs to estimate many parameters, including the weights and deviations of the recurrent neural network and the super-parameters of the stochastic volatility model. Estimates may require significant calculations and may require careful adjustments to the optimization algorithm. The SR-SV model is a model that requires a large amount of data, which means that it needs a large amount of training data to accurately understand the potential fluctuations. If the training data is limited or noisy, the model may perform poorly. The SR-SV model assumes that the potential volatility dynamics can be modeled using a stochastic volatility model and that the time series of earnings and volatility are linearly correlated. These assumptions may not hold true in all cases, especially in markets with complex nonlinear dynamics.

References

[1] Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica, 50(4), 987-1008. doi: 10.2307/1912773

[2] Harvey, A. C., & Chakravarty, T. (2008). Beta-t-(N-SV) models for financial time series. Journal of Business & Economic Statistics, 26(3), 349-360. doi: 10.1198/073500107000000406

[3] Kim, C. J., Shephard, N., & Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with ARCH models. Review of Economic Studies, 65(3), 361-393. doi: 10.1111/1467-937x.00044

[4] Wu, F., Ding, S., & Zhu, X. (2019) . Large-scale language modeling with maximal mini-batch backpropagation through transposed recurrent connections. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 2393-2403).

[5] Corsi, F. (2009). A simple approximate long-memory model of realized volatility. Journal of Financial Econometrics, 7(2), 174-196. doi: 10.1093/jjfinec/nbn004

[6] Robert, C. P., & Casella, G. (2004). Monte Carlo statistical methods (2nd ed.). Springer Science & Business Media.

[7] Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing, 10(3), 197-208. doi: 10.1023/A:1008935410038

[8] Singh, S. S., & Doucet, A. (2017). A general framework for sequential Monte Carlo sampling using determinantal point processes. Journal of Machine Learning Research, 18(98), 1-36.

[9] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307-327.

[10] Baillie, R. T., Bollerslev, T., & Mikkelsen, H. O. (1996). Fractionally integrated generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 74(1), 3-30.