Computational Economics (2025) 66:2199–2223
https://doi.org/10.1007/s10614-024-10777-0

Fast Computation of Randomly Walking Volatility with Chained Gamma Distributions

Di Zhang[1] · **Youzhou Zhou**[2]

Accepted: 26 October 2024 / Published online: 21 November 2024

Abstract

Volatility clustering is a ubiquitous phenomenon in financial time series analysis. In this study, we propose a Dynamic Bayesian Network (DBN) that leverages the conjugate prior relationships of normal-gamma and gamma-gamma distributions, preserving local posterior invariance at each node within the network. By incorporating dummy gamma nodes, we ensure that the model's volatility follows an independent incremental process. This proposed model offers two distinct advantages: (1) it exhibits heavier tails (manifesting as positive excess kurtosis) compared to Gaussian distributions, thereby contrasting sharply with conventional linear models; (2) it utilizes Variational Inference (VI) to expedite state estimation relative to Monte Carlo (MC) methods, ensuring deterministic convergence. In comparative experiments involving eight established methodologies and four variations of our approach across eight datasets, we observed the following: (1) When employing MC, our model achieves volatility forecasting performance comparable to leading methods; (2) when utilizing only VI, it attains acceptable accuracy, particularly for high-frequency data, albeit slightly lower than the state-of-the-art; (3) notably, VI reduces the runtime of our Gam-Chain model to generally less than 5% of that of MC-based methods.

Keywords Stochastic volatility · Variational inference · Dynamic Bayesian network · Cryptocurrency

This work was supported by projects TDF22/23-R25-196 and CoESE232403 from Xi'an Jiaotong-Liverpool University. Contribution Statement: The conception, experimentation, and writing of the paper were completed by Di Zhang. The proofs in the appendix of the paper were completed by Youzhou Zhou.

Extended author information available on the last page of the article

1 Introduction

In financial markets, asset prices exhibit continuous fluctuations, with the intensity of these variations commonly characterized by the logarithmic variance[1] of returns.[2] Numerous financial applications, including risk management, derivatives pricing, and portfolio management, necessitate a reliable estimation of current volatility. Prolonged observations have revealed two key findings: firstly, the volatility inherent in time series data is not constant but appears to adhere to another distribution, resulting in the heavy-tailed phenomenon of returns; secondly, the underlying distribution of volatility is not static but evolves over time, displaying a certain level of positive autocorrelation.

A widely recognized model addressing this phenomenon is Stochastic Volatility (SV) (Andersen & Benzoni, 2009). This model initially defines an unobservable stochastic process to depict volatility changes, subsequently employs these changes as instantaneous variances of returns, and ultimately generates a sequence of observable values. When volatility changes are modeled using a linear Gaussian framework, such as the Autoregressive Moving Average Model (ARMA), this approach yields interpretable results and ensures computational convergence. However, it also possesses two notable limitations: (1) empirical data analysis indicates that the increments of volatility do not necessarily follow a Gaussian distribution and often exhibit heavy tails; (2) in estimating volatility, the linear Gaussian model lacks a closed-form posterior. Consequently, it relies, or partially relies, on sampling to represent the volatility distribution as a collection of particles. This sampling process is frequently time-consuming. Furthermore, when integrated with parameter estimation within the Expectation-Maximization (EM) framework, assessing the convergence of the entire process becomes non-trivial.

To address this issue, we revisit the problem from the perspective of Dynamic Bayesian Networks (DBNs). We observe that the conventional Stochastic Volatility (SV) model can be interpreted as a two-layer state space model (Bishop, 2006), wherein volatility changes are defined in the transition equation, and price changes are defined in the observation equation. To facilitate computation, we reformulate the model as follows: (1) Given that the rate parameter of a gamma distribution has a conjugate prior that is also a gamma distribution, we initially utilize a sequence of interconnected gamma distributions to represent volatility. (2) While this direct connection implies that the obtained volatility increment is not independent but relies on the absolute value of the previous step, the increments between any two consecutive steps remain independent, akin to a random walk. (3) At each interval, we associate the volatility with the first gamma distribution through a prior on the precision (i.e., the reciprocal of variance) of a normal distribution. (Alternatively, we insert a 'dummy node' between adjacent

[1] Although variance, standard deviation, or logarithmic standard deviation are also employed in various literatures, for the sake of brevity, this paper uniformly adopts 'volatility' to represent 'logarithmic variance'.

[2] i.e., rate of returns.

steps.) (4) The ensemble of all random variables corresponding to normal distributions forms the return sequence, which is observable.

The primary benefit of this model lies in its computational efficiency. At each node within the model, the posterior retains the form of a gamma distribution, enabling relatively swift approximate state estimation. Although convergence may necessitate multiple scans of the sequence, as opposed to the two passes typically required by the forward-backward approach commonly employed in DBNs, this is not a concern in practical applications. This is because model parameters cannot be predetermined manually; hence, state estimation is often conducted within the iteration of the EM algorithm. Even if state estimation only involves a single scan of the sequence per round, it will ultimately converge to a (local) optimum alongside the model parameters. Furthermore, elements that remain constant across each scan can be precomputed outside the EM loop, further accelerating the process (Sect. 3.5).

The significance of this model extends beyond mere computational efficiency. It can be demonstrated that this construction effectively implements a random walk of volatility. Similar to the conventional Gaussian random walk, its incremental offset is zero, and its variance can span the interval $(0,\infty)$. However, the kurtosis it can attain lies between (3, 6), which exceeds the kurtosis of 3 characteristic of the Gaussian random walk. This indicates that, to some degree, our model can capture the heavy-tailed nature of volatility increments. The existence of heavy tails in volatility has been discussed in empirical research (Carnero et al., 2004). This supports the notion that our model offers at least an alternative perspective, which inherently incorporates this effect without necessitating additional random variables.

We have evaluated the Gam-Chain model across different markets and at various resolutions. Experimental results reveal that, although the VI method can only provide an approximate solution for volatility, it successfully scales a significant portion of instruments' returns to the standard normal distribution, benefiting from the heavy tail of the gamma distribution. In comparison, generating the posterior based on the log-normal chain can be challenging to use directly due to its excessively thin tail, resulting in a poor normalization effect. Although the algorithmic complexity of the VI method is not significantly different from that of MC methods, it exhibits faster runtime since it does not require sampling during state estimation, relying only on basic arithmetic operations $(+,-,\times,\div)$. Furthermore, convergence is straightforward to assess since the calculation process is entirely deterministic.

In the subsequent sections, we first review classic models and recent advancements in the field of volatility in Sect. 2. Then, in Sect. 3, we introduce the design and algorithmic process of the Chain–Gamma model. Finally, in Sect. 4, we present comparative experiments and conduct in-depth analyses. Additionally, in Appendix A, we provide proofs of some related theoretical results.

2 Related Work

2.1 Volatility Clustering

The phenomenon of volatility clustering was first discussed in Mandelbrot (1963), where it was noted that "large changes tend to be followed by large changes, of either sign, and small changes tend to be followed by small changes." Cont (2001) identifies volatility clustering as one of the "stylized facts" and posits that it is a common occurrence in securities markets across different periods and countries. Lux and Marchesi (2000) employs an artificial market simulation to replicate this phenomenon by incorporating a specified proportion of chartists and fundamentalists. Within this market framework, when the price surpasses a certain threshold, the trading activities of chartists lead to an explosion in volatility, which is gradually redirected towards stability by the fundamentalists.

Two prominent classes of models have been utilized to describe this phenomenon: the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model (Bollerslev, 1986) and the SV model (Heston, 1993). Both models describe current volatility as a function of past volatilities. If the function is ARMA and entirely deterministic (i.e., with zero noise), it is classified as GARCH. Conversely, if the function itself is derived from another random process, it is categorized as SV. The SV model encompasses a broad class of models, with some popular variants discussed in Andersen and Benzoni (2009), including discrete and continuous, linear and nonlinear, among others. In terms of estimation methods, GARCH typically employs a two-step Maximum Likelihood Estimation (MLE) approach, where the residuals of returns are first estimated, followed by the estimation of ARMA coefficients for volatility. On the other hand, SV models generally require estimation using pseudo-likelihood or Markov chain Monte Carlo (MCMC) methods (Broto & Ruiz, 2004).

The models and their variants discussed in this article (Sects. 3.2 and 3.6) represent one of the simplest forms of expressing SV by directly assuming that volatility follows a random walk (Eq. 1). The rationale for adopting this restricted form is twofold: (1) it allows this paper to concentrate the discussion on the core concept and reserve potential extensions for future work; (2) it may suffice for numerous applications where overfitting, resulting from an excessive number of parameters, is undesirable (Salvatier et al., 2016). This approach is consistent with the assertion made in Hansen and Lunde (2005).

2.2 Dynamic Bayesian Network

SV can be naturally represented using DBNs. Research in this domain can be traced back to Jacquier et al. (1994), which employs MCMC for model estimation (Wu et al., 2014). Within this context, we focus particularly on the application of variational methods to this problem. Two primary approaches emerge:

- Maintaining the original SV form, wherein variational methods are utilized to derive an approximate solution, followed by sampling. For instance, Kleppe and Skaug (2012) employs Laplace approximation (LA) to generate proposal distributions, enhancing sampling efficiency. Furthermore, it is feasible to eliminate the MC process and directly apply nested LA for approximation (Zea Bermudez et al., 2021).
- Modifying the SV form to better suit VI. A pivotal aspect here involves the gamma distribution. The variance-gamma distribution can be derived by directly using the gamma distribution to represent variance and combining it with the normal distribution (Madan & Seneta, 1990). Subsequently, Gelman (2004) proposed switching to an inverse gamma distribution (or equivalently, modeling all returns' precision (i.e., 1/sigma) as gamma) to preserve the posterior's form during Bayesian inference. Moreover, Langrené et al. (2015) utilizes the inverse gamma process to describe the variance of variance, aiming for improved option pricing results. Leveraging the conjugate relationship between inverse gamma and normal distributions, León-González (2018) samples directly from the posterior to estimate fluctuations. Both Santos (2018) and Rezende (2022) alter the observation distribution from normal to Generalized Error Distribution (GED), while still employing the gamma distribution for its precision, allowing the likelihood to be marginalized and manipulated in closed form. This approach also accommodates the representation of heavy tails within the variance.

The distinguishing feature of this paper is our direct utilization of the nonlinear gamma chain to express fluctuations. We refrain from resorting to other, more complex or indirect methodologies.

2.3 Deep Learning

Deep learning has gained considerable popularity in recent years and has found application in volatility forecasting. An early contribution in this domain is the Neural Stochastic Volatility Model (NSVM) (Luo et al., 2018). This model employs a Recurrent Neural Network (RNN)-based stochastic volatility model to automatically learn the dynamic characteristics of volatility through neural networks, thereby circumventing assumptions and constraints imposed by manual design. Shortly thereafter, the Deep Stochastic Volatility Model (DSVM) (Xu & Chen, 2021) also utilized RNN to model the generative process of volatility and incorporated a variational inference network to approximate the posterior distribution. Furthermore, Ramos-Pérez et al. (2021) adopted the Transformer model, which has achieved remarkable success in natural language processing, to model volatility, claiming to have attained superior results compared to RNN. Additionally, Chen and Robert (2022) utilized graph-structured data to represent asset correlations, capturing the intricate dependencies among assets and offering a novel perspective for multivariate volatility forecasting.

In summary, considering both the prevalence and reproducibility of the models, we have selected the following eight models for comparison in the experiments presented in Sect. 4: GARCH (Bollerslev, 1986), TGARCH (Ali, 2013), EGARCH (Ali, 2013), GJR (Ali, 2013), StoVol (Taylor, 1994), StoVolL (Taylor, 1994), GPVol (Wu et al., 2014), and DSVM (Xu & Chen, 2021).

3 Gamma Chain Models

3.1 Problem Statement

Consider a SV model of the form:

$$\begin{aligned} \boldsymbol{y}_t - \boldsymbol{y}_{t-1} &\sim N(0, \boldsymbol{u}_t^{-1}), \\ \log \boldsymbol{u}_t - \log \boldsymbol{u}_{t-1} &\sim f^*(\boldsymbol{\theta}) \end{aligned} \tag{1}$$

Among them, $\{\boldsymbol{y}_t\}$ is the observation sequence, N is the normal distribution (for the convenience of the variational derivation in Sect. 3.3, the variance here is set as the reciprocal of $\boldsymbol{u}_t$), $\{\boldsymbol{u}_t\}$ is the volatility[3] sequence (unobservable), f^* is a probability distribution function. It depends only on its parameter $\boldsymbol{\theta}$ and does not change over time. Obviously, $\{\log \boldsymbol{u}_t\}$ has the property of independent increment, and $\{\Delta \log \boldsymbol{u}_t\}$ is stationary. when $f^*(\boldsymbol{\theta}) \triangleq N(0, S^2)$, Eq. 1 is

$$\begin{aligned} \boldsymbol{y}_t - \boldsymbol{y}_{t-1} &\sim N(0, \boldsymbol{u}_t^{-1}), \\ \boldsymbol{u}_t &\sim LogN(\log \boldsymbol{u}_{t-1}, S^2) \end{aligned} \tag{2}$$

where LogN is lognormal distribution. Here $\{\log \boldsymbol{u}_t\}$ obeys a Gaussian random walk process. Note that the variance of $\{\Delta \log \boldsymbol{u}_t\}$ is S^2 and the kurtosis is 3.

The questions to be studied in this section is, (1) Can a new f^* be defined such that the kurtosis of $\{\Delta \log \boldsymbol{u}_t\}$ is greater than 3? (2) How to quickly estimate $\boldsymbol{u}_t$ for a given observation $\boldsymbol{y}_{1:T}$; 3) Estimation of parameter $\boldsymbol{\theta}$.

3.2 Model Definition

As a preliminary attempt, we tentatively use straightforwardly a gamma chain to express the change in volatility, which is

$$\boldsymbol{u}_t \sim Ga(A, \boldsymbol{u}_{t-1}) \tag{3}$$

Where $Ga(A, \boldsymbol{u}_{t-1})$ represents the gamma distribution with shape A and rate $\boldsymbol{u}_{t-1}$. To align Eq. 3 with Eq. 1, we denote $\boldsymbol{w}_t \triangleq \Delta \log(\boldsymbol{u}_t)$, then Eq. 3 is rewritten as (details in A.1)

[3] Different from above, it is actually logarithmic precision, i.e. negative logarithmic variance. This setting is only for convenience and does not affect our final result.

$$p(\boldsymbol{w}_t) = \frac{e^{-e^{\boldsymbol{w}_t}\boldsymbol{u}_{t-1}^2}\left(e^{\boldsymbol{w}_t}\boldsymbol{u}_{t-1}^2\right)^A}{\Gamma(A)} \tag{4}$$

where Γ is the gamma function. Note that the distribution parameter in Eq. 4 is $(A, \boldsymbol{u}_{t-1})$, where $\boldsymbol{u}_{t-1}$ is obviously not time-invariant, and does not meet the requirements of f^* in Eq. 1.

To fix this defect, we insert a gamma-distributed random variable $\boldsymbol{v}_t$ after each $\boldsymbol{u}_t$ (refer to Fig. 1) to construct a random process as

$$\boxed{\begin{aligned} \boldsymbol{y}_t - \boldsymbol{y}_{t-1} &\sim N(0, \boldsymbol{u}_t^{-1}), \\ \boldsymbol{u}_t &\sim Ga(A, \boldsymbol{v}_{t-1}), \\ \boldsymbol{v}_{t-1} &\sim Ga(A, \boldsymbol{u}_{t-1}) \end{aligned}} \tag{5}$$

We marginalize out $\boldsymbol{v}_{t-1}$ and align it again to the Eq. 1 (details in A.1):

$$p(\boldsymbol{w}_t) = \frac{e^{-A\boldsymbol{w}_t}(e^{\boldsymbol{w}_t}+1)^{-2A}\Gamma(2A)}{\Gamma(A)^2}. \tag{6}$$

$\boldsymbol{u}_{t-1}$ no longer exists here because it has been eliminated, and only parameter A remains. Thus we can define a qualified f^* as the right side of Eq. 6. Next, the variance and kurtosis are calculated to compare the expressivity with Eq. 2. The moment generation function of Eq. 6 is (details in A.2):

$$\phi(\lambda) = \mathbb{E}[e^{\lambda \boldsymbol{w}_t}] = \frac{\Gamma(A-\lambda)\Gamma(A+\lambda)}{\Gamma(A)^2}. \tag{7}$$

Deriving Eq. 7 and calculating the central moments from the 1st to 4th order, its variance can be obtained as

$$V = Var(\boldsymbol{w}_t) = 2\psi^{(1)}(A) \tag{8}$$

, and kurtosis as (details in A.3)

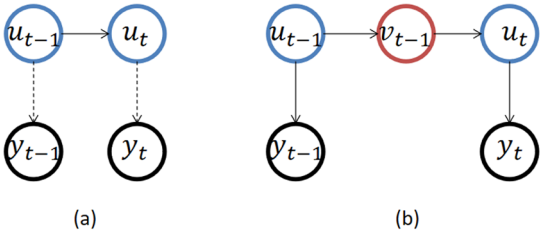


Fig. 1 Compare the lognormal distribution chain (**a**) and the gamma distribution chain (**b**). Black represents observations, blue represents fluctuations, and red represents dummy nodes. The solid line represents a closed-form local posterior, and the dashed line represents that the closed-form no longer exists after adding that node

$$K = Kurt(\boldsymbol{w}_t) = 3 + \frac{\psi^{(3)}(A)}{2\left[\psi^{(1)}(A)\right]^2} \tag{9}$$

Where ψ is the digamma function. Note that $\lim_{A\to 0} V = \infty$, $\lim_{A\to\infty} V = 0$, thus the variance can be assume to $(0, \infty)$, which is equivalent to the expressive power of Eq. 2. The value range of kurtosis is wider, such as $\lim_{A\to 0} K = 6$, $\lim_{A\to\infty} K = 3$. It can be shown that the kurtosis of Eq. 6 is exactly within the interval (3,6) (details in A.4).

3.3 State Estimation

Under the Bayesian perspective, $\boldsymbol{z} \triangleq \{\boldsymbol{u}, \boldsymbol{v}\}_{1:T}$ is the set of all state variables. State estimation is to find the posterior $p(\boldsymbol{z} \mid \boldsymbol{y})$ under the given observation $\boldsymbol{y} \triangleq \{\boldsymbol{y}_{1:T}\}$. It should be pointed out that neither the model of Eq. 2 nor the model of Eq. 5 can analytically give an exact posterior. However, for the Eq. 5, at each local $p(\boldsymbol{u}_t \mid \boldsymbol{v}_t, \boldsymbol{y}_t, \boldsymbol{v}_{t+1})$, $p(\boldsymbol{v}_t \mid \boldsymbol{u}_{t-1}, \boldsymbol{u}_t)$, yet it can give the exact posterior, and keep the form of the gamma distribution unchanged. Thus, we can use variational inference to find every local analytical solution and iteratively find a global approximate solution.

Based on variational inference, we set the optimization objective to maximize the loss function

$$\mathcal{L}(q) = \int q(\boldsymbol{z}) \ln\{\frac{p(\boldsymbol{z}, \boldsymbol{y})}{q(\boldsymbol{z})}\} \mathrm{d}\boldsymbol{z}. \tag{10}$$

where q is the posterior probability to be solved. Since finding an exact solution for the formula 10 is still difficult, we further give the mean filed assumption (Bishop, 2006):

$$q(\boldsymbol{z}) = \prod_{i=1}^{2T} q_i(\boldsymbol{z}_i) \tag{11}$$

After substituting Eq. 11 into Eq. 10, then for each q_i, the optimal solution should satisfy(Bishop, 2006)

$$q_i^*(\boldsymbol{z}_i) = \frac{\exp\{\mathbb{E}_{j\neq i}[\ln p(\boldsymbol{z}, \boldsymbol{y})]\}}{\int \exp\{\mathbb{E}_{j\neq i}[\ln p(\boldsymbol{z}, \boldsymbol{y})]\} \mathrm{d}\boldsymbol{z}_i} \tag{12}$$

. Then, substitute Eq. 1 and Eq. 5 into Eq. 12. and simplify it. We can see that $q_i^*(\boldsymbol{z}_i) \sim Ga(\boldsymbol{a}_i, \boldsymbol{b}_i)$, and their parameters are

$$\boldsymbol{a}_t^{(u)}, \boldsymbol{b}_t^{(u)} = \begin{cases} A + 3/2, & \boldsymbol{a}_{t+1}^{(v)} / \boldsymbol{b}_{t+1}^{(v)} + (\Delta \boldsymbol{y}_t)^2/2 \quad for\ t = 1 \\ 2A + 1/2, & \boldsymbol{a}_t^{(v)} / \boldsymbol{b}_t^{(v)} + \boldsymbol{a}_{t+1}^{(v)} / \boldsymbol{b}_{t+1}^{(v)} + (\Delta \boldsymbol{y}_t)^2/2 \quad for\ t > 1 \end{cases} \tag{13}$$

$$\boldsymbol{a}_t^{(v)}, \boldsymbol{b}_t^{(v)} = \begin{cases} 2A,\ \boldsymbol{a}_{t-1}^{(u)}/\boldsymbol{b}_{t-1}^{(u)} + \boldsymbol{a}_t^{(u)}/\boldsymbol{b}_t^{(u)} & for\ t < T \\ A, \qquad \boldsymbol{a}_{t-1}^{(u)}/\boldsymbol{b}_{t-1}^{(u)} & for\ t = T \end{cases} \tag{14}$$

At the beginning of the iteration, we set the initial value of all $\{\boldsymbol{a}_t, \boldsymbol{b}_t\}$ to $\{1/2, \boldsymbol{y}_t^2/2\}$ (that is, the corresponding posterior of one single observation, where the prior is $Ga(0, 0)$). After iteratively updating Eqs. 13 and 14, the desired result is obtained after convergence.

3.4 Parameter Estimation

Below, we estimate the parameter A by the EM algorithm (Bishop, 2006). In each iteration, it maximizes the following objective (M-step) based on the state estimate (E-step) in Sect. 3.3

$$\mathcal{Q}(A, A^{(\text{old})}) \triangleq \mathbb{E}\left[l(A) \mid \boldsymbol{y}, A^{(\text{old})}\right] \tag{15}$$

where $l(A) \triangleq \sum_{t=1}^{T} \log p(\boldsymbol{u}_t, \boldsymbol{v}_t, \Delta \boldsymbol{y}_t \mid A)$. Substitute Eq. 5 into Eq. 15, and get

$$\begin{aligned} \mathcal{Q}(A, A^{(\text{old})}) =& A\left\{ \sum_{t=1}^{T} 2\mathbb{E}\left[\log \boldsymbol{u}_t\right] + \mathbb{E}\left[\log \boldsymbol{v}_t\right] + \sum_{t=2}^{T} \mathbb{E}\left[\log \boldsymbol{v}_{t-1}\right] \right\} \\ & - \sum_{t=1}^{T} 2\log\Gamma(A) \\ & + \text{const.} \end{aligned} \tag{16}$$

Its gradient is

$$\begin{aligned} \nabla \mathcal{Q}(A, A^{(\text{old})}) = & \sum_{t=1}^{T} 2\mathbb{E}\left[\log \boldsymbol{u}_t\right] + \mathbb{E}\left[\log \boldsymbol{v}_t\right] + \sum_{t=2}^{T} \mathbb{E}\left[\log \boldsymbol{v}_{t-1}\right] \\ & - \sum_{t=1}^{T} 2\psi^{(0)}(A) \end{aligned} \tag{17}$$

In addition, for the posterior $z \sim Ga(a, b)$ of any hidden state, we have

$$\begin{aligned} \mathbb{E}[z] &= a/b \\ \mathbb{E}[\log z] &= \psi^{(0)}(a) - \log b \end{aligned} \tag{18}$$

Just substitute Eqs. 13 and 14 into Eq. 18 to calculate the expected expectation of Eq. 17 to get the current gradient. At the beginning of each M-step, set the initial value of A to 1, and then use the gradient ascent method to find A that maximizes Eq. 15.

3.5 Algorithm

Algorithm 1 The brief procedure of Gam-Chain.

1: **while** A has not coveraged **do**
2: ▷ E Step
3: **for** t=1:T **do**
4: update $\boldsymbol{b}_t^{(u)}$,$\boldsymbol{b}_t^{(v)}$
5: **end for**
6: **for** t=1:T **do**
7: calculate $\mathbb{E}[\boldsymbol{u}_t], \mathbb{E}[\log \boldsymbol{u}_t], \mathbb{E}[\log \boldsymbol{v}_t]$
8: **end for**
9: ▷ M Step
10: **for** t=1:T **do**
11: calc $\nabla \mathcal{Q}_t(A, A^{(\text{old})})$
12: **end for**
13: $A \leftarrow A^{(old)} + \lambda \sum_t \nabla \mathcal{Q}_t(A, A^{(\text{old})})$ ▷ gradient ascent
14: **end while**

See Algo. 1 for the program's main process. Note that the running time of Line 4 is s_{a},[4] the running time of line 7 is e, Line 11-Line 13 is g. And, the number of iterations of the loop 1–14 is L, then the running time of the algorithm is roughly $\mathcal{O}\big(L * T * (s_{\text{a}} + g + e + a)\big)$. During implementation, we should try to put the repeated calculations outside the loop as much as possible. For example, the $(\Delta \boldsymbol{y}_t)^2/2$ operation in Eq. 13 actually only needs to be calculated once. For another example, although both Line 7 and 13 contain $\psi^{(0)}(A)$, it can be extracted up to the outer loop 1–14, and it complexity will not increase with sequence length.

For each step: Since $g + a$ is usually arithmetic operations, there should be not too much impact on performance; e needs to calculate the logarithm, which is the same whether in this algorithm or Algo. 3. Thus, the key to boosting is s_{a}, and its runtime should be critical. In Sect. 4.3 we will give detailed comparisons.

3.6 Several Variants

3.6.1 Gam-Chain/MC

For the algorithm in Sect. 3.5, we name it Gam-Chain/VI. We can also use MC to estimate the model of Eq. 5. Compared with Gam-Chain/VI, it differs only in E-step, where particle smoothing is used for state estimation(Godsill et al., 2004).

[4] 'a' represents arithmetic calculation.

Algorithm 2 The MC version of state estimation of Gam-Chain.

▷ E Step
for t=1:T **do** ▷ forward sampling
  for i=1:N **do** ▷ number of particles
    $\boldsymbol{w}_{\boldsymbol{u}_t}^{(i)} \propto \boldsymbol{w}_{\boldsymbol{v}_{t-1}}^{(i)} p(\boldsymbol{y}_t \mid \boldsymbol{u}_t)$ ▷ update weights
    $\boldsymbol{w}_{\boldsymbol{v}_t}^{(i)} = \boldsymbol{w}_{\boldsymbol{u}_t}^{(i)}$
  end for
end for
for t=1:T **do** ▷ backward smoothing
  for i=1:N **do**
    $\boldsymbol{w}_{\boldsymbol{u}_t|\boldsymbol{v}_t}^{(i)} \propto \boldsymbol{w}_{\boldsymbol{u}_t}^{(i)} p(\widetilde{\boldsymbol{v}}_t \mid \boldsymbol{u}_t^{(i)})$
    choose $\widetilde{\boldsymbol{v}}_{t-1}$ with probability $\boldsymbol{w}_{\boldsymbol{u}_t|\boldsymbol{v}_t}^{(i)}$
    $\boldsymbol{w}_{\boldsymbol{v}_t|\boldsymbol{u}_{t+1}}^{(i)} \propto \boldsymbol{w}_{\boldsymbol{v}_t}^{(i)} p(\widetilde{\boldsymbol{u}}_{t+1} \mid \boldsymbol{v}_t^{(i)})$
    choose $\widetilde{\boldsymbol{u}}_t$ with probability $\boldsymbol{w}_{\boldsymbol{v}_t|\boldsymbol{u}_{t+1}}^{(i)}$
  end for
end for
for t=1:T **do** ▷ estimate of expectation
    $\mathbb{E}[\boldsymbol{u}_t] \approx \sum_{i=1}^{N} \boldsymbol{w}_t^{(i)} \boldsymbol{u}_t^{(i)}$
  $\mathbb{E}[\log \boldsymbol{u}_t] \approx \sum_{i=1}^{N} \boldsymbol{w}_t^{(i)} \log \boldsymbol{u}_t^{(i)}$
  $\mathbb{E}[\log \boldsymbol{v}_t] \approx \sum_{i=1}^{N} \boldsymbol{w}_t^{(i)} \log \boldsymbol{v}_t^{(i)}$
end for

The complexity of this Algo. 2 is $\mathcal{O}\big(L * T * (4 * N * s_{\mathrm{g}} + g + e + a)\big)$.[5] Compared to Algo. 1, it differs at $2 * N * s_{\mathrm{g}}$ and s_{a}. In order to compare more fairly, when comparing performance at Sect. 4.3, for any algorithm that requires MC, we set the number of particles to the minimum value, i.e., 2; when comparing accuracy at Sect. 4.2.3, we set the number of particles to a large enough value, i.e., 20.

3.6.2 LogN-Chain/MC

Similar to Gam-Chain/MC, we can also define the MC version of Eq. 2 as shown in Algo. 3.

[5] 'g' represents the calculation of the gamma function.

Algorithm 3 The MC version of LogN-Chain's EM process.

1: ▷ E Step
2: **for** t=1:T **do** ▷ forward sampling
3: **for** i=1:N **do**
4: $\boldsymbol{w}_{\boldsymbol{u}_t}^{(i)} \propto \boldsymbol{w}_{\boldsymbol{u}_{t-1}}^{(i)} p(\boldsymbol{y}_t \mid \boldsymbol{u}_t)$ ▷ update weights
5: **end for**
6: **end for** ▷ backward smoothing
7: **for** t=1:T **do**
8: **for** i=1:N **do**
9: $\boldsymbol{w}_{\boldsymbol{u}_t|\boldsymbol{u}_{t+1}}^{(i)} \propto \boldsymbol{w}_{\boldsymbol{u}_t}^{(i)} p(\widetilde{\boldsymbol{u}}_{t+1} \mid \boldsymbol{u}_t^{(i)})$
choose $\widetilde{\boldsymbol{u}}_t$ with probability $\boldsymbol{w}_{\boldsymbol{u}_t|\boldsymbol{u}_{t+1}}^{(i)}$
10: **end for**
11: **end for**
▷ estimate of expectation
12: **for** t=1:T **do**
$\mathbb{E}[\log^2 \boldsymbol{u}_t] \approx \sum_{i=1}^{N} \boldsymbol{w}_t^{(i)} \log^2 \boldsymbol{u}_t^{(i)}$
13: $\mathbb{E}[\log \boldsymbol{u}_t \log \boldsymbol{u}_{t-1}] \approx \sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{w}_t^{(i)} \boldsymbol{w}_{t-1}^{(j)} \log \boldsymbol{u}_t^{(i)} \log \boldsymbol{u}_{t-1}^{(j)}$
14: **end for**
15: ▷ M Step
16: $S^2 \leftarrow 1/T \left(\sum_{i=2}^{T} \mathbb{E}[\log^2 \boldsymbol{u}_t] - 2\mathbb{E}[\log \boldsymbol{u}_t \log \boldsymbol{u}_{t-1}] + \mathbb{E}[\log^2 \boldsymbol{u}_{t-1}] \right)$

Its complexity is the same as Algo. 2, the difference is that only 'exp' needs to be calculated in Line 2 instead of Γ. However, note that $\Gamma(A)$ is the same for every EM iteration and it can be extracted outside the loop 10–12 in Algo. 1, so there should be no substantial difference in performance between them.

3.6.3 LogN-Chain/VI

Although the posterior of Eq. 2 has no closed form, it is still possible to approximate it with LA (Kleppe & Skaug, 2012).

Algorithm 4 The VI version of LogN-Chain's EM process, where W at line 3 is Lambert W function.

1: ▷ E Step
2: **for** t=1:T **do**
$\boldsymbol{\mu}_t \leftarrow \frac{S^2}{4} + \frac{\boldsymbol{\mu}_{t-1}}{2} + \frac{\boldsymbol{\mu}_{t+1}}{2} - W\left(\frac{1}{4}S^2\boldsymbol{y}_t^2 e^{\frac{S^2}{4}+\frac{\boldsymbol{\mu}_{t-1}}{2}+\frac{\boldsymbol{\mu}_{t+1}}{2}}\right)$
3:
$\boldsymbol{\sigma}_t^2 \leftarrow \frac{2}{e^{\boldsymbol{\mu}_t}\boldsymbol{y}_t^2 + 4/S^2}$
4: **end for** ▷ estimate expectations
5: **for** t=1:T **do**
$\mathbb{E}[\log^2 \boldsymbol{u}_t] = \boldsymbol{\mu}_t^2 + \boldsymbol{\sigma}_t^2$
6:
$\mathbb{E}[\log \boldsymbol{u}_t] = \boldsymbol{\mu}_t$
7: **end for**
8: ▷ M Step
9: $S^2 \leftarrow 1/T\left(\sum_{i=2}^{T} \boldsymbol{\mu}_t^2 + \boldsymbol{\sigma}_t^2 - 2\boldsymbol{\mu}_t\boldsymbol{\mu}_{t-1} + \boldsymbol{\mu}_{t-1}^2 + \boldsymbol{\sigma}_{t-1}^2\right)$

Specifically in the E-step of Algo. 4, we use $LogN(\boldsymbol{\mu}_{\boldsymbol{u}_t}, \boldsymbol{\sigma}^2_{\boldsymbol{u}_t})$ distribution to approximate $p(\boldsymbol{u}_t|\boldsymbol{y}_{1:T})$. And, based on the assumption of mean-field, there is $\mathbb{E}[\log \boldsymbol{u}_t \log \boldsymbol{u}_{t-1}] = \mathbb{E}[\log \boldsymbol{u}_t]\mathbb{E}[\log \boldsymbol{u}_{t-1}]$. Therefore, the calculation of the M-step can be simplified to the expression at Line 9 in Algo. 4. Overall, Algo. 4 has the same algorithmic complexity as Algo. 1, but is much slower than arithmetic operations due to the use of Lambert W functions (refer to Sect. 4.3 for details)).

For the convenience of discussion, when the above four algorithms are mentioned later, they are sorted and named as C1[6] (LogN-Chain/VI), C2 (LogN-Chain/MC), C3 (Gam-Chain/VI)), C4 (Gam-Chain/MC). We need to pay special attention to **C3** as this is the primary method recommended in this paper.

4 Experiments

4.1 Data

For our datasets, we selected cryptocurrency data,[7] Nasdaq data,[8] and Forex[9] market data. Among these, cryptocurrency markets exhibit the most extreme volatility, which is advantageous for testing the model's state estimation capabilities. Additionally, similar to Forex markets, cryptocurrency markets operate 24 h a day, facilitating comparisons across different time resolutions. The Nasdaq market, with its high trading volume and diverse stock offerings, also lacks intra-day limits. This aligns

[6] i.e., the 1st combination.
[7] obtained from Binance exchange.
[8] retrieved using pandas_datareader.
[9] obtained from www.myfxbook.com, quoted in USD.

with Eq. 1, where the support of historical returns should span $(-\infty, \infty)$. In terms of the time span, we directly selected thousands of periods preceding the experiment day. At the minute and hour levels, the cryptocurrency data include extreme fluctuations such as the LUNA crash, while at the day level, the Nasdaq data encompass the COVID-19 market crash, both of which are representative events.

During data preprocessing, we transformed all raw closing prices into log-returns, whose properties are presented in Table 1. Data points corresponding to periods with no transactions, i.e., where volume equals zero, were removed. These empty points are devoid of business significance. If they were not eliminated, the returns would not conform to a continuous distribution but rather a mixture of zero and non-zero values, which would contradict the fundamental assumption outlined in Eq. 1.

From Table 1, it is evident that the Nasdaq and Forex markets exhibit lower volatility compared to cryptocurrencies. This is understandable given that the tokens traded in the cryptocurrency market are neither stocks, which are guaranteed by future dividends, nor are they legal tender, endorsed by national credit. As a loosely defined "proof of stake," their value is often highly uncertain. Furthermore, while volatilities are not directly observable, we can approximate them on a point-by-point basis, as shown in the last columns of Table 1. It is apparent that the so-called "kurtosis of variance," which refers to a measure of the tailedness of the distribution, should be present in both crypto and stock markets (with values greater than 3), and it appears to be even more pronounced in low-frequency data.

4.2 Result of State Estimation

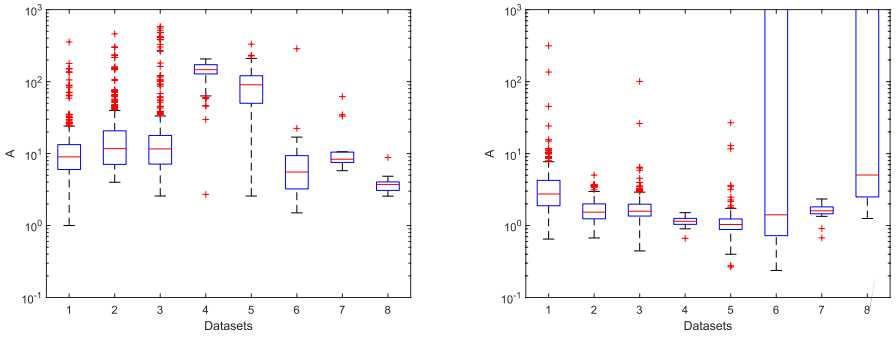
4.2.1 Distribution of Parameters

This section will examine the distribution of parameters estimated by **C3** over different datasets. As the only parameter in the model, 'A' uniquely determines the following values: the kurtosis of returns γ_r, the variance of volatilities' increments σ_v^2,

Table 1 Summary of Datasets

ID	Market	Start T	End T	#Ins	Freq	Len	$\sigma_r(\times 10^{-3})$	γ_r	σ_v	γ_v
D1	crypto	22-05-01	22-05-31	345	1 m	28463	4.146	82.16	1.893	4.646
D2	crypto	21-01-01	22-05-31	381	1 h	9350	20.79	91.48	2.851	3.743
D3	crypto	17-08-17	22-05-31	406	1d	573.1	107.6	29.77	3.069	3.893
D4	nasdaq	17-01-03	22-05-31	102	1d	1283.7	59.06	116.9	3.173	4.058
D5	nasdaq	17-01-03	22-05-31	300	1d	734.5	72.58	87.55	2.777	5.011
D6	forex	22-05-20	22-05-31	27	1 m	15840	0.294	23.41	3.123	2.937
D7	forex	22-01-01	22-05-31	27	1 h	3600	1.687	14.31	3.248	4.331
D8	forex	20-01-01	22-05-31	27	1d	881	7.409	3.857	3.154	3.783

The 'Len.' in the header is the average length of the sequence (with no-transaction blanks removed). Define $\boldsymbol{r} = \Delta \log(\boldsymbol{price})$, $\boldsymbol{v} = \Delta \log(\boldsymbol{r}^2)$. σ_r represents the standard deviation of $\boldsymbol{r}$, γ_r represents the kurtosis of $\boldsymbol{r}$, σ_v represents the standard deviation of $\boldsymbol{v}$, γ_v represents the kurtosis of $\boldsymbol{v}$. The dataset D4 contains Nasdaq 100 Index constituents. The dataset D5 is a subset of Nasdaq by sorting tradable Nasdaq tickers alphabetically and picking the top 300



(a) γ_r (b) A

Fig. 2 distribtions of γ_r vs. distribtions of A

and the kurtosis of volatilities' increments γ_v. Regarding the relationship between A and the latter two, it can be seen from Sect. A.3 that both σ_v^2 and γ_v are decreasing functions of A.

Next, we will study the relationship between γ_r and A. Consider integrating out $\boldsymbol{u}_t$ in Eq. 5, and let $\boldsymbol{v}_{t-1} \equiv B$ (i.e., remove the autocorrelation in volatilities), then get

$$p(\boldsymbol{y}_t \mid B) = \frac{2^A B^A \left(2B + \boldsymbol{y}_t^2\right)^{-A-\frac{1}{2}} \Gamma\left(A + \frac{1}{2}\right)}{\sqrt{\pi}\Gamma(A)} \tag{19}$$

In fact, this is a non-standardized Student's t-distribution[10] Its kurtosis is:

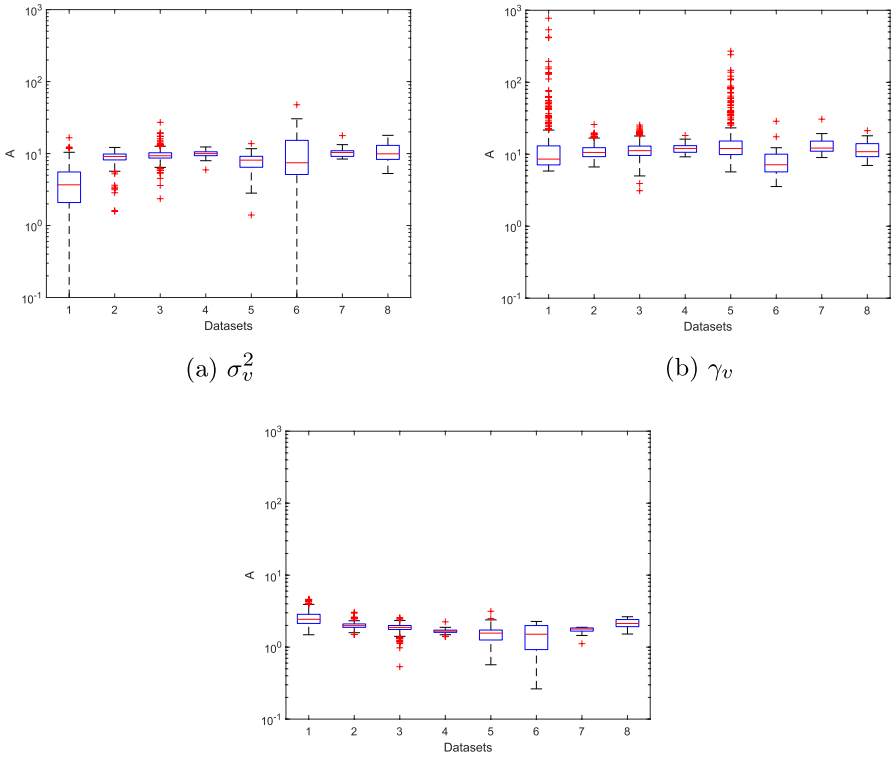
$$\frac{3\Gamma(A-2)\Gamma(A)}{\Gamma(A-1)^2} \text{ if } A > 2 \tag{20}$$

This formula is also a decreasing function of A. To verify this, we separately trained the model of Eq. 19 and compared its As with γ_r in Table 1, as shown in Fig. 2.

In general, the larger the average γ_r value of the data set, the smaller the corresponding A, and the inverse proportional relationship between them is generally validated.

Further, we run **C3** on all datasets and get the empirical distribution of A as shown in Fig. 3(c). By comparing with Fig. 2b, we find that the autocorrelation between volatilities does not strongly impact the model estimation. Compared to Fig. 3c with Fig. 3a, b, we cannot find a very clear correlation yet. However, possibly due to the introduction of two layers of noise, the model of Eq. 6 estimates $\boldsymbol{u}_t$ more smoothly, which makes it difficult to estimate A too large or too small. If it is too large, $\boldsymbol{u}_t$ will be nearly equal; if it is too small, it will cancel the volatility

[10] Compound probability distribution, https://en.wikipedia.org/wiki/Compound_probability_distribution.



(a) σ_v^2 (b) γ_v

(c) A

Fig. 3 distribtions of σ_v^2, γ_v vs. distribtions of A

aggregation. These situations are both difficult to occur in practice; thus, the Eq. 6 gives a more concentrated range of estimates than the Eq. 19.

4.2.2 Comparison of Accuracy

As mentioned in Sect. 2.3, we conducted comparative tests using GARCH, TGARCH, EGARCH, GJR, StoVol (Taylor, 1994), StoVolL (Taylor, 1994),[11] GPVol (Wu et al., 2014),[12] and DSVM,[13] along with the four variants mentioned in Sect. 3.6. The model parameters were set to the most commonly used or default settings, for example, GARCH with orders (1, 1), and the neural network settings were kept consistent with those in Xu and Chen (2021). For each forward prediction of 100 steps, the model was re-estimated within a window (uniformly set to a length of 1000). We used Negative Log-Likelihood (NLL) to evaluate the predictive

[11] https://cran.r-project.org/web/packages/stochvol.
[12] https://bitbucket.org/jmh233/code-gpvol-nips2014/src/master.
[13] https://pytorch.org.

Table 2 Comparison of predictive capabilities among multiple models

	D1	D2	D3	D4	D5	D6	D7	D8
GARCH	−4.759	−2.522	−1.54	−1.715	−1.619	−6.733	−5.011	−3.567
TGARCH	−4.767	−2.485	−1.704	−1.532	−1.476	−7.07	−5.083	−3.59
EGARCH	−4.761	−2.522	−1.543	−1.722	−1.673	−7.502	−5.048	−3.62
GJR	−4.762	−2.522	−1.542	−1.722	−1.633	−6.733	−4.959	−3.593
GPVol	−4.363	−2.316	−1.034	−1.493	−1.461	−7.683	−4.838	−3.449
StoVol	**− 5.906**	−2.194	−2.9	0.735	0.719	−6.677	−6.354	−3.594
StoVolL	−5.872	−2.159	**−2.903**	0.886	0.836	−6.89	**−6.378**	−3.616
DSVM	−4.227	−2.568	−0.954	−2.236	−2.194	−5.545	−4.451	−3.451
C1	−4.908	−2.754	−1.671	−2.728	−2.647	−7.737	−5.339	−3.702
C2	−4.923	−2.777	−1.689	−2.784	−2.696	−7.643	−5.338	−3.649
C3	−4.778	−2.529	−1.578	−1.72	−1.626	−6.727	−5.051	−3.584
C4	−5.01	**−2.914**	−1.813	**−2.896**	**−2.805**	**−7.862**	−5.443	**−3.885**

The optimal one in each dataset is marked in bold

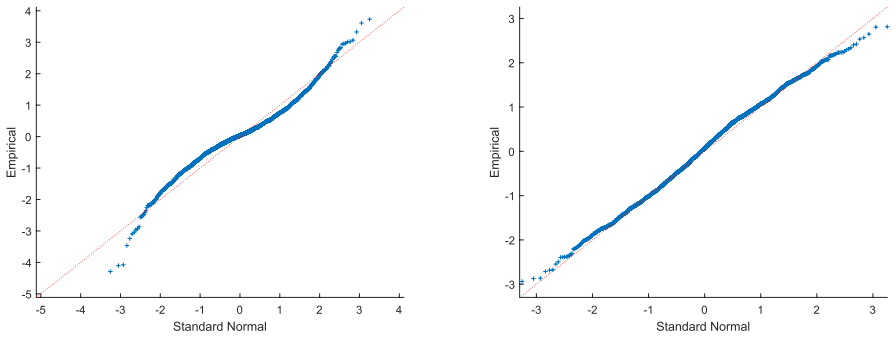
performance of the models, with smaller values indicating better model predictions. The results are presented in Table 2.

We observe significant performance variations among the models, with no single model consistently outperforming all others across all scenarios. Relatively, the most competitive models are StoVol(L) and C4. The GARCH-type and deep learning-type models did not exhibit the expected level of performance. Notably, although **C3** did not surpass C4-which is understandable given that variational methods only provide approximate calculations-its performance was quite stable. It did not produce particularly poor results on D4 and D5, as seen with StoVol(L) (which might be attributed to convergence difficulties).

The possible reasons behind these phenomena could be as follows. Cryptocurrency markets (D1, D2, D3) typically exhibit high volatility and non-normality, which may favor models capable of capturing volatility clustering and fat-tailed characteristics, such as StoVol(L). Although the NASDAQ market (D4, D5) also displays some volatility, differences in market structure and trading mechanisms may render the C4 model more advantageous in such markets. Foreign exchange markets (D6, D7, D8) are relatively stable but may still possess specific volatility patterns, leading to comparable performances of models like C4 and StoVol(L). Furthermore, high-frequency data (e.g., 1-minute data for D1, D2, D6, and 1-hour data for D7) may contain more noise and short-term volatility information, imposing higher requirements on the robustness and processing capabilities of the models. Overall, **C3** and C4 performed well. Low-frequency data (e.g., daily data for D3, D8) may encompass more complex intrinsic patterns (beyond simple random walks), resulting in a more stable performance of StoVol(L).

Table 3 Percentage of residuals for each instrument which can pass the KS-test

	Δy_t	C1	C2	**C3**	C4
D1	0	0.1246	**0.8260**	0.6869	0.7913
D2	0	0.1312	0.9632	0.9160	**0.9685**
D3	0.0073	0.4778	0.9679	0.9088	**0.9852**
D4	0	0.0196	0.8333	0.7745	**0.8627**
D5	0	0.1652	**0.9449**	0.8601	0.8986
D6	0.2352	0.8823	**1**	0.9411	**1**
D7	0	0	**0.9411**	0.8823	0.8823
D8	0	0.7647	0.8823	0.8235	**0.9411**



(a) $C1$ (b) $C3$

Fig. 4 QQ-plots of residuals generated by C1 and **C3**. The sequence is Tesla's day-level data, selected from dataset D4

4.2.3 Residual Test

The state estimation results of C1-C4 are compared below. If $\boldsymbol{u}_t$ is observable, such as $\boldsymbol{u}_t^*$, then all residuals $\left\{\boldsymbol{e}_t \triangleq \boldsymbol{y}_t/\boldsymbol{u}_t^*\right\}$ will be exactly the standard normal distribution, which is so-called 'normalization'. However, since $\boldsymbol{u}_t$ is unobservable, we only have the posterior $p(\boldsymbol{u}_t \mid \boldsymbol{y}_{1:T})$, thus a workaround is, respectively sampling from $p(\boldsymbol{u}_t \mid \boldsymbol{y}_{1:T})$ to obtain $\boldsymbol{u}_t^s$, and then generating a residual set $\left\{\boldsymbol{e}_t^s\right\}$. The more accurate $p(\boldsymbol{u}_t \mid \boldsymbol{y}_{1:T})$ has estimated, the higher the probability $\left\{\boldsymbol{e}_t^s\right\}$ will pass the standard normal distribution test.Here we use the Kolmogorov-Smirnov test, which, as a nonparametric method, compares the difference between the empirical cumulative distribution and the $N(0, 1)$'s cumulative distribution. This approach is intuitive and mimics the manual inspection we do in the Q-Q plot.

We run the four algorithms on all datasets and check whether the residuals of each sequence are standard Gaussian, shown in Table 3. It can be seen that the MC-based algorithm still gets the best results, and whether gamma or lognormal is used, the results are very similar. However, as far as VIs are concerned, there are distinct differences. Based on the VI of Gam-chain, we get results that are 5–10% worse than

Table 4 The elapsed time of the function being used

10^{-9}s	$+/-$	$\times/\div$	e^x	$\log(x)$	x^y	$\Gamma(x)$	$\psi(x)$	$W(x)$
Time	3.926	7.875	26.65	29.04	67.75	109.9	561.1	426.3

The test method is to generate 10^9 random numbers from $LogN(0, 1)$ and take their average running time

MC, yet basically acceptable in practice. The simple use of LA, due to the thin tail of the Gaussian distribution, is always worse than Gam-Chain.

We can also make an intuitive comparison for the effect of C1 and **C3** under the VI method. Figure 4 gives a Q-Q plot of the 'normalized' residuals for a specific sequence under both methods. By comparing the quantiles with the standard normal distribution, it can be seen that C1 underestimates the fluctuation in the tail and overestimates the fluctuation around the mean; in contrast, the difference between **C3** and the standard normal distribution is much smaller.

4.3 Performance Comparison

Next, we will compare the performance of C1–C4 under different sequence lengths.[14] If we change the specific instrument in testing, the impact on performance is minimal, so we must choose a long enough sequence to test.

4.3.1 Running Time of Used Functions

As we described in Sect. 3.6, the key to performance is the functions used in the E-step. We tested the running time of the functions used in four algorithms, as shown in Table 4.[15]

From the table, it can be inferred that, because of the different functions required, according to the discussion in the 3.6 section, the operation time order of the E-step should be: **C3**<C2<C4<C1. In M-step, the order of operation time is C1<C2<**C3**<C4. However, the extra time of **C3** in the M step caused by the $\psi(x)$ function is limited, for the number of calculations of $\psi(x)$ is fixed to 1 in each iteration. Therefore, it can be expected that when the sequence length T increases, the consumption of **C3** on the $\psi(x)$ will be covered by the advantages obtained by the E-step, and it will run faster than any other method.

[14] The environment configuration is as follows. CPU: Intel64 Family 6 Model 142 Stepping 9 GenuineIntel 2803 Mhz; Memory: 16,223 MB; OS: Win10; Compiler: MSVC 14.16; Additional Dependencies: boost 1.79.0.

[15] This is implemented in C++. It does not require a virtual machine like Java or Python. It can directly use pointers (addresses) to read array elements, which is very efficient and removes the overhead of address translation. This allows us to have a more precise assessment of algorithm performance.

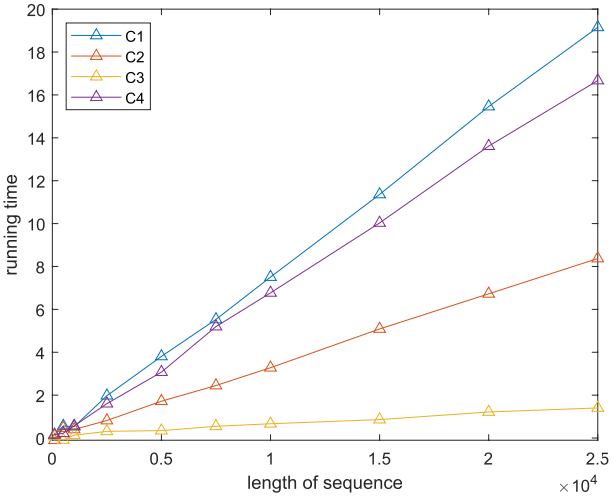


Fig. 5 Comparison of time consumed. Here BTC/USDT is selected from D1 as the test sequence. It is truncated when a different sequence length is required. For the MC method, we only set the number of particles to 2

4.3.2 Actual Measurement

The time consumption of each step is tested below under different sequence lengths. For fairness, we have fixed the number of iterations to 1000. (It has been observed that there is no significant difference in the number of iterations for C1–C4. Moreover, the number of iterations for a longer sequence may not necessarily be more. Therefore, it is feasible to set a fixed value.) The test results are shown in the Fig. 5.

The results show that the time of the E-step almost dominates the increase of the running time as the sequence length grows. To be precise, **C3** only has a growth rate of about 20% of the commonly used C2. This verifies the effectiveness of the scheme in this paper. In fact, the number of particles in C2 will not be set to 2 in practice, but to a larger number, such as 10. In this way, the speedup of **C3** will be multiplied, such as $20\%/5 = 4\%$. Moreover, its calculation process is deterministic, unlike the MC method, which requires additional iterations to determine whether or not it has converged.

5 Conclusion

This paper introduces an alternative approach for estimating stochastic volatility based on the variational method. This method enables rapid volatility estimation for a large number of time series, and the entire calculation process is deterministic, facilitating easy convergence assessment. This characteristic is particularly valuable for high-frequency trading applications. In fact, the VI and MC methods are independent, and our approach can be utilized to perform fast initialization for any MC method. When compared to the LA, which also belongs to the VI-class methods, the

approximate posterior tail obtained by LA is heavier and does not require the computation of the Lambert W function. Consequently, our method can enhance both accuracy and performance.

In future research, we plan to explore the incorporation of a more complex gamma network capable of capturing the autocorrelation effect between volatilities. Alternatively, we may employ multiple layers of latent gamma variables to achieve a wider range of kurtosis representation. Furthermore, there is additional scope for optimizing this scheme, such as utilizing Taylor expansion to expedite the calculation of the digamma function, which would further enhance its performance.

Appendix: Details of Derivation

Derivation of the Distribution of Volatility Increments

For the density function of the random variable function $Y = h(X)$, the general formula is

$$g(y) = f\left(h^{-1}(y)\right)\left|\left(h^{-1}(y)\right)'\right| \tag{21}$$

where $f(\cdot)$ is the density function of the random variable X.

For 3, its $f(\cdot)$ is a gamma distribution. Since $\boldsymbol{w}_t \triangleq \Delta \log(\boldsymbol{u}_t)$, then $\boldsymbol{u}_t = \boldsymbol{u}_{t-1}e^{\boldsymbol{w}_t}$. Substitute it into 21 to get

$$p(\boldsymbol{w}_t) = \frac{\boldsymbol{u}_{t-1}^{A} e^{\boldsymbol{u}_{t-1}^{2}(-e^{\boldsymbol{w}_t})}\left(\boldsymbol{u}_{t-1}e^{\boldsymbol{w}_t}\right)^{A-1}}{\Gamma(A)}\left|\boldsymbol{u}_{t-1}e^{\boldsymbol{w}_t}\right|, \tag{22}$$

which is 4 after simplification.

For 5, first integrate out $\boldsymbol{v}_{t-1}$, the steps are as follows

$$\begin{aligned} p(\boldsymbol{u}_t|\boldsymbol{u}_{t-1}) &= \int_0^{\infty} p(\mathbf{u}_t|\mathbf{v}_{t-1})p(\mathbf{v}_{t-1}|\boldsymbol{u}_{t-1})\mathrm{d}\mathbf{v}_{t-1} \\ &= \int_0^{\infty} \frac{\boldsymbol{u}_{t-1}^{A}\boldsymbol{u}_t^{A-1}\mathbf{v}_{t-1}^{2A-1}e^{-\boldsymbol{u}_{t-1}\mathbf{v}_{t-1}-\boldsymbol{u}_t\mathbf{v}_{t-1}}}{\Gamma(A)^2}\mathrm{d}\mathbf{v}_{t-1} \\ &= \frac{\boldsymbol{u}_{t-1}^{A}\boldsymbol{u}_t^{A-1}(\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)^{-2A}}{\Gamma(A)^2} \\ &\cdot \int_0^{\infty} e^{-(\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)\mathbf{v}_{t-1}}\left((\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)\mathbf{v}_{t-1}\right)^{2A-1}(\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)\mathrm{d}\mathbf{v}_{t-1} \\ &= \frac{\boldsymbol{u}_{t-1}^{A}\boldsymbol{u}_t^{A-1}(\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)^{-2A}}{\Gamma(A)^2}\Gamma(2A) \end{aligned} \tag{23}$$

Similar to 22, we substitute $\boldsymbol{u}_t = \boldsymbol{u}_{t-1}e^{\boldsymbol{w}_t}$ and 23 into 21, and get

$$p(\boldsymbol{w}_t) = \frac{e^{(A-1)\boldsymbol{w}_t}\left(e^{\boldsymbol{w}_t}+1\right)^{-2A}\Gamma(2A)}{\cancel{\boldsymbol{u}_{t-1}}\Gamma(A)^2}|\cancel{\boldsymbol{u}_{t-1}}e^{\boldsymbol{w}_t}|, \tag{24}$$

which is 6 after simplification. It can be seen from the above formula that $\boldsymbol{u}_{t-1}$ has been eliminated.

Derivation of Moment Generating Function

In the model 5, the moment generating function corresponding to $\boldsymbol{w}_t$ is

$$\begin{aligned}
\phi(\lambda) &= \int_0^\infty \int_0^\infty e^{\lambda \log \frac{\boldsymbol{u}_t}{\boldsymbol{u}_{t-1}}} p(\boldsymbol{u}_t|\boldsymbol{v}_{t-1}) p(\boldsymbol{v}_{t-1}|\boldsymbol{u}_{t-1}) \mathrm{d}\boldsymbol{v}_{t-1}\mathrm{d}\boldsymbol{u}_t \\
&= \int_0^\infty \int_0^\infty \frac{\boldsymbol{v}_{t-1}^{2A-1}\boldsymbol{u}_{t-1}^{A-\lambda}\boldsymbol{u}_t^{A+\lambda-1}e^{-\boldsymbol{v}_{t-1}(\boldsymbol{u}_{t-1}+\boldsymbol{u}_t)}}{\Gamma(A)^2}\mathrm{d}\boldsymbol{v}_{t-1}\mathrm{d}\boldsymbol{u}_t \\
&= \frac{1}{\Gamma(A)^2}\int_0^\infty \left(\int_0^\infty (\boldsymbol{u}_t\boldsymbol{v}_{t-1})^{A+\lambda-1}e^{-\boldsymbol{u}_t\boldsymbol{v}_{t-1}}\boldsymbol{v}_{t-1}\mathrm{d}\boldsymbol{u}_t\right) \\
&\cdot (\boldsymbol{v}_{t-1}\boldsymbol{u}_{t-1})^{A-\lambda-1}e^{-\boldsymbol{v}_{t-1}\boldsymbol{u}_{t-1}}\boldsymbol{u}_{t-1}\mathrm{d}\boldsymbol{v}_{t-1} \\
&= \frac{1}{\Gamma(A)^2}\int_0^\infty \Gamma(A+\lambda)\cdot(\boldsymbol{v}_{t-1}\boldsymbol{u}_{t-1})^{A-\lambda-1}e^{-\boldsymbol{v}_{t-1}\boldsymbol{u}_{t-1}}\boldsymbol{u}_{t-1}\mathrm{d}\boldsymbol{v}_{t-1} \\
&= \frac{\Gamma(A+\lambda)\Gamma(A-\lambda)}{\Gamma(A)^2}
\end{aligned} \tag{25}$$

This is 7.

Derivation of Kurtosis

After taking the derivative of 7, the 1st to 4th moments of $\boldsymbol{w}_t$ are obtained as:

$$\begin{aligned}
\mathbb{E}[\boldsymbol{w}_t] &= \phi'(0) = 0, \\
\mathbb{E}[\boldsymbol{w}_t^2] &= \phi''(0) = 2\psi^{(1)}(A), \\
\mathbb{E}[\boldsymbol{w}_t^3] &= \phi'''(0) = 0, \\
\mathbb{E}[\boldsymbol{w}_t^4] &= \phi''''(0) = 2\left(6\psi^{(1)}(A)^2 + \psi^{(3)}(A)\right)
\end{aligned} \tag{26}$$

Therefore, its variance and kurtosis are

$$\begin{aligned}
Var(\boldsymbol{w}_t) &= \mathbb{E}[\boldsymbol{w}_t^2] - \mathbb{E}[\boldsymbol{w}_t]^2 \\
&= 2\psi^{(1)}(A) \\
K(\boldsymbol{w}_t) &= \frac{\mathbb{E}[\boldsymbol{w}_t]^4 - 4\mathbb{E}[\boldsymbol{w}_t]^3\mathbb{E}[\boldsymbol{w}_t] + 6\mathbb{E}[\boldsymbol{w}_t]^2\mathbb{E}[\boldsymbol{w}_t^2] - 4\mathbb{E}[\boldsymbol{w}_t]\mathbb{E}[\boldsymbol{w}_t^3] + \mathbb{E}[\boldsymbol{w}_t^4]}{Var(\boldsymbol{w}_t)^2} \\
&= 3 + \frac{\psi^{(3)}(A)}{2\psi^{(1)}(A)^2}
\end{aligned} \tag{27}$$

Proof of Kurtosis Bound

By the formula 6.4.10 on page 260 in Abramovitz and Stegun (1968), we know

$$\psi^{(n)}(A) = (-1)^{n+1} n! \sum_{k=0}^{\infty} \frac{1}{(A+k)^{n+1}}. \tag{28}$$

Then

$$K(w_t) = 3 + \frac{\psi^{(3)}(A)}{2[\psi^{(1)}(A)]^2} = 3 + 3\frac{\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}}{[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2}$$

Since $[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2 = \sum_{k=0}^{\infty} \frac{1}{(A+k)^4} + \sum_{k\neq l} \frac{1}{(A+k)^2(A+l)^2} \geq \sum_{k=0}^{\infty} \frac{1}{(A+k)^4}$, then it is easy to see that $\frac{\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}}{[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2} \leq 1$. So $3 \leq K(w_t) \leq 6$. Moreover, $\frac{\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}}{[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2}$ is a continuous function of A on $(0, +\infty)$. When $A \to 0$, we see $\sum_{k=0}^{\infty} \frac{1}{(A+k)^4} \sim \frac{1}{A^4}$ and $\sum_{k=0}^{\infty} \frac{1}{(A+k)^2} \sim \frac{1}{A^2}$. Therefore,

$$\lim_{A\to 0} \frac{\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}}{[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2} = 1.$$

When $A \to \infty$, we should approximate series $\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}$ and $\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}$ by improper integrals. Because $\frac{1}{(k+1+A)^2} \leq \int_k^{k+1} \frac{dx}{(x+A)^2} \leq \frac{1}{(k+A)^2}$ and $\frac{1}{(k+1+A)^4} \leq \int_k^{k+1} \frac{dx}{(x+A)^4} \leq \frac{1}{(k+A)^4}$, then we know

$$\int_0^{\infty} \frac{dx}{(A+x)^4} \leq \sum_{k=0}^{\infty} \frac{1}{(A+k)^4} \leq \frac{1}{A^d} + \int_0^{\infty} \frac{dx}{(A+x)^4} \tag{29}$$

$$\int_0^{\infty} \frac{dx}{(A+x)^2} \leq \sum_{k=0}^{\infty} \frac{1}{(A+k)^2} \leq \frac{1}{A^d} + \int_0^{\infty} \frac{dx}{(A+x)^2} \tag{30}$$

Then we know $\sum_{k=0}^{\infty} \frac{1}{(A+k)^4} \sim \int_0^{\infty} \frac{dx}{(A+x)^4} = \frac{1}{3A^3}$ and $\sum_{k=0}^{\infty} \frac{1}{(A+k)^2} \sim \int_0^{\infty} \frac{dx}{(A+x)^2} = \frac{1}{A}$ when $A \to \infty$. Thus,

$$\lim_{A\to\infty} \frac{\sum_{k=0}^{\infty} \frac{1}{(A+k)^4}}{[\sum_{k=0}^{\infty} \frac{1}{(A+k)^2}]^2} = \lim_{A\to\infty} \frac{\frac{1}{3A^3}}{\frac{1}{A^2}} = 0.$$

Now we know $K(w_t) = 3 + \frac{\psi^{(3)}(A)}{2[\psi^{(1)}(A)]^2}$ is a continuous function of A. Both $K(w_t) = 3$ and $K(w_t) = 6$ are their horizontal asymptotic lines. Then $K(w_t)$ can assume all values in (3, 6).

Funding The authors have not disclosed any funding.

Declarations

Conflict of interest The authors have not disclosed any conflict of interests.

References

Abramovitz, M., & Stegun, I. (1968). *Mathematical Functions with Formulas*. New York: Graphs and Mathematical Tables. Dover.

Ali, G., et al. (2013). Egarch, gjr-garch, tgarch, avgarch, ngarch, igarch and aparch models for pathogens at marine recreational sites. *Journal of Statistical and Econometric Methods, 2*(3), 57–73.

Andersen, T. G., & Benzoni, L. (2009). Stochastic volatility. In Encyclopedia of Complexity and Systems Science.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307–327.

Broto, C., & Ruiz, E. (2004). Estimation methods for stochastic volatility models: A survey. *Derivatives eJournal*.

Carnero, M. A., Peña, D., & Ruiz, E. (2004). Persistence and kurtosis in garch and stochastic volatility models. *Journal of Financial Econometrics, 2*(2), 319–342.

Chen, Q., & Robert, C. -Y. (2022). Multivariate realized volatility forecasting with graph neural network. In *Proceedings of the Third Acm International Conference on Ai in Finance,* pp. 156–164.

Cont, R. (2001). Empirical properties of asset returns: Stylized facts and statistical issues. *Quantitative Finance, 1*, 223–236.

Gelman, A. (2004). Prior distributions for variance parameters in hierarchical models (comment on article by browne and draper). *Bayesian Analysis, 1*, 515–534.

Godsill, S. J., Doucet, A., & West, M. (2004). Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association, 99*(465), 156–168.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a garch (1, 1)? *Journal of Applied Econometrics, 20*(7), 873–889.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies, 6*(2), 327–343.

Jacquier, E., Polson, N. G., & Rossi, P. E. (1994). Bayesian analysis of stochastic volatility models. *Journal of Business & Economic Statistics, 20*, 69–87.

Kleppe, T. S., & Skaug, H. J. (2012). Fitting general stochastic volatility models using Laplace accelerated sequential importance sampling. *Computational Statistics & Data Analysis, 56*, 3105–3119.

Langrené, N., Lee, G., & Zhu, Z. (2015). *Switching to non-affine stochastic volatility: A closed-form expansion for the inverse gamma model*. Econometrics: Econometric & Statistical Methods - Special Topics eJournal.

León-González, R. (2018). Efficient bayesian inference in generalized inverse gamma processes for stochastic volatility. *Econometric Reviews, 38*, 899–920.

Luo, R., Zhang, W., Xu, X., & Wang, J. (2018). A neural stochastic volatility model. In *Proceedings of the AAAI Conference on Artificial Intelligence 32*.

Lux, T., & Marchesi, M. (2000). Volatility clustering in financial markets: A microsimulation of interacting agents. *International Journal of Theoretical and Applied Finance, 3*(04), 675–702.

Madan, D. B., & Seneta, E. (1990). The variance gamma (v.g.) model for share market returns. *The Journal of Business, 63*, 511–524.

Mandelbrot, B. (1963). The variation of certain speculative prices. *The Journal of Business, 36*(4), 394–419.

Ramos-Pérez, E., Alonso-González, P. J., & Núñez-Velázquez, J. J. (2021). Multi-transformer: A new neural network-based architecture for forecasting s &p volatility. *Mathematics, 9*(15), 1794.

Rezende, T. (2022). A new filtering inference procedure for a ged state-space volatility model. *Journal of Statistical Planning and Inference*.

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in python using pymc3. *PeerJ Computer Science, 2*, 55.
Santos, T. R. D. (2018). *A bayesian ged-gamma stochastic volatility model for return data: A marginal likelihood approach*. arXiv: Statistical Finance.
Taylor, S. J. (1994). Modeling stochastic volatility: A review and comparative study. *Mathematical Finance, 4*(2), 183–204.
Wu, Y., Hernández-Lobato, J. M., & Ghahramani, Z. (2014). Gaussian process volatility model. *Advances in Neural Information Processing Systems*, *27*.
Xu, X., & Chen, Y. (2021). *Deep stochastic volatility model.* arXiv preprint arXiv:2102.12658
Zea Bermudez, P., Marín, J. M., Rue, H., & Veiga, H. (2021). Integrated nested laplace approximations for threshold stochastic volatility models. *Econometrics and Statistics*.

Authors and Affiliations

Di Zhang[1] · Youzhou Zhou[2]

✉ Di Zhang
di.zhang@xjtlu.edu.cn

Youzhou Zhou
youzhou.zhou@xjtlu.edu.cn

[1] School of AI and Advanced Computing, Xi'an Jiaotong-Liverpool University, No. 111 Taicang Avenue, Suzhou 215400, Jiangsu, People's Republic of China

[2] School of Mathematics and Physics, Xi'an Jiaotong-Liverpool University, No. 111 Ren'ai Road, Suzhou 215123, Jiangsu, People's Republic of China