A Differentiable Particle Filter for Jump-Diffusion Stochastic Volatility Models

by

Michelle Ko

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Stochastic volatility with jumps has emerged as a crucial tool for understanding and modelling the stochastic and intermittently discontinuous nature of many processes in finance. Due to the highly nonlinear structure of these models, their likelihood functions are often unavailable in closed-form. A common numerical approach is to reformulate the original model as a state-space model, where under this framework, the marginal likelihood of the parameters can be estimated efficiently by integrating the latent variables via particle filtering. A combination of such particle-estimated likelihood and Markov Chain Monte Carlo can be used to sample from parameter posteriors, but imposes a substantial computational burden in multi-dimensional parameter space. Bayesian normal approximation serves as a more efficient alternative, if the mode and quadrature of the stochastic approximation of the posterior can be obtained via a gradient-based method. This is not immediately possible, however, as the particle-estimated marginal posterior is not differentiable due to (1) the inherent discontinuity of jumps in the model, and (2) the widely used multinomial resampling technique in particle filtering. This thesis presents a novel construction of a particle filter that incorporates a multivariate normal resampler and circumvents the jump-induced discontinuity with a customized proposal density, thereby attaining full differentiability of the marginal posterior estimate. A comprehensive simulation study and application to S&P 500 Index data are provided to investigate the performance of the differentiable particle filter for parameter inference and volatility recovery.

Acknowledgements

First and foremost, I would like to thank Dr. Martin Lysy for introducing me to the captivating world of computational inference through STAT 440 in Winter 2020. Since then, I have had the privilege to explore various applications of stochastic differential equations under his supervision, ranging from biochemistry to quantitative finance. This thesis would not have been a possibility without his unwavering support throughout my undergraduate and graduate studies combined.

I would also like to express my gratitude to the committee members, Dr. Samuel Wong and Dr. Tony Wirjanto, for their precious time and effort to revise this thesis.

My sincere thanks go to Dr. Shoja'eddin Chenouri for believing in my potential and granting admission to the program. In addition, I would also like to thank the administrative staff of the Department of Statistics and Actuarial Science. In particular, I am grateful for Mary Lou Dufton's guidance on the logistical aspect of completing my degree and for her insights into balancing between academics and other facets of life.

I am very grateful for the teachings of Dr. Adam Kolkiewicz, Dr. Tony Wirjanto, Dr. Yuying Li, and Dr. Fan Yang on subjects relating to quantitative finance, which contributed to shaping the context of this thesis.

I extend my heartfelt gratitude to Omnium—Johnny, Colin, and Adam—for continuing to offer me valuable employment opportunities, allowing me to support myself financially and grow as a professional throughout my degree. It was a great pleasure to have a highly skilled and hardworking individual like Arvind as my first intern.

I wholeheartedly appreciate both my longstanding friends and those I have met along the way. Thanks to Jonathan, Max, and Zac—for their talent and passion—my first term of graduate studies was one of the most memorable and inspiring periods of my academic career. Connecting with Kyumin, Qing, Chris, and Sohoon, my fellow academically motivated MMath friends, has made this journey less lonely. I cannot imagine how my time in Waterloo for the past two years would have been without Alan, Hawon, Kate, and Mike. Leanne, my best friend of over a decade, holds my deepest appreciation for her kindness, companionship, and everything else that I could ask for in a friend.

Most importantly, I am forever indebted to my family. No words can possibly express the magnitude of the love, support, and patience they provided throughout my life.

Dedication

This is dedicated to the younger selves of me and my mother, for our enduring belief in the better things in life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Jump-diffusion models have emerged as a crucial tool in understanding the stochastic and intermittently discontinuous nature of various financial processes. Stochastic volatility with jumps (SVJ) is an important variant of jump-diffusion, which offers a sensible depiction of asset price dynamics that captures many phenomena in real-world financial markets. Some of the empirical characteristics captured by this class of models include (1) abrupt movements in price, (2) leptokurtic return distribution, and (3) auto-correlation in squared returns, also known as volatility clustering (Cont & Tankov, 2004).

Despite a wealth of literature addressing the forward problem—numerical methods for simulating process trajectories and pricing options within the jump-diffusion framework (Broadie & Kaya, 2006; Casella & Roberts, 2011; Glasserman & Merener, 2003; Metwally & Atiya, 2002)—the inverse problem, or inferring model parameters from observed data, remains notoriously challenging. As nonlinear models like SVJ rarely admit a closed-form solution, their likelihood functions are analytically intractable, thus rendering standard maximum likelihood methods for parameter estimation unusable.

A common approach to tackle these challenges involves data augmentation, whereby highly nonlinear and continuous-time models are approximately discretized and recast into state-space models consisting of parameters, latent variables, and observations. Under the state-space framework, the latent state-space variables can be efficiently integrated out by particle filtering (Doucet & Johansen, 2011; Murray, 2015). A combination of particle filter and Markov Chain Monte Carlo (MCMC), namely particle MCMC, has been the method of

choice for sampling from the parameter posterior densities (Andrieu, Doucet, & Holenstein, 2010). However, without an informative parameter proposal, the computational cost of particle MCMC grows substantially with the dimensions of the parameter space.

Recent studies have revealed a promising avenue for differentiable particle filters coupled with gradient-based techniques (Corenflos et al., 2021; Jonschkowski, Rastogi, & Brock, 2018). For the purpose of parameter inference for jump-diffusion models, the interest lies in incorporating particle filters to gradient-based stochastic optimization over the model parameter space, which can offer a fast inference method via Bayesian normal approximation of the posterior densities (Gelman et al., 1995). This requires the particle-estimated marginal likelihood to be differentiable with respect to the model parameters. However, due to the usual resampling algorithm and the discrete random variables in the jump-diffusion model, particle filters do not immediately yield a differentiable stochastic approximation of the marginal likelihood.

1.2 Contribution and Thesis Outline

This thesis presents a fully differentiable particle filter for jump-diffusion models, particularly tailored to a highly complex stochastic volatility with contemporaneous jump (SVCJ) model. The said particle filter is not only differentiable in its resampling step, but also through the non-differentiable compound Poisson process that governs the jump occurrences in the model. A simulation study demonstrates the effectiveness of this approach, producing reasonable parameter estimates for models with a high number of latent variables requiring integration. The results are achieved in orders of few minutes by leveraging JAX, a high-performance computing library in Python. The performance of differentiable particle filter for parameter inference was further validated using S&P 500 Index data, where the recovered volatility closely matched SPOTVOL, CBOE's spot volatility index.

Chapter 2 provides a detailed description of jump-diffusion models and SVCJ models, along with their full state-space representation. The standard particle filter and a bridge proposal for state-space models with error-free observations are also outlined. Chapter 3 begins by identifying the sources of non-differentiability, both in the model and the particle filter. Suitable treatments for each of the discontinuities are proposed in detail. As well, the gradient-based optimization framework for parameter inference via the mode-quadrature method is justified. Chapter 4 presents a simulation study, in which the performance of the differentiable particle filter in an inference problem is assessed with a synthetic dataset. The same test is conducted on a real-world S&P 500 dataset, with comparison between

the particle-recovered volatility and SPOTVOL. Chapter 5 summarizes the findings of this thesis and discusses its limitations, which may evolve into further research directions.

Chapter 2

Background

2.1 Jump-Diffusion Models

Jump-diffusion models are a class of stochastic processes that incorporate both continuous diffusion processes and discontinuous jumps, introduced by Merton (1976) to capture the stochastic and intermittently discontinuous behaviour of asset price movement. A typical formulation of jump-diffusion models is given by combining a stochastic differential equation (SDE) model with a compound Poisson jump process, where jumps are considered rare events and have finite occurrences in any finite intervals. Then, a one-dimensional jump-diffusion process $S(t)$ satisfies

$$\mathrm{d}S(t) = \mu_{\boldsymbol{\theta}}(S(t), t)\mathrm{d}t + \sigma_{\boldsymbol{\theta}}(S(t), t)\mathrm{d}B(t) + J(t)\mathrm{d}N_{\lambda}(t), \tag{2.1}$$

where $\mu_{\boldsymbol{\theta}}$ and $\sigma_{\boldsymbol{\theta}}$ are drift and diffusion functions governed by a set of unknown parameters $\boldsymbol{\theta}$, respectively, $B(t)$ is a Brownian motion process, $N_{\lambda}(t)$ is a Poisson process with intensity λ, and $J(t)$ is a random variable that follows a specific jump size distribution $\pi_J(\cdot)$. Some well-known examples that adhere to this structure are the Merton jump-diffusion model (Merton, 1976) and the Kou model (Kou, 2002), whereby a Black-Scholes-type (Black & Scholes, 1973) geometric Brownian motion (GBM) is coupled with log-normal and double-exponential jump size distributions, respectively.

The Merton jump-diffusion model is one of the only few exceptional cases where a closed-form solution and a tractable likelihood are available. Under different specifications of the SDE and/or the jump component, the dynamics of the processes become highly

complex and nonlinear, posing a critical obstacle for analytical approaches to simulation or inference of these models.

2.1.1 Stochastic Volatility with Contemporaneous Jump

Stochastic volatility models are an extension of Black-Scholes diffusion that addresses the heteroskedastic nature of asset returns. These models consider volatility as a stochastic process, unlike the constant value assumption in GBM or the treatment as a deterministic function in local volatility models. The Heston model (Heston, 1993) is one prominent example of such extension, in which the volatility follows a stochastic square-root process.

Stochastic volatility with jump—a combination of stochastic volatility and jump-diffusion models—were promptly investigated, and early works by Bates (1996) introduced incorporating log-normal jumps to the price process of the Heston model. After Bates (2000), Duffie, Pan, and Singleton (2000), and Pan (2002) identified the need for a jump component in the volatility process as well, one of the specifications reviewed in Eraker, Johannes, and Polson (2003) is the stochastic volatility with contemporaneous jump (SVCJ) model. This paper examines a generalized version of the SVCJ model with the following system of SDEs for the price-volatility pair $(S(t), V(t))$:

$$\begin{aligned} \mathrm{d}S(t) &= \mu_{\boldsymbol{\theta}}(S(t),t)\mathrm{d}t + \sigma_{\boldsymbol{\theta}}(V(t),S(t),t)\mathrm{d}B^S(t) + J^S(t)\mathrm{d}N_\lambda(t) \\ \mathrm{d}V(t) &= \alpha_{\boldsymbol{\theta}}(V(t),t)dt + \beta_{\boldsymbol{\theta}}(V(t),t)\mathrm{d}B^V(t) + J^V(t)\mathrm{d}N_\lambda(t) \\ \mathrm{Corr}(B_S(t), B_V(t)) &= \rho. \end{aligned} \tag{2.2}$$

Notice that the volatility process $V(t)$ satisfies an SDE with jumps—occurring simultaneously with price but varying in jump sizes.

As with jump-diffusion models in Section 2.1, arbitrary assignments of the drift/diffusion functions or jump distributions in SVCJ models usually lead to analytical intractability. However, modelling volatility as a distinct stochastic process presents an additional caveat to the inference problem. As volatility is not an observable quantity, it must be inferred from observed asset price data along with unknown model parameters.

2.2 State-Space Model

State-space model refer to a graphical characterization of the dependence structure between two layers of random variables: the (latent) state and the measurement models. In discrete

and homogeneous time, when the state $\{\mathbf{X}_k\}$ is Markovian and observations $\{\mathbf{Y}_k\}$ are independently distributed when conditioned on the state process for $k = 0, 1, ..., N$, the resulting model can be succinctly described in three probability statements:

$$\begin{aligned} \mathbf{X}_0 &\sim \pi(\mathbf{X}_0 \mid \boldsymbol{\theta}) \\ \mathbf{X}_k &\sim f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta}) \\ \mathbf{Y}_k &\sim g(\mathbf{Y}_k \mid \mathbf{X}_k, \boldsymbol{\theta}), \end{aligned} \tag{2.3}$$

where $f(\cdot)$ and $g(\cdot)$ in Equation (2.3) are referred to as the transition density and the measurement density, respectively. Figure 2.1 illustrates the evolution of the latent process and the observations.

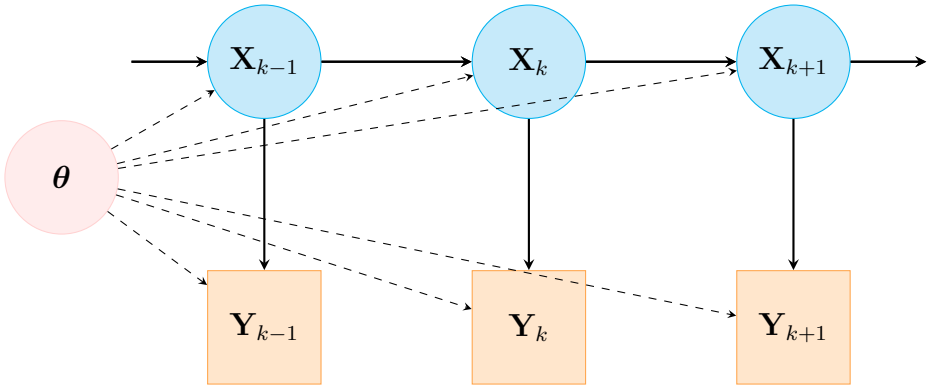

Figure 2.1: State-space model with latent process $\{\mathbf{X}_k\}$ and observations $\{\mathbf{Y}_k\}$.

The joint likelihood of the state and observation variables conditioned on $\boldsymbol{\theta}$ is

$$\boldsymbol{\mathcal{L}}(\mathbf{X}_{0:N}, \mathbf{Y}_{0:N} \mid \boldsymbol{\theta}) = \pi(\mathbf{X}_0 \mid \boldsymbol{\theta}) \prod_{k=1}^{N} f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta}) \prod_{k=0}^{N} g(\mathbf{Y}_k \mid \mathbf{X}_k, \boldsymbol{\theta}). \tag{2.4}$$

In Bayesian inference, a suitable choice of prior $\pi_{\boldsymbol{\theta}}$ leads to the joint posterior distribution on parameters and latent states

$$p(\mathbf{X}_{0:N}, \boldsymbol{\theta} \mid \mathbf{Y}_{0:N}) \propto \pi(\boldsymbol{\theta}) \boldsymbol{\mathcal{L}}(\mathbf{X}_{0:N}, \mathbf{Y}_{0:N} \mid \boldsymbol{\theta}). \tag{2.5}$$

The quantity of interest in this paper is the marginal posterior, obtained by integrating the posterior over $\mathbf{X}_{0:N}$:

$$p(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N}) = \int p(\mathbf{X}_{0:N}, \boldsymbol{\theta} \mid \mathbf{Y}_{0:N}) \mathrm{d}\mathbf{X}_{0:N}. \tag{2.6}$$

It is generally not possible to obtain the marginal posterior in closed-form for nonlinear or non-Gaussian state-space models (Doucet & Johansen, 2011). Along with the task of integrating over $\mathbf{X}_{0:N}$, it becomes necessary to take a numerical approach to estimating this quantity.

2.2.1 Discretization of Jump-Diffusion Models

To cast a continuous-time jump-diffusion model into a state-space model, the model must be first discretized. This can be done conveniently with the Euler-Maruyama scheme (Maruyama, 1955), where the Equation (2.1) transforms into

$$S(t+\Delta t) - S(t) = \mu_{\boldsymbol{\theta}}(S(t),t)\Delta t + \sigma_{\boldsymbol{\theta}}(S(t),t)\Delta B(t) + J(t+\Delta t)Q(t+\Delta t),$$

where $\Delta B_k \sim N(0,\Delta t)$ is a Brownian increment, and $Q(t+\Delta t) \sim Bern(\lambda\Delta t)$ is a Bernoulli random variable with jump probability $\lambda\Delta t \ll 1$ (Golightly, 2009; Pedersen, 1995). For data observed at equally-spaced intervals, we have

$$S_{k+1} - S_k = \mu_{\boldsymbol{\theta}}(S_k,t_k)\Delta t + \sigma_{\boldsymbol{\theta}}(S_k,t_k)\Delta B_k + J_{k+1}Q_{k+1}, \tag{2.7}$$

where $t_k = k\Delta t$ and $S_k = S(t_k)$.

Further discretization may be needed if Δt is not sufficiently small. That is, each interval Δt can be partitioned into M inter-observation times such that $\Delta^* t = \Delta t/M$ and $S_k^{(m)} = S\left(t_k^{(m)}\right)$ with $t_k^{(m)} = k\Delta t + m\Delta^* t$. Note that $t_k^{(0)} = t_k$ and $t_k^{(M)} = t_{k+1}^{(0)}$, thus $S_k^{(M)} = S_{k+1}^{(0)} = S_{k+1}$. Then, Equation (2.7) can be rewritten as

$$S_k^{(m+1)} = S_k^{(m)} + \mu_{\boldsymbol{\theta}}\left(S_k^{(m)},t_k^{(m)}\right)\Delta^* t + \sigma_{\boldsymbol{\theta}}\left(S_k^{(m)},t_k^{(m)}\right)\Delta^* B_k^{(m)} + J_k^{(m+1)}Q_k^{(m+1)} \tag{2.8}$$

for all $m = 0,...,M-1$ and $k = 0,...,N-1$, where $\Delta^* B_k \sim N(0,\Delta^* t)$ and $Q_k^{(m+1)} \sim Bern(\lambda\Delta^* t)$. This can be directly translated as a sampling procedure for the transition density in Equation (2.3).

Discretization of the SVCJ model follows suit, where Equation (2.2) transforms into

$$\begin{aligned}
S_k^{(m+1)} &= S_k^{(m)} + \mu_{\boldsymbol{\theta}}\left(S_k^{(m)},t_k^{(m)}\right)\Delta^* t + \sigma_{\boldsymbol{\theta}}\left(V_k^{(m)},S_k^{(m)},t_k^{(m)}\right)\Delta^* B_k^{S,(m)} + J_k^{S,(m+1)}Q_k^{(m+1)} \\
V_k^{(m+1)} &= V_k^{(m)} + \alpha_{\boldsymbol{\theta}}\left(V_k^{(m)},t_k^{(m)}\right)\Delta^* t + \beta_{\boldsymbol{\theta}}\left(V_k^{(m)},t_k^{(m)}\right)\Delta^* B_k^{V,(m)} + J_k^{V,(m+1)}Q_k^{(m+1)} \\
&\text{Corr}(\Delta^* B_k^{S,(m)},\Delta^* B_k^{V,(m)}) = \rho.
\end{aligned} \tag{2.9}$$

2.2.2 Error-Free Observation Model

Casting a jump-diffusion model in Section 2.1 into a state-space model requires more thoughtful consideration, as observations do not have random noise, i.e. $Y_k = S(t_k)$. In the context of state-space models, error-free observations imply a measurement density given by a Dirac measure $\delta_{\mathbf{X}_k}$, as illustrated in Figure 2.2. That is, the error-free observation model admits the measurement density

$$\mathbf{Y}_k \sim \delta_{\mathbf{X}_k} : \mathbf{Y}_k = \mathbf{X}_k \text{ with probability 1.}$$

Applying this to the discretized SVCJ model in Equation (2.9), the full state-space SVCJ model is given by $\mathbf{X}_k = (S_k^{(1:M)}, V_k^{(1:M)}, Q_k^{(1:M)}, J_k^{S,(1:M)}, J_k^{V,(1:M)})$ and $\mathbf{Y}_k = Y_k$ (one-dimensional observation).

Initial Density:

$S_0^{(0)} \sim \delta_{S_0}, V_0^{(0)} \sim \pi(V_0 \mid \boldsymbol{\theta})$

For $k = 0, \ldots, N-1$

Transition Density :

For $m = 0, \ldots, M-1$

$$Q_k^{(m+1)} \sim Bern(\lambda \Delta^* t)$$

$$J_k^{S,(m+1)} \sim \pi(J^S \mid Q_k^{(m+1)}, \boldsymbol{\theta})$$

$$J_k^{V,(m+1)} \sim \pi(J^V \mid Q_k^{(m+1)}, \boldsymbol{\theta})$$

$$\begin{bmatrix} S_k^{(m+1)} \\ V_k^{(m+1)} \end{bmatrix} \sim N\left(\begin{bmatrix} S_k^{(m)} + \mu_{\boldsymbol{\theta}}\left(S_k^{(m)}, t_k^{(m)}\right)\Delta^* t + J_k^{S,(m+1)} Q_k^{(m+1)} \\ V_k^{(m)} + \alpha_{\boldsymbol{\theta}}\left(V_k^{(m)}, t_k^{(m)}\right)\Delta^* t + J_k^{V,(m+1)} Q_k^{(m+1)} \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} \sigma_{\boldsymbol{\theta}}^2\left(S_k^{(m)}, t_k^{(m)}\right) & \rho\sigma_{\boldsymbol{\theta}}\left(S_k^{(m)}, t_k^{(m)}\right)\beta_{\boldsymbol{\theta}}\left(V_k^{(m)}, t_k^{(m)}\right) \\ \rho\sigma_{\boldsymbol{\theta}}\left(S_k^{(m)}, t_k^{(m)}\right)\beta_{\boldsymbol{\theta}}\left(V_k^{(m)}, t_k^{(m)}\right) & \beta_{\boldsymbol{\theta}}^2\left(V_k^{(m)}, t_k^{(m)}\right) \end{bmatrix} \right)$$

Measurement Density :

$$Y_{k+1} \sim \delta_{S_{k+1}}.$$

(2.10)

While error-free observations are not problematic when purely solving the forward problem, the marginal posterior in Equation (2.6) is inextricably conditioned on some

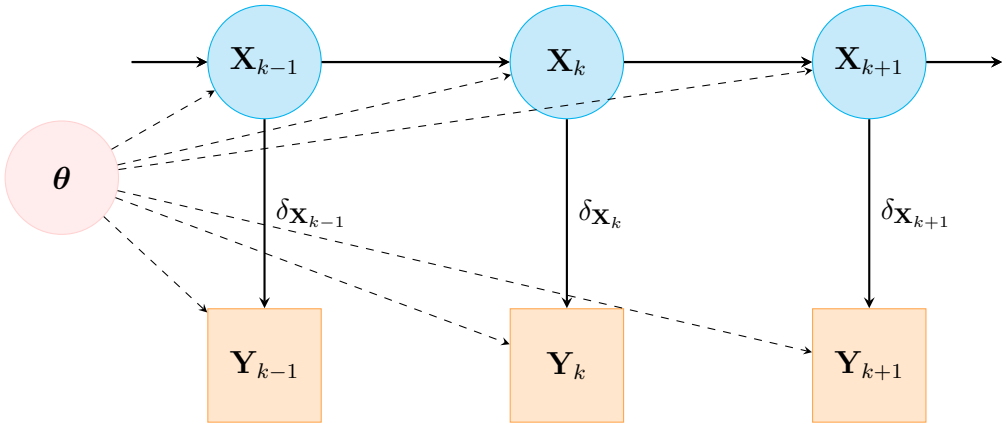

Figure 2.2: State-space model in 2.1 with error-free observations.

predetermined observed data. This implies that, for any sample trajectory $\tilde{\mathbf{X}}_{0:N}$ where $\tilde{\mathbf{X}}_k$ is not identical to $\mathbf{Y}_k$ at any time $k = 0, ..., N$, the joint likelihood reduces to zero due to

$$g(\mathbf{Y}_k \mid \tilde{\mathbf{X}}_k, \boldsymbol{\theta}) = \delta_{\tilde{\mathbf{X}}_k}(\mathbf{Y}_k) = 0 \text{ if } \tilde{\mathbf{X}}_k \neq \mathbf{Y}_k. \tag{2.11}$$

The consequences of this degeneracy to estimating the marginal posterior are to be discussed in subsequent sections.

2.3 Particle Filters

Particle filtering is a variant of importance sampling that allows to (1) sample trajectories from the latent process and (2) numerically approximate the marginal posterior in Equation (2.6) for any nonlinear state-space model (Del Moral, 1997; Murray, 2015). This is achieved efficiently by exploiting the Markovian structure of state-space models, thereby sampling trajectories, or particles, from the posterior distribution in Equation (2.5) sequentially (Doucet & Johansen, 2011).

The procedure of the filter at each observation time involves propagating, reweighing and resampling particles using the transition and measurement densities of the state-space model in Equation (2.3), accompanied with a proposal distribution $q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \mathbf{Y}_k, \boldsymbol{\theta})$. The filter is initialized with P particles $\{\tilde{\mathbf{X}}_0^i, i = 1, \ldots, P\}$ sampled independently from the prior $\pi(\mathbf{X}_0 \mid \boldsymbol{\theta})$, where each particle is assigned weight $w_0^i = \pi(\tilde{\mathbf{X}}_0^i \mid \boldsymbol{\theta})/q_0(\tilde{\mathbf{X}}_0^i \mid \mathbf{Y}_0, \boldsymbol{\theta})$. Then, the particle filter proceeds follows the steps below recursively through time:

- **Propagate:** Each particle is propagated to the next time step by sampling from the proposal distribution: $\tilde{\mathbf{X}}_k^i \sim q(\mathbf{X}_k \mid \tilde{\mathbf{X}}_{k-1}^i, \mathbf{Y}_k, \boldsymbol{\theta})$.
- **Reweigh:** The weight of each of the particles is updated with an incremental weight

$$\omega_k^i = \frac{g(\mathbf{Y}_k \mid \tilde{\mathbf{X}}_k^i, \boldsymbol{\theta}) f(\tilde{\mathbf{X}}_k^i \mid \tilde{\mathbf{X}}_{k-1}^i, \boldsymbol{\theta})}{q(\tilde{\mathbf{X}}_k^i \mid \tilde{\mathbf{X}}_{k-1}^i, \mathbf{Y}_k, \boldsymbol{\theta})}. \tag{2.12}$$

- **Resample:** The particles are replaced with a fresh set of particles, where the selection of each new particle is determined by its weight, following a particular resampling mechanism. One common choice is the multinomial resampler, in which for each particle, the number of its copies (offspring) to be passed to the next step is determined by a multinomial distribution.

Algorithm 1 Particle Filter

for each $i = 1, \ldots, P$ **do** // Initialize
  $\tilde{\mathbf{X}}_0^i \sim \pi(\mathbf{X}_0 \mid \boldsymbol{\theta})$
  $w_0^i = \pi(\tilde{\mathbf{X}}_0^i \mid \boldsymbol{\theta}) / q_0(\tilde{\mathbf{X}}_0^i \mid \mathbf{Y}_0, \boldsymbol{\theta})$
end for
for each $k = 1, \ldots N$ **do**
  $\tilde{w}_{k-1}^i = w_{k-1}^i / \sum_{j=1}^P w_{k-1}^j$ // Normalize weights
  $\bar{\mathbf{X}}_{k-1}^{1:P} \sim \text{Resampler}(\tilde{\mathbf{X}}_{k-1}^{1:P}; \tilde{w}_{k-1}^{1:P})$ // Resample
  for each $i = 1, \ldots, P$ **do**
    $\tilde{\mathbf{X}}_k^i \sim q(\mathbf{X}_k \mid \bar{\mathbf{X}}_{k-1}^i, \mathbf{Y}_k, \boldsymbol{\theta})$ // Propagate
    $\omega_k^i = g(\mathbf{Y}_k \mid \tilde{\mathbf{X}}_k^i, \boldsymbol{\theta}) f(\tilde{\mathbf{X}}_k^i \mid \bar{\mathbf{X}}_{k-1}^i, \boldsymbol{\theta}) / q(\tilde{\mathbf{X}}_k^i \mid \bar{\mathbf{X}}_{k-1}^i, \mathbf{Y}_k, \boldsymbol{\theta})$ // Incremental weight
    $w_k^i = w_{k-1}^i \cdot \omega_k^i$ // Update weight
  end for
end for
return Marginal likelihood estimate: $\hat{p}(\mathbf{Y}_{0:N} \mid \boldsymbol{\theta}) = \pi(\boldsymbol{\theta}) \prod_{k=0}^N \left(\sum_{j=1}^P \frac{1}{P} w_k^j \right)$
return (Optional) Particles and final weights: $\tilde{\mathbf{X}}_{0:N}^{1:P}, w_N^{1:P}$

For the purpose of Bayesian inference, the marginal posterior estimate can be obtained with a particle filter by

$$\hat{p}(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N}) \propto \pi(\boldsymbol{\theta}) \hat{p}(\mathbf{Y}_{0:N} \mid \boldsymbol{\theta}). \tag{2.13}$$

If the proposal distribution $q(\cdot)$ is chosen to be the transition density, then the particle filter in Algorithm 1 reduces to the bootstrap particle filter (Gordon, Salmond, & Smith,

1993), where the incremental weight is then given by the measurement density. However, in the case of error-free observations of jump-diffusion models, the degeneracy problem described by Equation (2.11) renders the bootstrap particle filter unfit for marginal posterior estimation, as incremental weight in Equation (2.12) for every particle will be zero. To overcome this issue, Subsection 2.3.2 will introduce the idea of diffusion bridge to the construction of the proposal distribution.

2.3.1 Bridge Proposal for Jump-Diffusion Models

Diffusion bridge serves as a remedy to the error-free observation cases (Durham & Gallant, 2002), in which the the trajectory at inter-observational time is sampled from a Gaussian distribution conditioned on its previous step and and the end point. For SVCJ models, given observations $y_{0:N}$, Golightly (2009) suggested simulating a bridge proposal as outlined in Algorithm 2. Note that this SVCJ bridge proposal can reduce to that of the jump-diffusion model in Equation (2.1) by omitting the volatility process and replacing $\sigma_{\boldsymbol{\theta}}(\tilde{V}_k^{(m)}, \tilde{S}_k^{(m)}, t_k^{(m)})$ with $\sigma_{\boldsymbol{\theta}}(\tilde{S}_k^{(m)}, t_k^{(m)})$.

Algorithm 2 Bridge Proposal for SVCJ model

$\tilde{S}_0^{(0)} = y_0$ // Initialize
for each $k = 0, \ldots, N-1$ **do**
  $\tilde{Q}_k^{(1:M)} \overset{iid}{\sim} Bern(\lambda\Delta^* t)$ // Simulate jumps
  $\tilde{J}_k^{S,(1:M)} \overset{iid}{\sim} \pi_{J^S}(\boldsymbol{\theta})$ // Simulate price jump sizes
  $\tilde{J}_k^{V,(1:M)} \overset{iid}{\sim} \pi_{J^V}(\boldsymbol{\theta})$ // Simulate volatility jump sizes
  for each $m = 0, \ldots, M-1$ **do**
    $\tilde{\alpha}_k^{(m)} = \alpha_{\boldsymbol{\theta}}(\tilde{V}_k^{(m)}, t), \tilde{\beta}_k^{(m)} = \beta_{\boldsymbol{\theta}}(\tilde{V}_k^{(m)}, t)$ // Drift and diffusion for volatility
    $\tilde{V}_k^{(m+1)} \sim N(V_k^{(m)} + \tilde{\alpha}_k^{(m)}\Delta^* t + \tilde{J}_k^{V,(m+1)}\tilde{Q}^{V,(m+1)},$
      $\tilde{\beta}_k^{(m)}\Delta^* t)$ // Sample volatility
    $\tilde{S}_k^{(m+1)} \sim N(\tilde{S}_k^{(m)} + \frac{y_k - \tilde{S}_k^{(m)}}{M-m} + \tilde{J}_k^{S,(m+1)}\tilde{Q}_k^{S,(m+1)} - \frac{\sum_{i=m+1}^{M} \tilde{J}_k^{S,(i)}\tilde{Q}_k^{S,(i)}}{M-m},$
      $\frac{M-m-1}{M-m}\tilde{\sigma}_{\boldsymbol{\theta}}^2(\tilde{V}_k^{(m)}, \tilde{S}_k^{(m)}, t_k^{(m)})\Delta^* t)$ // Sample price
  end for
end for
Return $\tilde{S}_k^{(1:M)}$ for $k = 0, \ldots, N-1$

The bridge proposal in Algorithm 2 reduces to a Brownian bridge between $S_k^{(m)}$ and y_k if the model contains no jump. In the presence of jumps, the mean of the bridge is

adjusted to incorporate the jump components. Note this bridge requires full knowledge of the future jump occurrences and jump sizes within the interval, which is possible as these do not depend on neither of (S_k^m, V_k^m).

2.3.2 Particle Filters for SVCJ Models with Error-Free Observations

While Algorithm 2 describes how to sample from the diffusion bridge, hence the propagation step, it is now necessary to derive the incremental weight ω_k^i with a bridge proposal to formalize the reweighing step. To obtain the incremental weight, instead of writing down the full expression for the transition and bridge proposal densities, a simpler way is to use the fact that there is only one source of difference between the two densities: the price. In the proposal distribution,

$$\tilde{S}_k^{(m+1)} \sim N(\tilde{S}_k^{(m)} + \underbrace{\frac{y_k - \tilde{S}_k^{(m)}}{M-m} + \tilde{J}_k^{S,(m+1)}\tilde{Q}_k^{S,(m+1)} - \frac{\sum_{i=m+1}^{M} \tilde{J}_k^{S,(i)}\tilde{Q}_k^{S,(i)}}{M-m}}_{\mu_{\text{prop}}(\tilde{S}_k^{(m)}, y_k)},$$
$$\underbrace{\frac{M-m-1}{M-m}\tilde{\sigma}_{\boldsymbol{\theta}}^2(\tilde{V}_k^{(m)}, \tilde{S}_k^{(m)}, t_k^{(m)})\Delta^* t}_{\sigma^2_{\text{prop}}(\tilde{S}_k^{(m)}, y_k)}),$$

$S_k^{(m+1)}$ is conditioned on the next observation y_k, as seen in Algorithm 2. However, the transition density, being not conditioned on y_k, simply becomes

$$\tilde{S}_k^{(m+1)} \sim N(\tilde{S}_k^{(m)} + \underbrace{\mu_{\boldsymbol{\theta}}(\tilde{S}_k^{(m)}, t_k^{(m)})\Delta^* t + \tilde{J}_k^{S,(m+1)}\tilde{Q}_k^{S,(m+1)}}_{\mu_{\text{trans}}(\tilde{S}_k^{(m)})},$$
$$\underbrace{\sigma_{\boldsymbol{\theta}}^2(\tilde{V}_k^{(m)}, \tilde{S}_k^{(m)}, t_k^{(m)})\Delta^* t}_{\sigma^2_{\text{trans}}(\tilde{S}_k^{(m)})}).$$

Revisiting the expression for the incremental weight in Equation (2.12), it is easy to note that the transition and the proposal densities are positioned in opposite sides of the fraction. This indicates that the densities of all other states generated by the same law—volatility, jump, and jump sizes—will appear on both the numerator and the denominator, thus cancelling each other. The mathematical details are provided in Appendix A. With

M intervals between observations, the incremental weight becomes

$$\omega_k^i = \frac{\prod_{j=1}^{M} \phi(\tilde{S}_k^{(j)}; \tilde{S}_k^{(j-1)} + \mu_{\text{trans}}(\tilde{S}_k^{(j-1)}), \sigma^2_{\text{trans}}(\tilde{S}_k^{(j-1)}))}{\prod_{j=1}^{M-1} \phi(\tilde{S}_k^{(j)}; \tilde{S}_k^{(j-1)} + \mu_{\text{prop}}(\tilde{S}_k^{(j-1)}, y_k), \sigma^2_{\text{prop}}(\tilde{S}_k^{(j-1)}, y_k))}. \tag{2.14}$$

It is worth noting that the measurement density in this case trivially equals to one, as $\delta_{\tilde{S}_k^{(M)}}(y_k) = \delta_{y_k}(y_k) = 1$. This derivation aligns with the acceptance ratio used in a Metropolis-Hastings update outlined in Golightly (2009).

Chapter 3

Methodology

3.1 Differentiable Particle Filters

A particle filter with bridge proposal in Subsection 2.3.2 coupled with a multinomial resampling scheme may be sufficient if parameter inference is performed via MCMC. Methods that blend particle filtering with MCMC, namely particle MCMC (PMCMC), include particle marginal Metropolis-Hastings (PMMH) and particle Gibbs (PG) (Andrieu, Doucet, & Holenstein, 2010):

- **PMMH**: The acceptance ratio incorporates the particle-estimated marginal likelihood, such that $A = \frac{\hat{\pi}(\boldsymbol{\theta}^*)p(\mathbf{Y}_{0:N}|\boldsymbol{\theta}^*)\varphi(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})\hat{p}(\mathbf{Y}_{0:N}|\boldsymbol{\theta})\varphi(\boldsymbol{\theta}^*|\boldsymbol{\theta})}$, where $\varphi(\cdot)$ is the parameter proposal.
- **PG**: The sampler alternates between:
  - The parameters, sampled from a parameter proposal, and
  - The states, sampled by the particle filter conditioned on the parameters.

However, these methods are computationally intensive, especially when the proposal of candidate parameters in each MCMC step is not efficient. The inference result of PG with an adaptive learning rate is presented in Appendix E.

An alternate way to estimate the posterior density is by Bayesian normal approximation, which states that the posterior density is approximately normal under the conditions in which the maximum likelihood estimator (MLE) is asymptotically normal. That is, instead of sampling from the posterior via MCMC, finding the mode and the quadrature of

the posterior suffices to provide a normal approximation to the posterior density (Gelman et al., 1995). That is,

$$\begin{aligned} &\boldsymbol{\theta} \mid \mathbf{Y}_{0:N} \sim N(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\Sigma}}) \\ &\hat{\boldsymbol{\theta}} = \operatorname{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N}), \hat{\boldsymbol{\Sigma}} = \left[\frac{\partial^2}{\partial \boldsymbol{\theta}^2} - \log p(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N}) \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \right]^{-1}. \end{aligned} \tag{3.1}$$

While $p(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N})$ is unavailable due to the intractability of the model, the particle filter yields a stochastic approximation $\hat{p}(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N})$, as shown in Equation (2.13). A natural extension is to replace $p(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N})$ with $\hat{p}(\boldsymbol{\theta} \mid \mathbf{Y}_{0:N})$, thus finding the mode and the quadrature becomes a stochastic optimization problem. However, this approach requires that the posterior be differentiable with respect to the parameters.

This prompts the idea of differentiable particle filters, in which its marginal posterior estimate is differentiable with respect to the parameters, such that the mode and quadrature estimates are efficiently computed using gradient-based optimization (Jonschkowski, Rastogi, & Brock, 2018; Rosato et al., 2020). However, particle filters for jump-diffusion models encounters two main sources of discontinuity: the multinomial resampler and the jump components. The subsequent subsections address the challenges and how to mitigate them to build a fully differentiable particle filter for SVCJ models.

3.1.1 Multivariate Normal Resampler

Particle filter suffers from non-differentiability when the resampling step involves a discontinuous distribution. Multinomial resampler is one example, in which discontinuity arises between each particle and its ancestor. While it is possible to compute the gradient estimate by ignoring the non-differentiable terms from the multinomial resampler, this estimate leads to a discernible bias and does not propagate through time (Corenflos et al., 2021).

Multivariate normal (MVN) resampler is the simplest particle resampling scheme that offers differentiability with respect to the model parameters. At each observation time, the MVN resampler calculates the weighted mean and variance of the particles to draw a new set from the resulting MVN distribution. While extremely fast, the MVN resampler may present bias if the weighted particles exhibit multimodality.

Note that for error-free observations in SVCJ models, the MVN only needs to resample the volatility component $\tilde{V}_k^{(M)}$. This is evident from the fact that every particle is coerced to

have its price component $\tilde{S}_k^{(M)}$ land on y_M at observation time. Then, the MVN resampler takes a simple form:

$$\begin{aligned} &\mu_V = \sum_{j=1}^{M} \bar{w}_k^j \tilde{V}_k^{(M),j}, \sigma_V = \frac{1}{M-1} \sum_{j=1}^{M} \bar{w}_k^j (\tilde{V}_k^{(M),j} - \mu_V)^2 \\ &\bar{V}_k^{(M),1:P} \overset{iid}{\sim} N(\mu_V, \sigma_V^2). \end{aligned} \tag{3.2}$$

3.1.2 Discontinuity due to Jumps

For a diffusion-only state-space model, the MVN resampler would be sufficient to turn the particle filter into a differentiable one. However, the task at hand deals with jump-diffusion, where discontinuities in particle trajectories are inherent and seemingly inevitable. This causes the particles to be non-differentiable in λ, the jump propensity. To see why this is the case, we revisit the propagation mechanism. In generating the latent states, the jump occurrence variable is obtained by

$$\begin{aligned} & \tilde{Q}_k^{(m)} \sim Bern(\lambda \Delta^* t) \\ \iff \quad & \tilde{Q}_k^{(m)} = H(\lambda \Delta^* t - U_k^{(m)}), U_k^{(m)} \sim Unif(0,1) \end{aligned} \tag{3.3}$$

where $H(x)$ is a Heaviside step function such that $H(x) = 1$ if $x \geq 0$ and $H(x) = 0$ otherwise. For a fixed $U_k^{(m)} = u$,

$$(\tilde{Q}_k^{(m)} \mid \tilde{U}_k^{(m)} = u) = H(\lambda \Delta^* t - u)$$

is not differentiable with respect to λ at $\lambda = u/(\Delta^* t)$.

3.1.3 Gumbel-Softmax Smoothing

The Gumbel-Softmax distribution is a continuous approximation of the categorical distribution, where the Gumbel-Softmax samples are differentiable with respect to the probabilities (Jang, Gu, & Poole, 2016). In the instance of a Bernoulli variable, where there are only two categories, the Gumbel-Softmax approximation W reduces to

$$\begin{aligned} &Z \sim Logistic(0,1) \\ &W = \frac{1}{1 + \exp\left\{ \left(Z + \log\left(\frac{1-p}{p} \right) \right) / \tau \right\}}, \end{aligned} \tag{3.4}$$

where $p \in [0,1]$ is the Bernoulli probability and $\tau > 0$ is the smoothing parameter. Full derivation is available in Appendix B. The distribution of W converges to $Bern(p)$ as $\tau \to 0$, and to $Unif(0,1)$ as $\tau \to +\infty$.

The jump component $\tilde{Q}_k^{(m)}$ can be reparameterized with Gumbel-Softmax smoothing such that each particle is differentiable with respect to λ. However, this method inevitably introduces bias in the marginal posterior estimate, the extent of which is controlled by τ. The choice of τ becomes a nontrivial task, where the trade-off between bias and numerical stability of the gradient estimate must be carefully considered.

3.1.4 Bridge Proposal under Different Jump Propensity

While Gumbel-Softmax smoothing offers per-particle differentiability, it is not the only way to attain differentiability of the marginal posterior estimate for the SVCJ model. It is to simply use a separate jump propensity $\lambda^\star$ for the bridge proposal, such that

$$q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta}^*) = q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \{\lambda^\star\} \cup \boldsymbol{\theta} \setminus \{\lambda\}).$$

Then, since the jumps are generated under a different parameter than that in the transition density, the incremental weight is modified to

$$\omega_k^i = \frac{\prod_{j=1}^{M} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{trans}}(S_k^{(j-1)}), \sigma_{\text{trans}}(S_k^{(j-1)}))}{\prod_{j=1}^{M-1} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{prop}}(S_k^{(j-1)}, y_k), \sigma_{\text{prop}}(S_k^{(j-1)}, y_k))} \frac{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda \Delta^* t)}{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda^\star \Delta^* t)}, \quad (3.5)$$

followed by the derivation in Equation (A.1).

The incremental weight in Equation (3.5) is differentiable with respect to λ with a fixed seed. To see this, if $Q_k^{(j)} = q_k^{(j)}$ for $j = 1, \ldots, M$ using $\lambda^\star$ and a certain seed, $n(q_k) = \sum_{j=1}^{M} q_k^{(j)}$ is fixed and does not depend on λ. Then, the only component of the incremental weight that depends on λ is

$$\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda \Delta^* t) = (\lambda \Delta^* t)^{n(q_k)} (1 - \lambda \Delta^* t)^{M - n(q_k)},$$

which is evidently differentiable with respect to λ.

When Jumps are Rare Events

Selecting an adequate proposal jump propensity $\lambda^\star$ involves a trade-off in effective sample sizes, as outlined in Table 3.1. In cases where λ is suspected to be sufficiently small, a small $\lambda^\star$ will increase the variability in jump-related parameter estimates, whereas a large $\lambda^\star$ will produce unnecessary particles that jump more than once in the given interval.

Small $\lambda^\star$	Large $\lambda^\star$
Better effective sample size for intervals with no jumps Poor performance on estimating jump-related parameters	Better effective sample size for intervals with a jump Particles with $\sum_{j=1}^{M} \tilde{Q}_k^{(j)} \geq 2$ wasted when jumps are rare events

Table 3.1: Trade-off in Selecting $\lambda^\star$

If λ is suspected to be small such that the probability of two or more jumps occurring between adjacent observations is negligible, this trade-off can be mitigated by modifying the jump distribution in the bridge proposal.

Instead of sampling M Bernoulli random variables (as a result of sampling at each inter-observation time), just one can be sampled from $Q_k \sim Bern(\lambda^\star \Delta t)$, thereby assigning each particle whether it will jump or not in the given interval. If $Q_k = 0$, then $Q_k^{(j)} = 0$ for all $j = 1, \ldots, M$, whereas if $Q_k = 1$, then a random index J is sampled from $Cat(M; [1/M]^M)$ such that $Q_k^{(J)} = 1$ and $Q_k^{(j)} = 0$ for $j \neq J$. Then,

$$p^*(Q_k^{(j)} \mid \boldsymbol{\theta}^*) = \begin{cases} 1 - \lambda^\star \Delta t & \text{if } Q_k = 0 \\ \frac{1}{M} \lambda^\star \Delta t & \text{if } Q_k = 1, \end{cases} \tag{3.6}$$

where $p^*(Q_k^{(j)} \mid \boldsymbol{\theta}^*)$ replaces $\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda^\star \Delta^* t)$ in Equation (3.5).

Furthermore, if one wishes to use Equation (3.6) as an approximation to the jump distribution of the transition density, the resulting incremental weight simplifies to

$$\omega_k^{*,i} = \frac{\prod_{j=1}^{M} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{trans}}(S_k^{(j-1)}), \sigma_{\text{trans}}(S_k^{(j-1)}))}{\prod_{j=1}^{M-1} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{prop}}(S_k^{(j-1)}, y_k), \sigma_{\text{prop}}(S_k^{(j-1)}, y_k))} R(Q_k; \lambda, \lambda^\star), \tag{3.7}$$

where

$$R(Q_k; \lambda, \lambda^\star) = \begin{cases} \frac{1-\lambda \Delta t}{1-\lambda^\star \Delta t} & \text{if } Q_k = 0 \\ \frac{\lambda}{\lambda^\star} & \text{if } Q_k = 1. \end{cases}$$

Note that the λ from this approximation is interpreted as the propensity of one jump happening in Δt under Bernoulli assumption. Then, the jump propensity under Poisson assumption can be recovered by

$$\frac{\lambda\Delta t}{1-\lambda\Delta t} = \frac{Pr_{Poi}(\text{Jump})}{Pr_{Poi}(\text{No Jump})} = \lambda_{Poi}\Delta t \Longrightarrow \lambda_{Poi} = \frac{\lambda}{1-\lambda\Delta t}.$$

3.2 Parameter Inference

3.2.1 Variable Transformation

A preliminary step before employing an unconstrained optimizer is to transform the parameter space in which the marginal posterior estimate is well-defined everywhere. This is necessary since the optimization algorithm fails when the updated parameter yields an undefined posterior estimate. Table 3.2 summarizes two common variable transformations.

Parameter Space	Transform Name	Transform Formula	Recovery
$\theta \in \mathbb{R}^+$	Log: $\log(\theta)$	$\theta^* = \log(\theta)$	$\theta = \exp(\theta^*)$
$\theta \in [a, b]$	Expit: $\text{expit}(\theta; a, b)$	$\theta^* = \log\frac{\theta-a}{b-\theta}$	$\theta = \frac{b\exp(\theta^*)+a}{\exp(\theta^*)+1}$

Table 3.2: Variable Transformations for Unconstrained Optimization

3.2.2 Gradient-Based Methods

The efforts dedicated to the construction of a differentiable particle filter culminate in a gradient-based approach for parameter inference. That is, instead of sampling the posterior with particle MCMC, the mode of the particle-estimated marginal posterior can be directly found by a gradient-based optimizer. Along with the Hessian matrix taken at the mode, the posterior can be obtained via normal approximation, as discussed in Section 3.1.

Single-Seed Optimization

An optimization is said to be deterministic if the objective function and its gradient have no randomness. For particle-estimated marginal posterior, which are inherently stochastic, deterministic optimization is performed by fixing a single seed. That is, for all iterations

of the optimizer, the seed that realizes all the random variables in the system remains unchanged.

The simplest choice for deterministic gradient-based optimizer is gradient descent, outlined in Algorithm 3. A typical convergence criterion is given by a combination of tolerance on the magnitude of the gradient update and a maximum number of iterations.

Algorithm 3 Gradient Descent

Initialize starting point $\boldsymbol{\theta}_0$, learning rate η
while not converged **do**
  $t \leftarrow t + 1$
  $\nabla J(\boldsymbol{\theta}_{t-1}) = \frac{\partial J}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{t-1}}$ // Compute gradient
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta \nabla J(\boldsymbol{\theta}_{t-1})$ // Update parameter
end while
Return $\boldsymbol{\theta}_t$

Multi-Seed Optimization

The marginal posterior estimate from particle filtering, as mentioned above, has an inherent randomness (and so as its gradient) which requires a fixed seed across all iterations to employ deterministic optimization methods. However, single-seed optimization incurs a bias associated with that particular choice of seed. This is because fixing a seed is equivalent to using the same subset of random components to calculate the gradient throughout the optimization process.

To mitigate this, one can either increase the number of particles, hence the size of the subset, or consider multi-seed optimization. That is, instead of relying on one seed, the posterior estimate and its gradient are fed a different seed in each of the iterations. Algorithm 4 outlines a popular stochastic optimization algorithm: Adam, a momentum-based method with adaptive learning rate (Kingma & Ba, 2014).

While multi-seed optimization methods may not suffer seed-specific bias and work better when the objective function is multimodal at a fixed seed, it is more difficult to determine when convergence is achieved. Because of the noise of the objective function, thresholds on magnitude of gradient updates become unusable for convergence criterion. Calculation of the Hessian matrix at the mode may also be difficult without knowing whether the mode has been reached.

An alternate multi-seed approach is to perform many single-seed optimization routines, and build the normally approximated posterior using the average mode and quadrature from the optimization results. While it may be easier to find the mode and take the quadrature per seed, the obvious drawback is the computational cost that grows with the number of single-seed optimization routines to perform. However, if a compute cluster is accessible, the optimization routines can be distributed and run completely in parallel as they do not depend on each other.

Algorithm 4 Adam

Initialize hyperparameters: learning rate α, parameters $\beta_1, \beta_2 \approx 1$, small constant ϵ
Initialize starting point $\boldsymbol{\theta}_0$, $\mathbf{m}_0 \leftarrow 0$, $\mathbf{v}_0 \leftarrow 0$, $t \leftarrow 0$
while not converged **do**
  $t \leftarrow t + 1$
  $\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$ // Compute gradient
  $\mathbf{m}_t \leftarrow \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot \mathbf{g}_t$ // Update first moment estimate
  $\mathbf{v}_t \leftarrow \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{g}_t^2$ // Update second moment estimate
  $\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{1-\beta_1^t}$ // Compute bias-correction for first moment
  $\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{1-\beta_2^t}$ // Compute bias-correction for second moment
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}+\epsilon}$ // Update parameter
end while
Return $\boldsymbol{\theta}_t$

Chapter 4

Experimental Results

4.1 SVCJ Model Specification

The SVCJ model of interest in this thesis is given by the following:

$$\begin{aligned}
&\mathrm{d}S(t) = \alpha S(t)\mathrm{d}t + \sqrt{V(t)}S(t)\mathrm{d}B^S(t) + J^S(t)S(t)\mathrm{d}N_\lambda(t)\\
&\mathrm{d}V(t) = \kappa(\theta^\star - \log V(t))V(t)dt + \sigma V(t)\mathrm{d}B^V(t) + J^V(t)\mathrm{d}N_\lambda(t)\\
&\log\big(J^S(t)+1\big) \sim N(\mu_x, \sigma_x^2), J^V(t) \sim Expo(1/\mu_z)\\
&\mathrm{Corr}(\mathrm{d}B_S(t), \mathrm{d}B_V(t)) = \rho.
\end{aligned}$$

Applying Ito's lemma for $X(t) = \log S(t)$ and $Z(t) = \log V(t)$, the resulting model becomes

$$\begin{aligned}
&\mathrm{d}X(t) = (\alpha - \frac{1}{2}V(t))\mathrm{d}t + \sqrt{V(t)}\mathrm{d}B^S(t) + J^X(t)\mathrm{d}N_\lambda(t)\\
&\mathrm{d}Z(t) = \kappa(\theta - Z(t))dt + \sigma\mathrm{d}B^V(t) + J^Z(t)\mathrm{d}N_\lambda(t)\\
&J^X(t) \sim N(\mu_x, \sigma_x^2), (e^{J^Z(t)} - 1)V(t) \sim Expo(1/\mu_z)\\
&\mathrm{Corr}(\mathrm{d}B_S(t), \mathrm{d}B_V(t)) = \rho.
\end{aligned} \tag{4.1}$$

The underlying log volatility follows an Ornstein–Uhlenbeck (OU) process, which renders the volatility an Exponential OU (ExpOU) process (Fouque, Papanicolaou, & Sircar, 2000). The asset price jump is the well-established Merton jump, whereas the choice of the volatility jump requires additional justification: the jump, whose magnitude is given by an Exponential distribution, is additive in the volatility, while the diffusion of the volatility is geometric. The transformation on $J^Z(t)$ is the direct result of the Ito formula for jump-diffusion processes in Equation (C.1) (Cont & Tankov, 2004).

The model parameters are

$$\boldsymbol{\theta} = (\alpha, \theta, \kappa, \sigma, \lambda, \mu_x, \sigma_x, \mu_z, \rho), \tag{4.2}$$

where transformation and interpretation of each parameter is given in Table 4.1.

Parameter	Transformation	Interpretation
α	None: α	Constant drift rate of log price
θ	None: θ	Long-term mean of log volatility
κ	Log: $\log(\kappa)$	Mean reversion speed of log volatility
σ	Log: $\log(\sigma)$	Volatility of log volatility
λ	Expit: $\text{expit}(\lambda; 0, 1)$	Jump propensity
μ_x	None: μ_x	Mean jump size in log price
σ_x	Log: $\log(\sigma_x)$	Standard deviation of jump size in log price
μ_z	Log: $\log(\mu_z)$	Mean jump size in volatility
ρ	Expit: $\text{expit}(\rho; -1, 1)$	Correlation between log price and log volatility in their Brownian increments

Table 4.1: Interpretation of SVCJ Model Parameters

The log volatility at starting time is given by $Z(0) \sim N(\theta, \sigma^2/2\theta)$, using the long-term mean and unconditional variance of an OU process. For the parameters, an uninformative prior is taken in the transformed parameter space, such that

$$\pi(\boldsymbol{\theta}_{tf}) \propto 1 \text{ for } T(\boldsymbol{\theta}_{tf}) \in \mathbb{R}^9.$$

Since the prior is flat, the marginal posterior estimate is essentially equivalent to the marginal likelihood. Thus, the objective function to minimize is thus the particle-estimated negative marginal log-likelihood

$$f_{obj}(\boldsymbol{\theta}_{tf}) = -\log(\hat{p}(\mathbf{Y}_{0:N} \mid \boldsymbol{\theta}_{tf}))$$

as in Algorithm 1. As given by Equation (3.1), the posterior density of $\boldsymbol{\theta}_{tf}$ is approximated by $N(\hat{\boldsymbol{\theta}}_{tf}, \hat{\boldsymbol{\Sigma}}_{tf})$, where

$$\hat{\boldsymbol{\theta}}_{tf} = \text{argmin}_{\boldsymbol{\theta}_{tf}} f_{obj}(\boldsymbol{\theta}_{tf}), \hat{\boldsymbol{\Sigma}}_{tf} = \left[\frac{\partial^2}{\partial \boldsymbol{\theta}^2} f_{obj}(\boldsymbol{\theta}_{tf}) \Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} \right]^{-1}.$$

4.1.1 Choice of Initial Point

For the gradient-based optimization, a well-informed starting point $\boldsymbol{\theta}_0$ is helpful for an efficient search of the mode. However, to ensure fairness of the assessment, it must not rely on some prior knowledge of the parameters that are unattainable in real-life scenarios. Nonetheless, educated guesses on certain parameters are possible based on empirical properties of the log price data and assumption on the rarity of the jump events, as outlined in Table 4.2. All other parameters are initialized at 0 in their transformed (hence unconstrained) scale.

Parameter	Starting point
θ	Log of the variance of log returns
λ	Some small probability (e.g. 0.03)
μ_x	Average of $\lambda\Delta t \cdot 100\%$ worst case log returns ($CVaR_{1-\lambda\Delta t}$)
σ_x	Standard deviation of $\lambda\Delta t \cdot 100\%$ worst case log returns

Table 4.2: Starting Points for SVCJ Model Parameters

4.1.2 Hyperparameter Configuration

The particle filter and optimizer for parameter inference of the SVCJ model are configured as in Table 4.3.

Hyperparameter	Setup
Number of particles	300
Number of inter-observations	10
Optimizer	Adam
Number of optimizer iterations	200
Learning rate	Exponential Decay (in Figure 4.1) • Initial Learning Rate: 0.1 • Decay Rate: 0.01 • Transition Steps: 1,000

Table 4.3: Configuration for Particle Filter and Optimizer

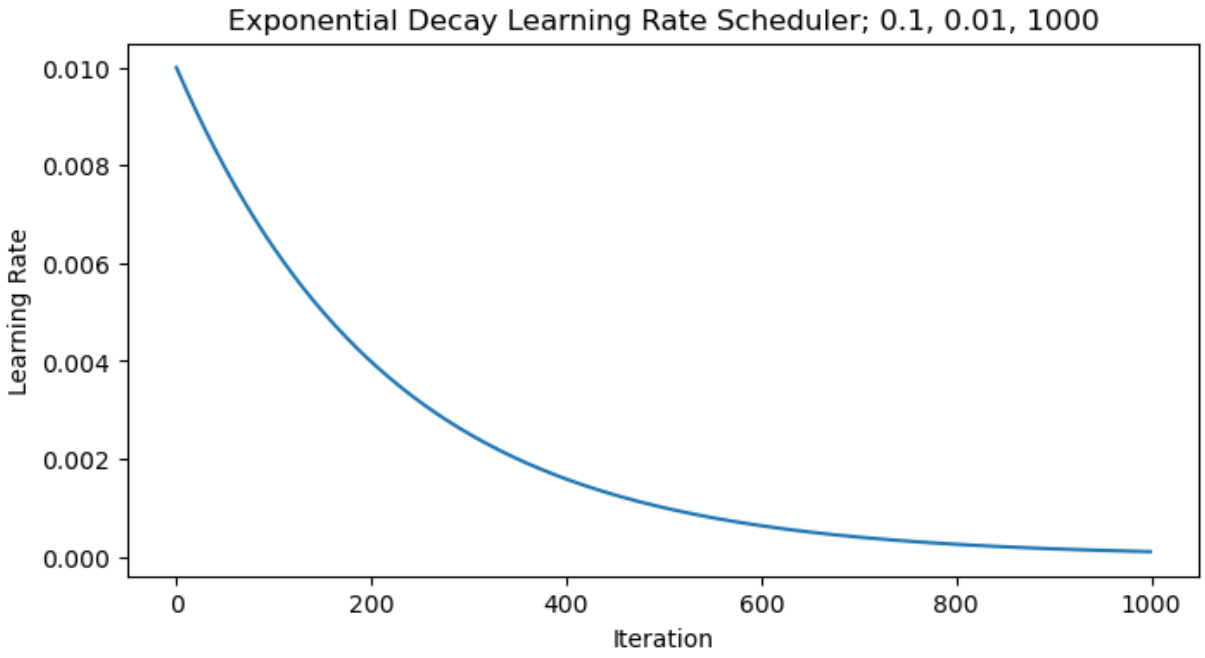

Figure 4.1: Exponential decay learning rate scheduler across 1,000 iterations.

The choice of hyperparameters in Table 4.3 amounts to the total number of latent variables of

$$\begin{aligned} N_{\text{total}} &= 10 \text{ inter-observations} \times 2 \text{ state variables} \times N_{\text{obs}} \\ &= 20 \times N_{\text{obs}}, \end{aligned}$$

where 300 particles are integrated at each of the 200 iterations.

4.2 Computational Specification

4.2.1 Implementation in JAX

JAX is a high-performance computing library in Python suitable for computationally intensive machine learning tasks. (Bradbury et al., 2018). Key features of JAX are just-in-time (JIT) compilation, automatic differentiation, vectorization, and parallelization, which in combination provide fast execution speed and low memory usage.

An efficient, JAX-based particle filter is provided in the `pfjax` library (Lysy et al., 2022). The stochastic volatility with jump model and MVN resampler are fully implemented in JAX and compatible with `pfjax` functionalities. A JAX-based optimization framework is provided by the `optax` library, in which various optimization algorithms and learning rate schedulers are implemented in a highly modularized fashion (DeepMind et al., 2020).

4.2.2 Computational Environment

Single-seed optimization routine was performed using an Apple M2 8-core CPU Processor (4 high-performance @3.5 GHz and 4 efficiency @2.42 GHz). Large-scale optimization jobs involving many single-seed optimization routines were run in parallel by leveraging the `hpc-pr3` cluster, provided by the University of Waterloo's Math Faculty Computing Facility (MFCF). The configuration details are given in Table 4.4.

Computational Parameter	Setup
CPU Type	Intel(R) Xeon(R) Gold 6326 CPU @2.90GHz
CPU per task	20 CPUs
Number of seeds per task	10 seeds
Memory per CPU	4GB

Table 4.4: Computational Parameters for Large-Scale Jobs

4.3 Simulation Study

This section presents the parameter inference results on the SVCJ model using a synthetic dataset comprising 1,260 observations—equivalent to 5 years of 252 trading days—to investigate the performance of the differentiable particle filter. The model parameters for the simulation study are given by:

$$\begin{aligned}\boldsymbol{\theta} &= (\alpha, \theta, \kappa, \sigma, \lambda, \mu_x, \sigma_x, \mu_z, \rho) \\ &= (0.15, \log(0.12), 0.022, 0.19, 0.0084, -3.1, 1.7, 0.65, -0.5) \\ X(0) &= 100, Z(0) \sim N(\theta, \sigma^2/2\theta).\end{aligned}$$

The synthetic dataset was generated with Euler discretization as outlined in Subsection 2.2.1 with 10 intra-day steps between adjacent observations. One instance of the resulting

log price and underlying volatility are shown in Figure 4.2. Note that only the end-of-day log price is taken as the observation, since both intra-day log price and underlying volatility are unobserved latent processes.

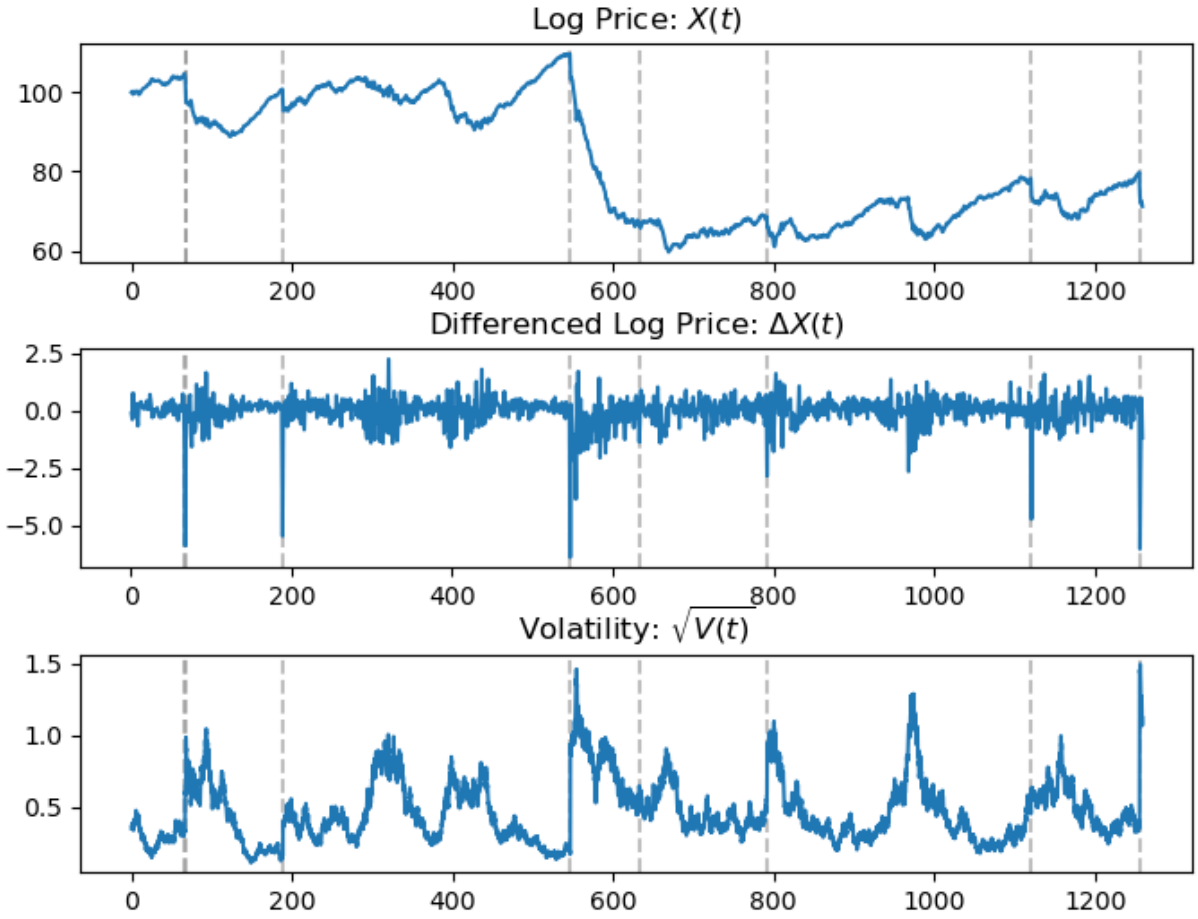

Figure 4.2: Synthetic example dataset generated from the SVCJ model, consisting of: log price (top), differenced log price (middle), underlying volatility (bottom), with jump occurrences shown in grey dashed line.

4.3.1 MVN Resampler vs Multinomial Resampler

It is first required to validate whether the MVN resampler is (1) differentiable, and (2) an adequate substitution for the multinomial resampler. A set of projection plots of the (neg-

ative) marginal likelihood estimate, obtained by varying the values of each (transformed) model parameter while holding all other constant (including the seed), serves as a visual check.

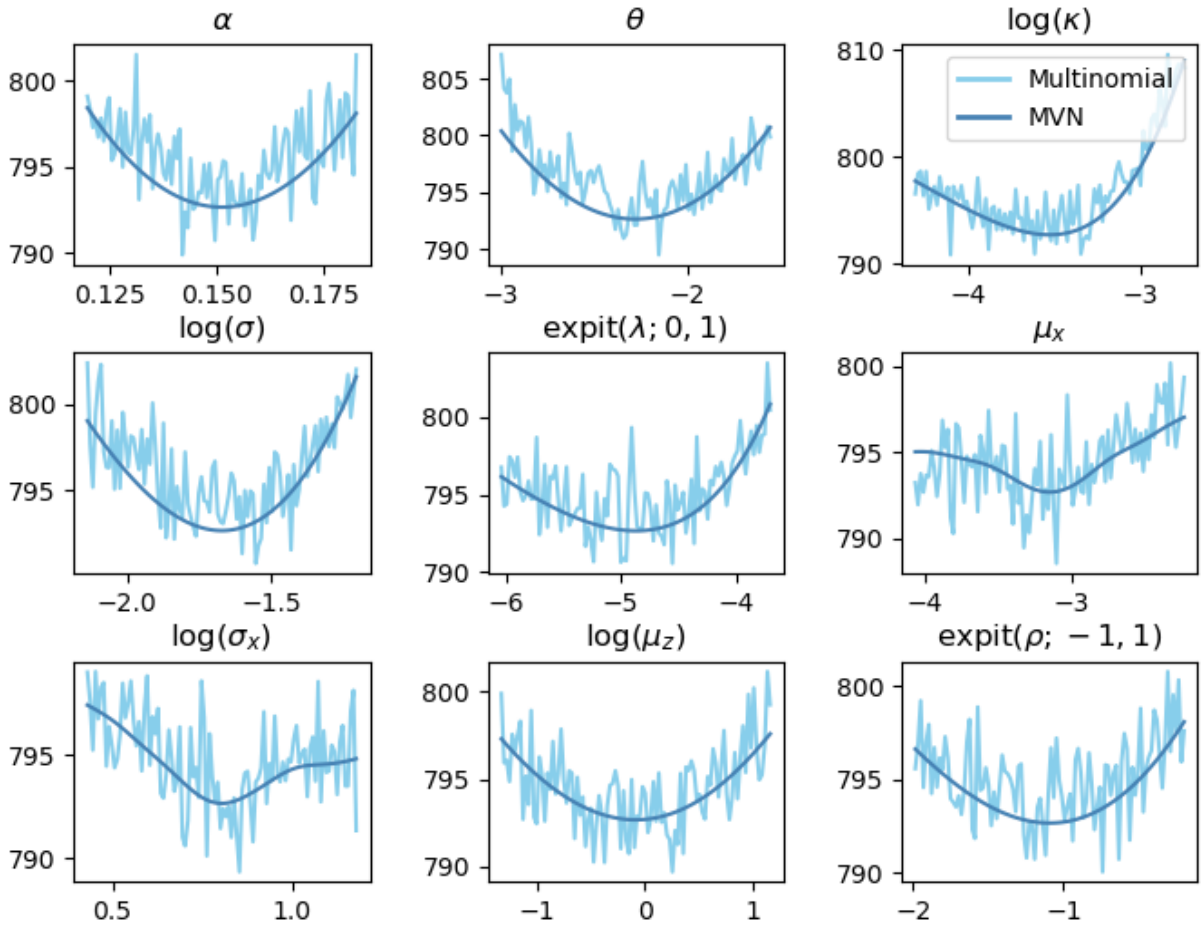

Figure 4.3: Projection plot of marginal likelihood for each model parameter, estimated from MVN resampler (blue) and multinomial resampler (sky blue).

Figure 4.3 confirms the differentiability of marginal likelihood estimates using the MVN resampler, as the curves are visibly smooth in all projection plots. In addition, from this instance, there is no evidence of discernible bias introduced by the MVN resampler in the location of the mode.

Projection Plot for Gumbel-Softmax Filter

Figure projection plot, generated from the differentiable particle filter that employs the Gumbel-Softmax reparameterization trick in Subsection 3.1.3. Note that the location of the mode is substantially different than in Figure 4.3 for most parameters. More importantly, the log-likelihood curve for the jump propensity parameter λ is shown to display either roughness (with small τ) or multimodality (with large τ). These may lead to non-differentiability with respect to λ or incorrect convergence to local minima.

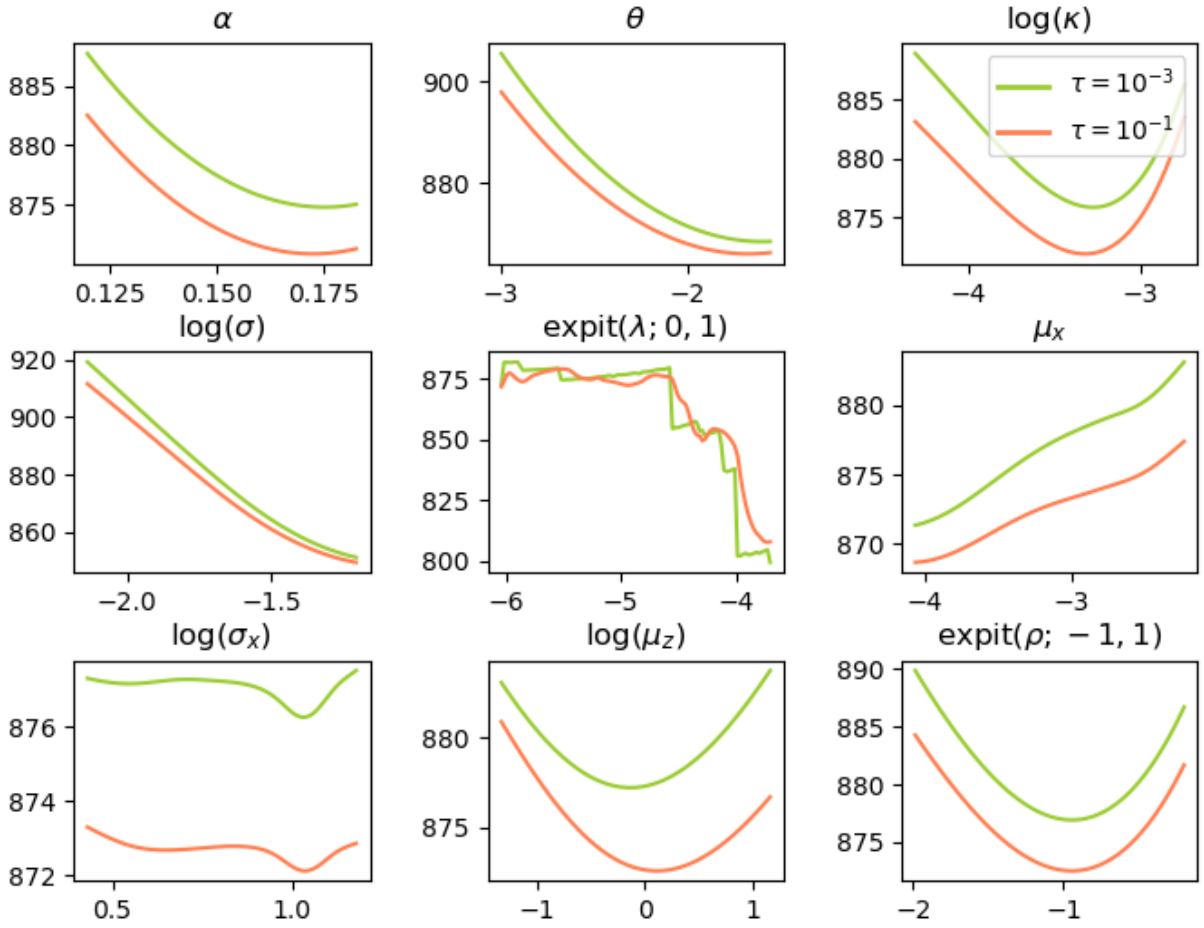

Figure 4.4: Projection plot of marginal likelihood for each model parameter, estimated from the Gumbel-Softmax filter with varying smoothing factor.

4.3.2 Parameter Inference

The inference results are presented in this section, consisting of three versions:

- Single-seed optimization on dataset in Figure 4.2 (hereinafter example dataset)
- Many single-seed optimizations on dataset in example dataset
- Single-seed optimization on many datasets

Single-seed Optimization on Example Dataset

A single single-seed optimization routine on the example dataset was performed with convergence result in Figure 4.5 and posterior densities in Figure 4.6. The optimization routine and Hessian matrix calculation were completed within 7-8 minutes. The model parameters are well-estimated, as evidenced by the mode located near the true value for most parameters. The standard deviation of the log price jump size is slightly misaligned with its true value, but this is to be expected as only 7 jumps occurred in the example dataset.

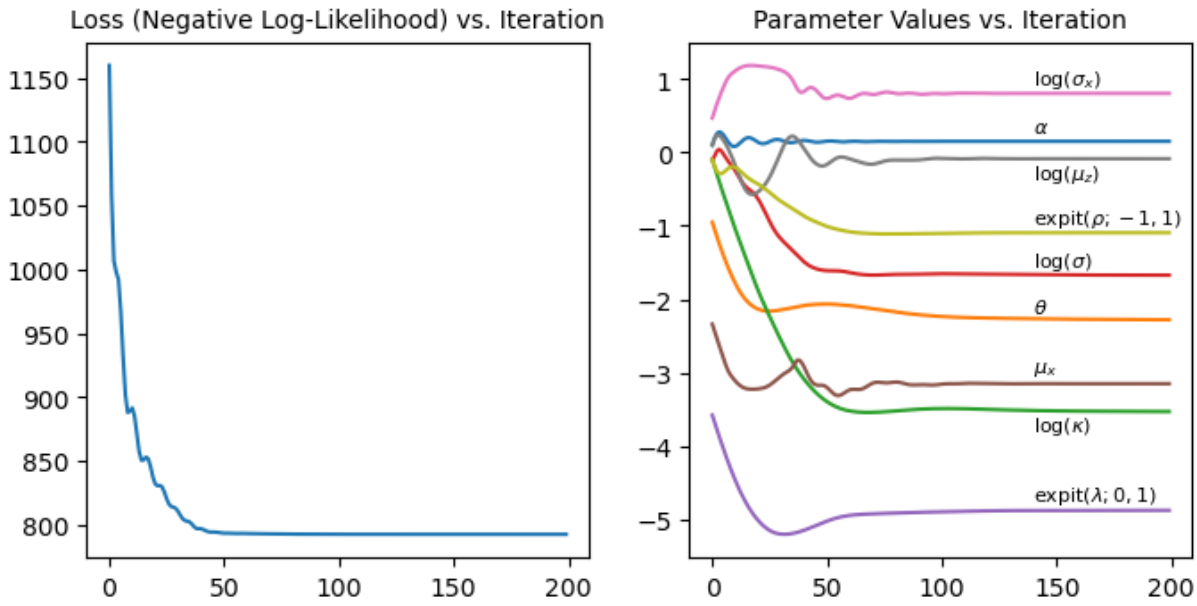

Figure 4.5: Loss (negative marginal log-likelihood estimate) and parameter values (in transformed scale) versus optimization iterations.

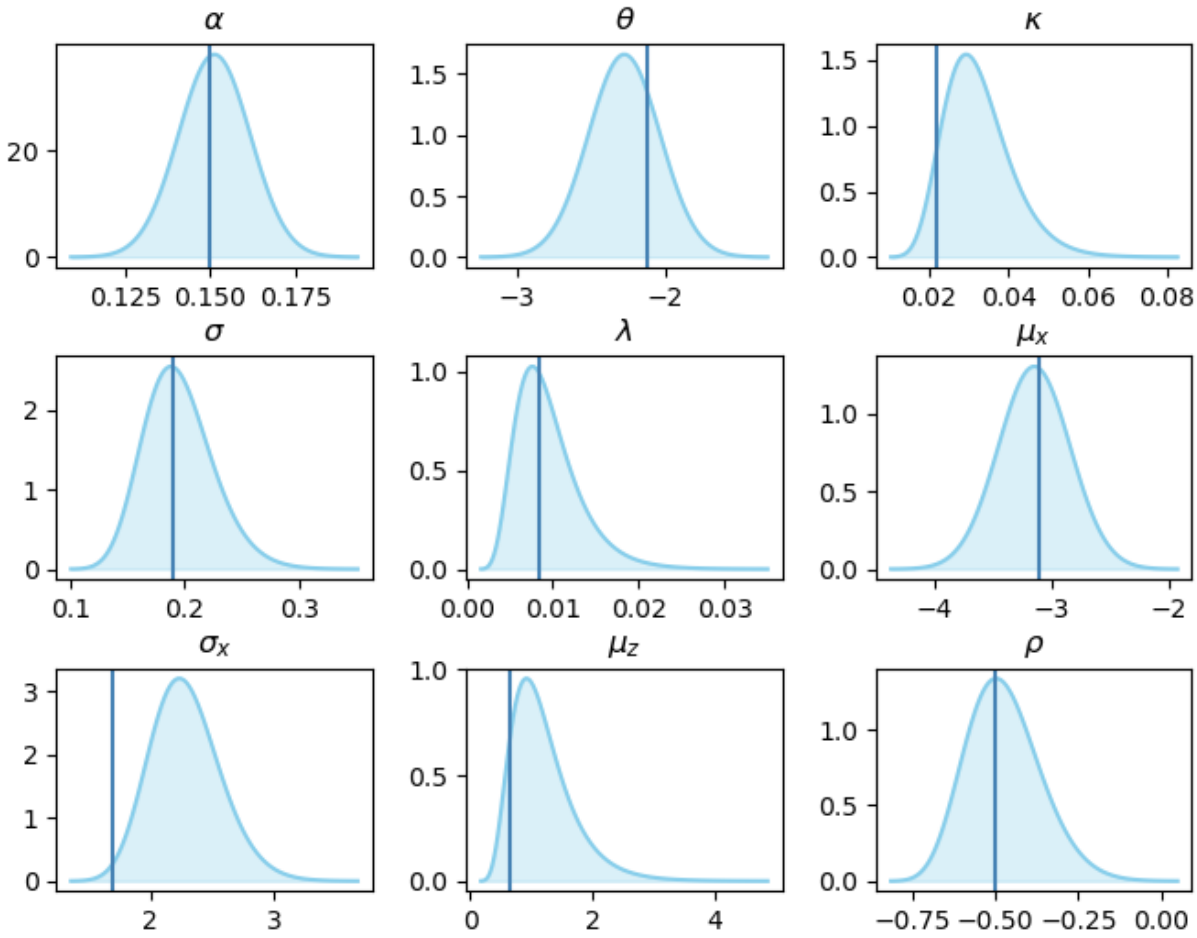

Figure 4.6: Normally-approximated posterior densities via single-seed optimization routine (converted into original parameter scale), with true values in vertical lines.

	α_{tf}	θ_{tf}	κ_{tf}	σ_{tf}	λ_{tf}	$\mu_{x,tf}$	$\sigma_{x,tf}$	$\mu_{z,tf}$	ρ_{tf}
True Value	0.150	-2.120	-3.817	-1.661	-4.771	-3.100	0.531	-0.431	-1.099
Estimate	0.151	-2.278	-3.527	-1.671	-4.872	-3.148	0.804	-0.086	-1.093
Std. Error	0.010	0.241	0.258	0.156	0.389	0.306	0.125	0.417	0.298

Table 4.5: Parameter Estimates in Transformed Scale for Example Dataset (Single-Seed)

Many Single-seed Optimization on Example Dataset

To verify the consistency of the result above, 100 single-seed optimization routines were performed on the example dataset. With configuration outlined in Table 4.4, 100 seeds were partitioned into 10 tasks and completed with average wall-time of 13-14 minutes per task. The log price jump size parameters μ_x, σ_x show relatively large variability in their estimates in Figure 4.7. This aligns with the observation made above in Subsection 4.3.2.

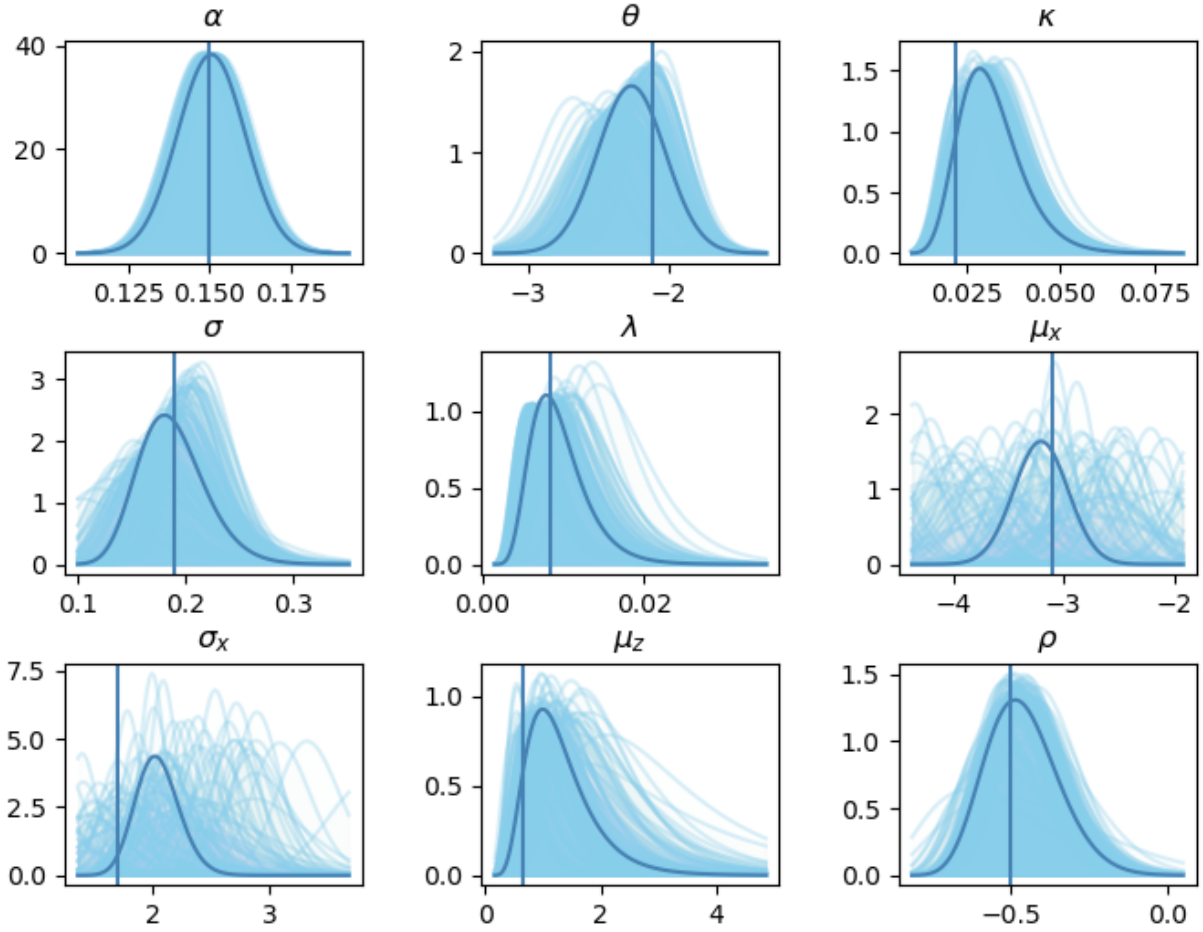

Figure 4.7: Normally-approximated posterior densities via 100 single-seed optimization routines (converted into original parameter scale). In each plot, density in sky blue represents each single-seed optimization result, and blue curve is given by the average mode and quadrature.

	α_{tf}	θ_{tf}	κ_{tf}	σ_{tf}	λ_{tf}	$\mu_{x,tf}$	$\sigma_{x,tf}$	$\mu_{z,tf}$	ρ_{tf}
True Value	0.150	-2.120	-3.817	-1.661	-4.771	-3.100	0.531	-0.431	-1.099
Estimate	0.150	-2.270	-3.559	-1.711	-4.828	-3.206	0.701	-0.002	-1.061
Std. Error	0.010	0.241	0.263	0.165	0.361	0.245	0.091	0.431	0.304

Table 4.6: Parameter Estimates in Transformed Scale for Example Dataset (100 Single-Seed, Average Mode and Quadrature)

Single-seed Optimization on Many Datasets

As the example dataset is only one instance of the SVCJ model, it prompts the question of how optimization method performs across different realizations. Figure 4.8 illustrates a single-seed optimization routine operated on 1,000 synthetic datasets. Similarly to above, 1,000 seeds were partitioned into 100 tasks and completed with similar wall-time of 13-14 minutes per task. 10 results contained a non-positive definite Hessian matrix and hence were excluded in Figure 4.8, but included in the calculation of average mode and quadrature. The average mode and quadrature appear to align well with each of the true parameter values. Again, it is observed that the jump size parameters μ_x, σ_x exhibit the most variability.

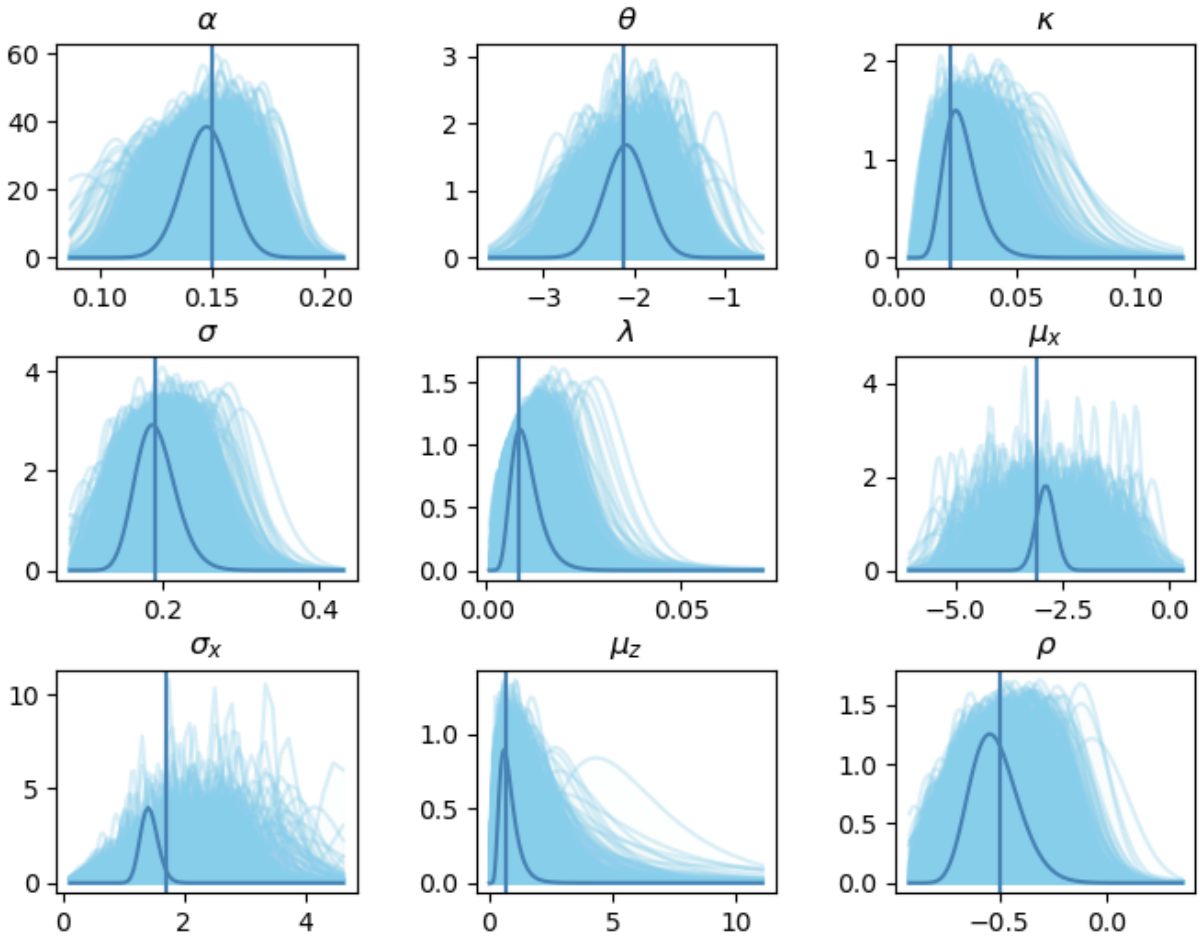

Figure 4.8: Normally-approximated posterior densities via single-seed optimization routine on each of the 1,000 synthetic datasets (converted into original parameter scale). In each plot, density in sky blue represents each single-seed optimization result, and blue curve is given by the average mode and quadrature.

	α_{tf}	θ_{tf}	κ_{tf}	σ_{tf}	λ_{tf}	$\mu_{x,tf}$	$\sigma_{x,tf}$	$\mu_{z,tf}$	ρ_{tf}
True Value	0.150	-2.120	-3.817	-1.661	-4.771	-3.100	0.531	-0.431	-1.099
Estimate	0.147	-2.084	-3.708	-1.683	-4.734	-2.887	0.336	-0.507	-1.225
Std. Error	0.010	0.238	0.266	0.136	0.355	0.219	0.100	0.447	0.317

Table 4.7: Parameter Estimates in Transformed Scale for 1,000 Synthetic Datasets (Single-Seed, Average Mode and Quadrature)

4.3.3 Volatility Filtering

Using the parameter estimates from the many single-seed optimization in Table 4.6, the underlying volatility process of the example dataset is recovered via the optional output of Algorithm 1. The mean filtered volatility is obtained by the weighted average of the particles, and the 95% confidence band by the ± 1.96 weighted standard deviation. As seen in Figure 4.9, the underlying volatility is well-captured by the confidence band.

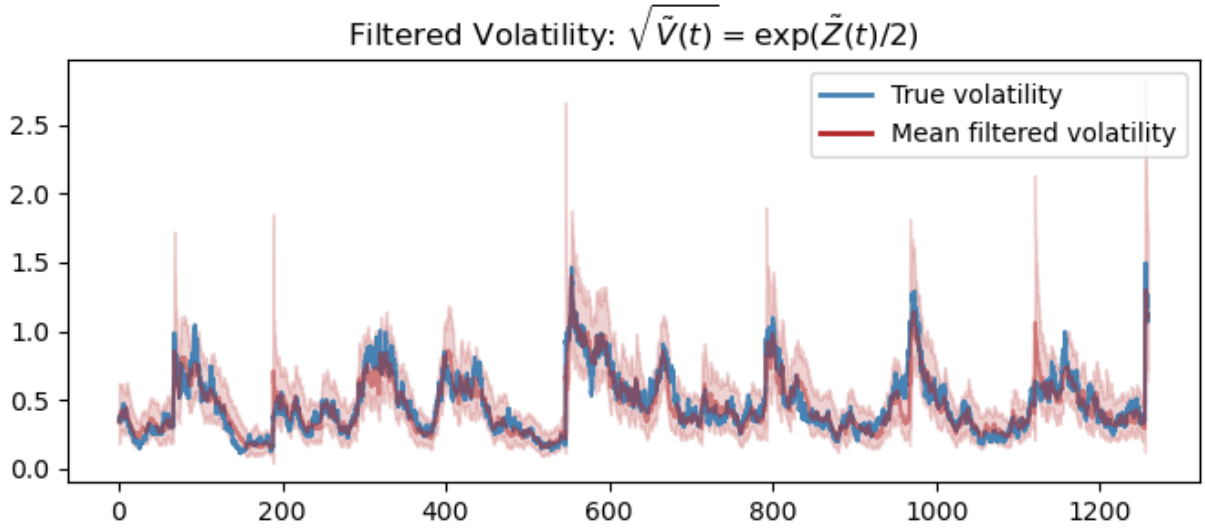

Figure 4.9: Filtered volatility with 95% confidence band constructed by particle filtering with parameter estimates in Table 4.6 for the example dataset.

4.4 Real-World Data: S&P 500

Daily observations on the adjusted closing price of the S&P 500 Index data were retrieved with Python's `yfinance` package for the period of January 1st, 2019 to January 1st, 2024, totalling 1,258 trading days. Because the SVCJ model in Equation (4.1) is specific to this thesis, there is no prior knowledge on the parameter values calibrated to S&P 500 data in literature (most previous studies have focused on the Merton jump-diffusion coupled with Heston model). The dataset was normalized such that the logarithm of the closing price begins at 100:

$$y_{0:N} \leftarrow \log(\text{Price}_{0:N}) \times \frac{100}{\log(\text{Price}_0)}.$$

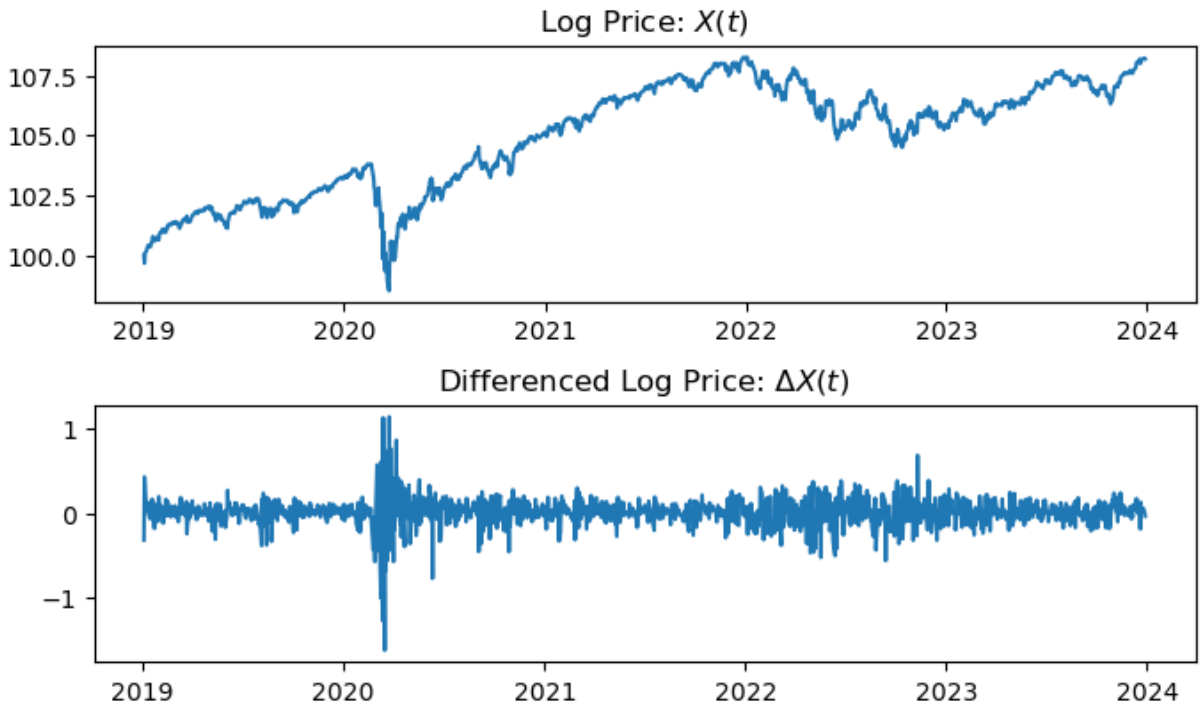

Figure 4.10: S&P 500 price data from January 1st, 2019 to January 1st, 2024, consisting of: rescaled log price (top) and differenced rescaled log price (bottom).

4.4.1 Parameter Inference

The many-single-seed approach was employed for parameter inference of the (rescaled) S&P 500 data, involving 100 single-seed optimization routines as in Subsection 4.3.2. The computing performance is very similar to that of Subsection 4.3.2, since the number of observations is almost identical. 7 results contained a non-positive definite Hessian matrix. Again, while it is excluded in Figure 4.11, it is factored into the average mode and quadrature.

Figure 4.11 shows the inference results from the 100 single-seed optimization routines. 7 of these resulted in a non-positive definite Hessian matrix, hence excluded from Figure 4.11, but nonetheless included in average mode and quadrature calculation, similarly to Subsection 4.3.2. While the diffusion parameters $(\alpha, \theta, \kappa, \sigma, \rho)$ show consistency in the average mode and quadrature, the remaining parameters governing the jumps exhibit larger variability. This aligns with the findings in the simulation study in Section 4.3.

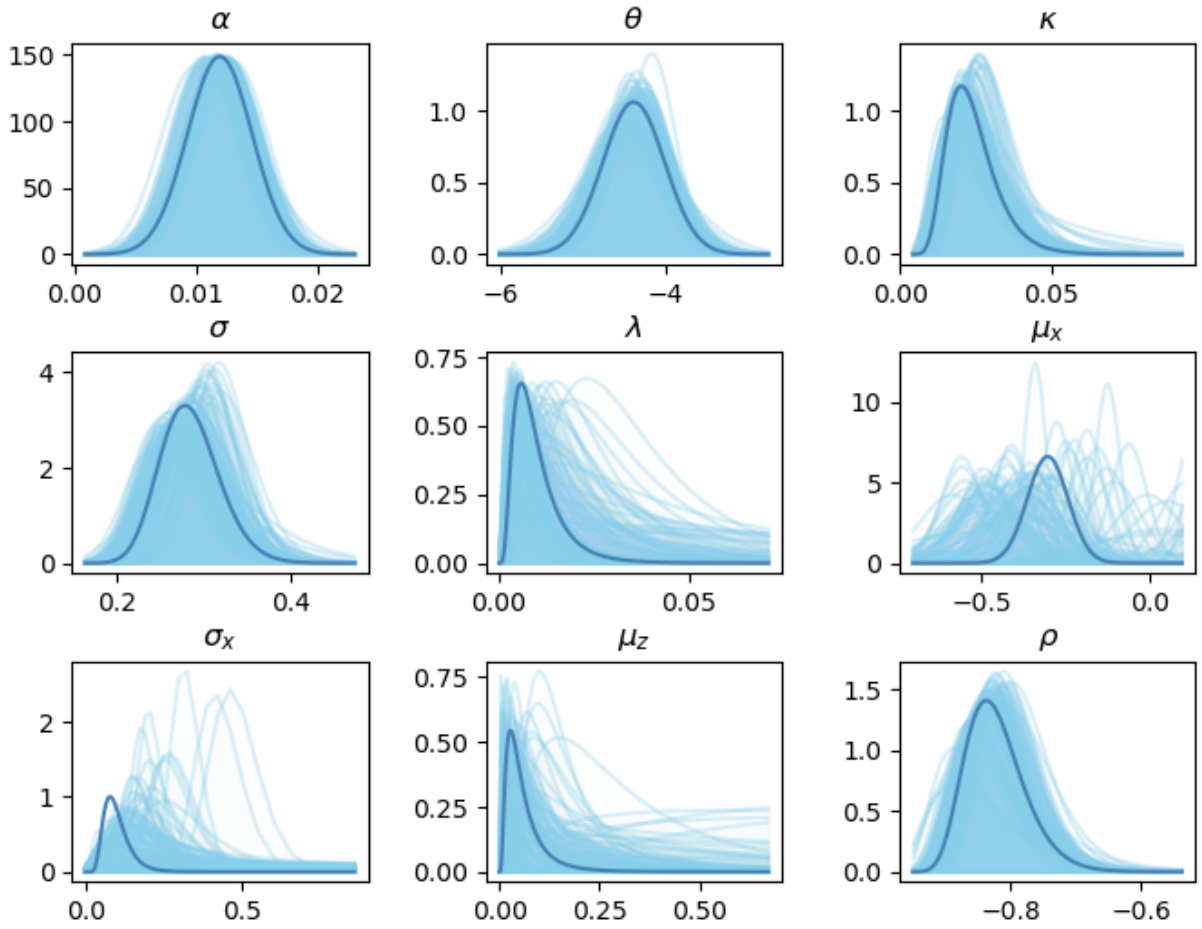

Figure 4.11: Normally-approximated posterior densities via 100 single-seed optimization routines (converted into original parameter scale). In each plot, density in sky blue represents each single-seed optimization result, and blue curve is given by the average mode and average quadrature.

	α_{tf}	θ_{tf}	κ_{tf}	σ_{tf}	λ_{tf}	$\mu_{x,tf}$	$\sigma_{x,tf}$	$\mu_{z,tf}$	ρ_{tf}
Estimate	0.012	-4.380	-3.903	-1.280	-5.106	-0.307	-2.522	-3.617	-2.424
Std. Error	0.003	0.376	0.340	0.121	0.608	0.060	0.397	0.733	0.284

Table 4.8: Parameter Estimates in Transformed Scale for S&P 500 (Single-Seed on 1,000 Datasets, Average Mode and Quadrature)

4.4.2 Volatility Filtering

While the ground truth for the underlying volatility of the S&P 500 data is unknown, there are several proxies available:

- **Close-to-Close Volatility**: Standard deviation of log returns over a certain time frame, such as 30-day or 90-day window.
- **VIX**: A commonly used measure for market volatility that estimates the implied volatility of S&P 500 options with an average expiration of 30 days (CBOE, 1993). Publicly available on the CBOE VIX dashboard.
- **SPOTVOL**: A recently released index by Chicago Board of Options Exchange (CBOE) for the spot volatility of S&P 500 Index (CBOE, 2024). Publicly available on the CBOE SPOTVOL dashboard.

SPOTVOL serves as the most appropriate benchmark for the filtered volatility, since the close-to-close volatility is a crude approximation highly dependent on the selection of the window. VIX includes additional investor sentiment of the volatility in the next 30 days. In Figure 4.12, it is demonstrated that the recovered volatility via particle filtering with the parameter estimates obtained in Table 4.8 aligns well with SPOTVOL.

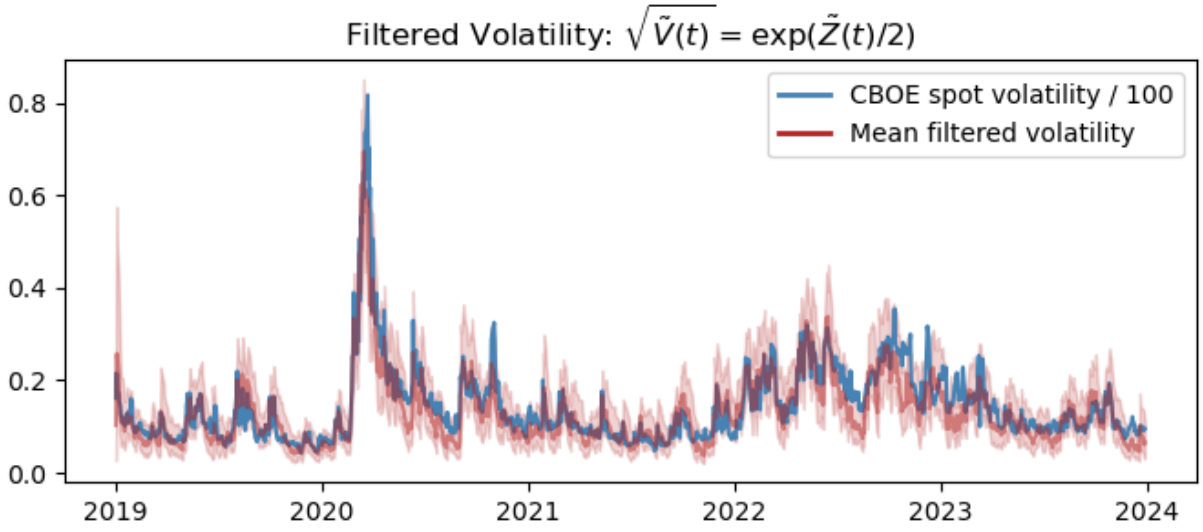

Figure 4.12: Filtered volatility with 95% confidence band constructed by particle filtering with parameter estimates in Table 4.8 for the S&P 500 dataset, with SPOTVOL.

4.4.3 Jump Filtering

The idea of latent state recovery can extend to filtering jump indicators. That is, the weighted mean of the particles' Bernoulli jump variables roughly gives the probability whether the jump has occurred in the day.

The critical days in which the mean probability of jump is greater than 0.4 are labeled in Figure 4.13. It is interesting to note that, during the COVID-induced 2020 stock crash between February 2020 and April 2020, the first fall on Monday February 24th was captured as a jump event, while the March Black Mondays (March 9th and 16th) and Black Thursday (March 12th) were not. This may be due to the already heightened level of fear in the market at the beginning of the pandemic that translated into high volatility, as seen in Figure 4.12.

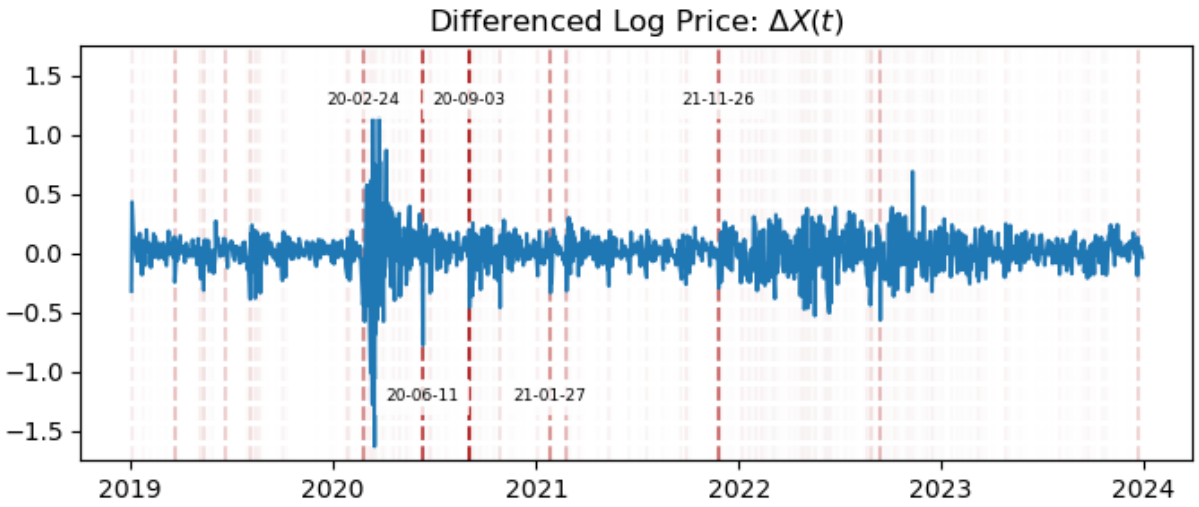

Figure 4.13: Filtered jumps from particle filtering for the S&P 500 dataset; represented as red vertical dashed lines with opacity given by the mean probability, overlaid onto differenced rescaled log price.

Chapter 5

Conclusion

In this thesis, we introduced a differentiable particle filter tailored to a class of jump-diffusion models, specifically the stochastic volatility with contemporaneous jump models. High levels of non-linearity and non-differentiability, exhibited by these models, pose a nontrivial challenge to the parameter inference problems.

Firstly, the jump-diffusion model was discretized, represented as a state-space model, and coupled with a bridge proposal to handle the degeneracy problem associated with error-free observations in particle filtering in Subsection 2.3.2. By identifying two sources of discontinuities—one in the model and one in the filter—and suggesting appropriate remedies in Section 3.1, the marginal likelihood estimate transformed into a differentiable quantity. This facilitated an efficient search of the mode and Hessian matrix calculation via a gradient-based optimization method, such that the posterior densities of the model parameters can be normally approximated, in lieu of a computationally intensive MCMC sampler.

The results of parameter inference performed in Section 4.3 using synthesized dataset showcased alignment between the mode of the particle-estimated marginal likelihood and the true parameter values. The variability in the jump-related parameter estimates were observed to be greater than the diffusion-related ones, which is an expected phenomenon given that jumps are assumed rare events. The performance of the differentiable particle filter for parameter inference on real-world S&P 500 Index data was demonstrated in 4.4. In particular, the underlying volatility recovered by the particle filter with its parameter estimates is shown to match well with CBOE's spot volatility index.

5.1 Limitation and Future Direction

Below are several inquiries that arise from this thesis and may be worthwhile to investigate in further detail:

- The SVCJ model specified in this thesis only considers one kind of jump in the asset price, usually jumping in a negative direction. It may be of interest to consider different types of jump processes that admit both directions, such as the Kou's double-exponential jumps.
- The model does not account for time-dependent market behaviours, such as the weekend effects. This is a more difficult point to consider; substantial knowledge on time-related trends of financial market data is required for an appropriate treatment.
- It is assumed that the model parameters stay constant for the entire duration of the data. This may be problematic if the data contains an event that permanently alters the dynamics of the asset price. The performance of this method for a regime-switching model remains a question.
- The differentiable particle filter here relies on the fact that the jump-diffusion model is Markovian. This implies that this approach is not immediately applicable for models with non-Markov processes, such as rough volatility models.
- Practitioners are often interested in the correlation structure across multiple assets. The work presented in this thesis may have the potential of extension to inference of multi-asset models.

References

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(3), 269–342.

Bates, D. S. (1996). Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. *The Review of Financial Studies*, *9*(1), 69–107. https://doi.org/10.1093/rfs/9.1.69

Bates, D. S. (2000). Post-'87 crash fears in the S&P 500 futures option market. *Journal of Econometrics*, *94*(1-2), 181–238. https://EconPapers.repec.org/RePEc:eee:econom:v:94:y:2000:i:1-2:p:181-238

Black, F., & Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable Transformations of Python+NumPy Programs* (Version 0.3.13). http://github.com/google/jax

Broadie, M., & Kaya, Ö. (2006). Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research*, *54*(2), 217–231.

Casella, B., & Roberts, G. O. (2011). Exact Simulation of Jump-Diffusion Processes with Monte Carlo Applications. *Methodology and Computing in Applied Probability*, *13*(3), 449–473. https://doi.org/10.1007/s11009-009-9163-1

CBOE. (1993). Volatility Index Methodology Cboe Volatility Index [Accessed: April 28, 2024].

CBOE. (2024). CBOE S&P 500 Spot Volatility Index Methodology [Accessed: April 28, 2024]. https://cdn.cboe.com/api/global/us_indices/governance/Cboe_SnP_500_Spot_Volatility_Index_Methodology.pdf

Cont, R., & Tankov, P. (2004). *Financial Modelling with Jump Processes*. Chapman & Hall/CRC Financial Mathematics Series.

Corenflos, A., Thornton, J., Deligiannidis, G., & Doucet, A. (2021). Differentiable Particle Filtering via Entropy-Regularized Optimal Transport. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 2100–2111.

DeepMind, Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Dedieu, A., Fantacci, C., Godwin, J., Jones, C., Hemsley, R., Hennigan, T., Hessel, M., Hou, S., ... Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/google-deepmind

Del Moral, P. (1997). Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, *325*(6), 653–658. https://doi.org/10.1016/S0764-4442(97)84778-7

Doucet, A., & Johansen, A. M. (2011). A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. *Oxford Handbook of Nonlinear Filtering*, *12*, 656–704.

Duffie, D., Pan, J., & Singleton, K. (2000). Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, *68*(6), 1343–1376.

Durham, G. B., & Gallant, R. A. (2002). Numerical Techniques for Maximum Likelihood Estimation of Continuous-Time Diffusion Processes. *Journal of Business & Economic Statistics*, *20*, 279–316.

Eraker, B., Johannes, M., & Polson, N. (2003). The Impact of Jumps in Volatility and Returns. *The Journal of Finance*, *58*(3), 1269–1300.

Fouque, J.-P., Papanicolaou, G., & Sircar, R. (2000). Mean-reverting stochastic volatility. *International Journal of Theoretical and Applied Finance*, *3*, 101–142.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian Data Analysis*. Chapman; Hall/CRC.

Glasserman, P., & Merener, N. (2003). Numerical solution of jump-diffusion LIBOR market models. *Finance and Stochastics*, *7*(1), 1–27. https://doi.org/10.1007/s007800200076

Golightly, A. (2009). Bayesian Filtering for Jump-Diffusions With Application to Stochastic Volatility. *Journal of Computational and Graphical Statistics*, *18*(2), 384–400. https://doi.org/10.1198/jcgs.2009.07137

Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, *140*(2), 107–113.

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.

Jang, E., Gu, S., & Poole, B. (2016). Categorical Reparameterization with Gumbel-Softmax. *5th International Conference on Learning Representations (ICLR) 2017 - Conference Track Proceedings*.

Jonschkowski, R., Rastogi, D., & Brock, O. (2018). Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors.

Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Kou, S. G. (2002). A Jump-Diffusion Model for Option Pricing. *Management Science*, *48*(8), 1086–1101. Retrieved March 31, 2024, from http://www.jstor.org/stable/822677

Lysy, M., Subramani, P., Ramkissoon, J., Wu, M., Ko, M., & Chopra, K. (2022). *pfjax: Particle Filtering Methods in JAX*. https://github.com/mlysy/pfjax

Maruyama, G. (1955). Continuous Markov processes and stochastic equations. *Rend. Circ. Mat. Palermo*, *4*, 48–90. https://doi.org/10.1007/BF02846028

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, *3*(1-2), 125–144. https://doi.org/10.1016/0304-405X(76)90022-2

Metwally, S. A., & Atiya, A. (2002). Using Brownian Bridge for Fast Simulation of Jump-Diffusion Processes and Barrier Options. *The Journal of Derivatives*, *10*(1), 43–54. https://doi.org/10.3905/jod.2002.319189

Murray, L. M. (2015). Bayesian State-Space Modelling on High-Performance Hardware Using LibBi. *Journal of Statistical Software*, *67*(10), 1–41. https://doi.org/10.18637/jss.v067.i10

Pan, J. (2002). The jump-risk premia implicit in options: Evidence from an integrated time-series study. *Journal of Financial Economics*, *63*(1), 3–50. https://doi.org/10.1016/S0304-405X(01)00088-5

Pedersen, A. (1995). A New Approach to Maximum Likelihood Estimation for Stochastic Differential Equations Based on Discrete Observations. *Scandinavian Journal of Statistics*, *22*, 55–71.

Rosato, C., Beraud, V., Horridge, P., Schön, T. B., & Maskell, S. (2020). Efficient Learning of the Parameters of Non-Linear Models using Differentiable Resampling in Particle Filters. *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1942–1946.

APPENDICES

Appendix A

Derivation of Incremental Weight

The derivation of incremental weight in Equation (2.12) is illustrated for $\rho = 0$. The general case $\rho \neq 1$ follows similar procedure.

The transition density for $\mathbf{X}_k = (S_k^{(1:M)}, V_k^{(1:M)}, Q_k^{(1:M)}, J_k^{S,(1:M)}, J_k^{V,(1:M)})$, the latent state variables of the SVCJ model at observation time k, is given by

$$
\begin{aligned}
f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta}) &= \prod_{j=1}^{M} p(\mathbf{X}_k^j \mid \mathbf{X}_k^{j-1}, \boldsymbol{\theta}) \\
&= \prod_{j=1}^{M} p(S_k^{(j)} \mid S_k^{(j-1)}, V_k^{(j)}, Q_k^{(j)}, J_k^{S,(j)}, J_k^{V,(j)}, \boldsymbol{\theta}) \\
&\quad \times p(V_k^{(j)} \mid V_k^{(j-1)}, Q_k^{(j)}, J_k^{V,(j)}, \boldsymbol{\theta}) \\
&\quad \times p(J_k^{S,(j)}, J_k^{V,(j)} \mid Q_k^{(j)}, \boldsymbol{\theta}) \\
&\quad \times p(Q_k^{(j)} \mid \boldsymbol{\theta}) \\
&= \prod_{j=1}^{M} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{trans}}(S_k^{(j-1)}), \sigma_{\text{trans}}(S_k^{(j-1)})) \\
&\quad \times \phi(V_k^{(j)}; V_k^{(j-1)} + \underbrace{\alpha_{\boldsymbol{\theta}}(V_k^{(j-1)}, t_k^{(j-1)})\Delta^* t + J_k^{S,(j)} Q_k^{(j)}}_{\alpha_{j-1}(V_k^{(j-1)})}, \underbrace{\beta_{\boldsymbol{\theta}}(V_k^{(j-1)}, t_k^{(j-1)})}_{\beta_{j-1}(V_k^{(j-1)})}) \\
&\quad \times \pi_{J^S}(J_k^{S,(j)}; \boldsymbol{\theta}) \pi_{J^V}(J_k^{V,(j)}; \boldsymbol{\theta}) \\
&\quad \times b(Q_k^{(j)}; \lambda \Delta^* t),
\end{aligned}
$$

where $\phi(\cdot)$ is the normal p.d.f. and $b(\cdot)$ is the Bernoulli p.m.f.

Now for the proposal density as described in Algorithm 2, it follows that

$$\begin{aligned} q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta}) = & \prod_{h=1}^{M-1} \phi(S_k^{(h)}; S_k^{(h-1)} + \mu_{\text{prop}}(S_k^{(h-1)}), \sigma_{\text{prop}}(S_k^{(h-1)})) \\ & \times \prod_{j=1}^{M} \phi(V_k^{(j)}; V_k^{(j-1)} + \alpha_{j-1}(V_k^{(j-1)}), \beta_{j-1}(V_k^{(j-1)})) \\ & \quad \times \pi_{J^S}(J_k^{S,(j)}; \boldsymbol{\theta}) \pi_{J^V}(J_k^{V,(j)}; \boldsymbol{\theta}) \\ & \quad \times b(Q_k^{(j)}; \lambda \Delta^* t). \end{aligned}$$

Then, we have

$$\frac{f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta})}{q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta})} = \frac{\prod_{j=1}^{M} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{trans}}(S_k^{(j-1)}), \sigma_{\text{trans}}(S_k^{(j-1)}))}{\prod_{j=1}^{M-1} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{prop}}(S_k^{(j-1)}, y_k), \sigma_{\text{prop}}(S_k^{(j-1)}, y_k))},$$

as required.

A.1 Diffusion Bridge with Different Jump Propensity

If the bridge proposal uses a different jump propensity λ^*, we have

$$\begin{aligned} q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta}^*) &= q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \{\lambda^*\} \cup \boldsymbol{\theta} \setminus \{\lambda\}) \\ &= \prod_{h=1}^{M-1} \phi(S_k^{(h)}; S_k^{(h-1)} + \mu_{\text{prop}}(S_k^{(h-1)}), \sigma_{\text{prop}}(S_k^{(h-1)})) \\ &\quad \times \prod_{j=1}^{M} \phi(V_k^{(j)}; V_k^{(j-1)} + \alpha_{j-1}(V_k^{(j-1)}), \beta_{j-1}(V_k^{(j-1)})) \\ &\quad\quad \times \pi_{J^S}(J_k^{S,(j)}; \boldsymbol{\theta}) \pi_{J^V}(J_k^{V,(j)}; \boldsymbol{\theta}) \\ &\quad\quad \times b(Q_k^{(j)}; \lambda^* \Delta^* t), \end{aligned}$$

and the resulting ratio becomes

$$\begin{aligned}\frac{f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta})}{q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta}^*)} &= \frac{\prod_{j=1}^{M} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{trans}}(S_k^{(j-1)}), \sigma_{\text{trans}}(S_k^{(j-1)}))}{\prod_{j=1}^{M-1} \phi(S_k^{(j)}; S_k^{(j-1)} + \mu_{\text{prop}}(S_k^{(j-1)}, y_k), \sigma_{\text{prop}}(S_k^{(j-1)}, y_k))} \\ &\quad \times \frac{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda \Delta^* t)}{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda^* \Delta^* t)} \\ &= \frac{f(\mathbf{X}_k \mid \mathbf{X}_{k-1}, \boldsymbol{\theta})}{q(\mathbf{X}_k \mid \mathbf{X}_{k-1}, y_k, \boldsymbol{\theta})} \times \frac{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda \Delta^* t)}{\prod_{j=1}^{M} b(Q_k^{(j)}; \lambda^* \Delta^* t)}.\end{aligned} \tag{A.1}$$

Appendix B

Gumbel-Softmax Approximation for Bernoulli Random Variable

The Gumbel-Softmax approximation method for a Bernoulli random variable with probability p is equivalent to:

$$
\begin{aligned}
G_1, G_2 &\overset{iid}{\sim} Gumbel(0,1) \\
W, W^A &= Softmax\left(\frac{\log(p)+G_1}{\tau}, \frac{\log(1-p)+G_2}{\tau}\right) \\
W &= \frac{\exp\{(\log(p)+G_1)/\tau\}}{\exp\{(\log(p)+G_1)/\tau\}+\exp\{(\log(1-p)+G_2)/\tau\}} \\
&= \frac{1}{1+\exp\left\{\frac{(\log(1-p)+G_2)-(\log(p)+G_1)}{\tau}\right\}} \\
&= \frac{1}{1+\exp\left\{\left(G_2-G_1+\log\left(\frac{1-p}{p}\right)\right)/\tau\right\}} \\
&= \frac{1}{1+\exp\left\{\left(Z+\log\left(\frac{1-p}{p}\right)\right)/\tau\right\}}, \text{ where } Z \sim Logistic(0,1),
\end{aligned}
$$

since the difference of two Gumbel$(0,1)$ random variables follows a Logistic$(0,1)$ distribution.

Appendix C

Ito Formula for Jump-Diffusion

The Ito formula for jump-diffusion models given by

$$\begin{aligned} \mathrm{d}f(X(t),t) =& \frac{\partial f(X(t),t)}{\partial t}\mathrm{d}t + \frac{\partial f(X(t),t)}{\partial x}\mathrm{d}X(t) + \frac{1}{2}\sigma^2(X(t),t)\frac{\partial^2 f(X(t),t)}{\partial x^2}\mathrm{d}t \\ &+[f(X(t^-)+\Delta X(t),t) - f(X(t^-),t^-)] \end{aligned} \tag{C.1}$$

from Cont and Tankov (2004).

Appendix D

Filtered Volatility against VIX and CBOE

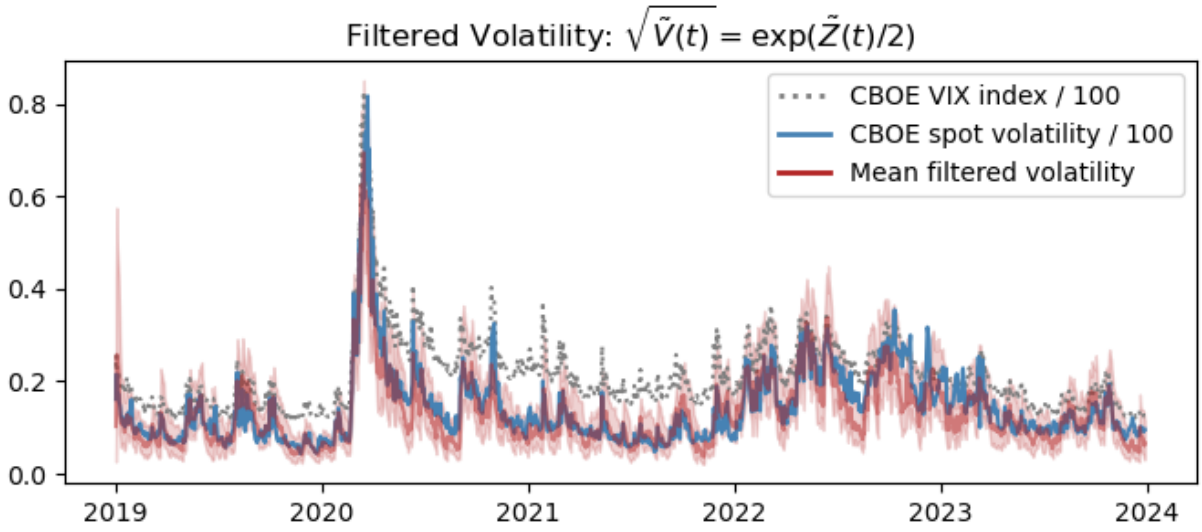

Figure D.1: Filtered volatility with 95% confidence band constructed by particle filtering with parameter estimates in Table 4.8 for the S&P 500 dataset, with VIX and SPOTVOL.

Appendix E

Inference Results of Particle Gibbs

An Adaptive Particle Gibbs (APG) sampler is run on the synthetic dataset in Figure 4.2 with configuration outlined in Table E.1. As the sampler suffers from divergence with a flat prior, a normal prior of $N(0, 50^2)$ was imposed on each transformed parameter for stabilization.

Hyperparameter	Setup
Adaptation rate	0.01
Maximum adaptation	0.1
Initial random walk size	0.001
Number of particles	300
Number of inter-observations	10
Number of MCMC iterations	1,000
Burn-in	100

Table E.1: Configuration for Adaptive Particle Gibbs

With the same Apple M2 Processor for single-seed optimization in Subsection 4.2.2, the APG sampler was completed in approximately 17 minutes. Figure E.1 compares the posteriors from APG with those from the methodology outlined in Section 3.2. With the exception of α, the discrepancy between the two posteriors is prominent in each parameter. The APG posteriors of the diffusion parameters $(\theta, \kappa, \sigma, \rho)$ suggest presence of bias, whereas the jump parameters $(\lambda, \mu_x, \sigma_x, \mu_z)$ show considerable variability.

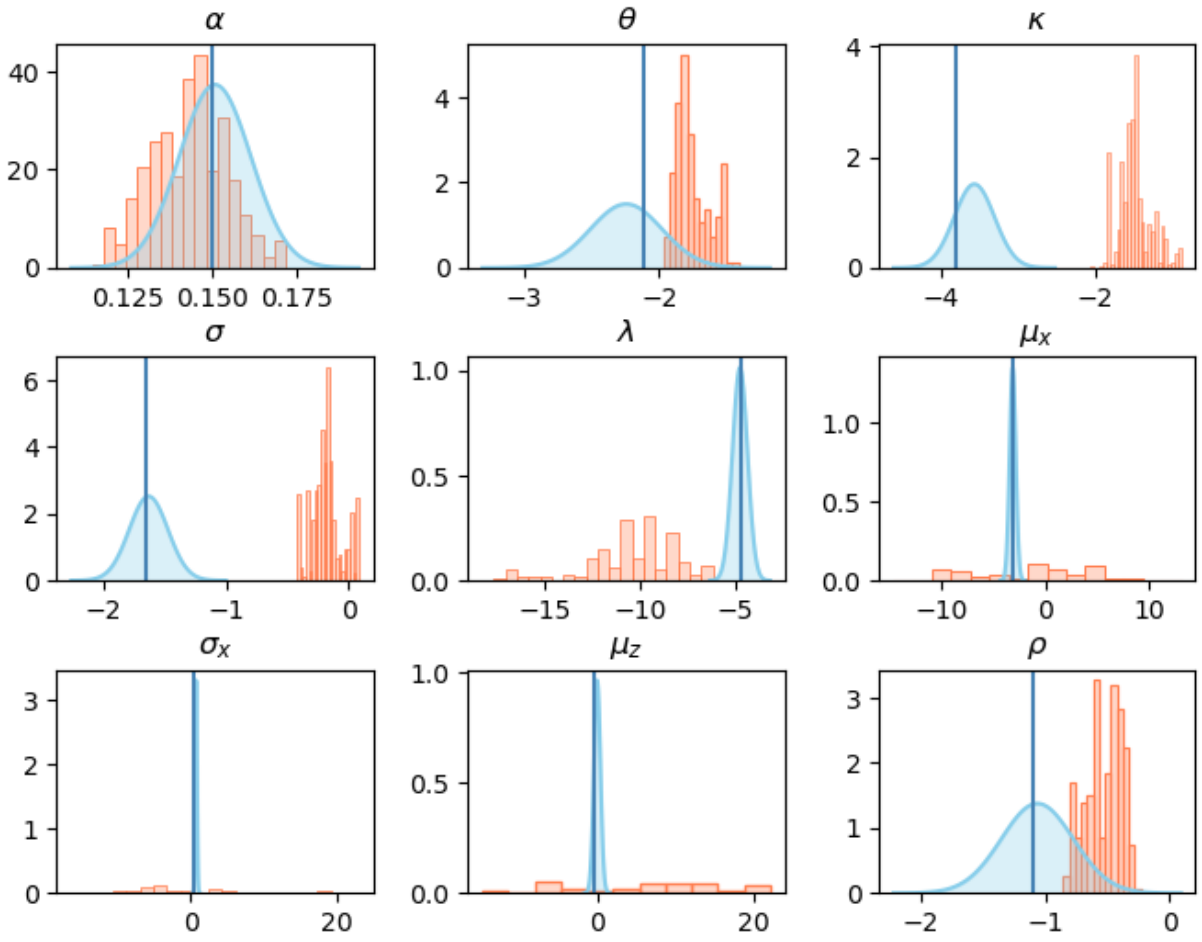

Figure E.1: Parameter posteriors sampled from APG (orange) versus approximate posterior densities in Figure 4.6 (sky blue), with true values in vertical lines (blue).