On the Bayesian Estimation of Jump-Diffusion Models in Finance

by

Louis Arsenault-Mahjoubi

B.A., McGill University, 2019

Project Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Statistics and Actuarial Science
Faculty of Science

SIMON FRASER UNIVERSITY
Summer 2021

Declaration of Committee

Name:	**Louis Arsenault-Mahjoubi**
Degree:	**Master of Science**
Thesis title:	**On the Bayesian Estimation of Jump-Diffusion Models in Finance**
Committee:	**Chair:** Joan Hu Professor, Statistics and Actuarial Science
	Jean-François Bégin Supervisor Assistant Professor, Statistics and Actuarial Science
	Liangliang Wang Committee Member Associate Professor, Statistics and Actuarial Science
	Himchan Jeong Examiner Assistant Professor, Statistics and Actuarial Science

Abstract

The jump-diffusion framework introduced by Duffie et al. (2000) encompasses most one-factor models used in finance. Due to the model complexity of this framework, the particle filter (e.g., Hurn et al., 2015; Jacobs & Liu, 2018) and combinations of Gibbs and Metropolis-Hastings samplers (e.g., Eraker et al., 2003; Eraker, 2004) have been the tools of choice for its estimation. However, Bégin & Boudreault (2020) recently showed that the discrete non-linear filter (DNF) of Kitagawa (1987) can also be used for fast and accurate maximum likelihood estimation of jump-diffusion models. In this project report, we combine the DNF with Markov chain Monte Carlo (MCMC) methods for Bayesian estimation in the spirit of the particle MCMC algorithm of Andrieu et al. (2010). In addition, we show that derivative prices (i.e., European option prices) can be easily included into the DNF's likelihood evaluations, which allows for efficient joint Bayesian estimation.

Keywords: Discrete nonlinear filtering; Bayesian estimation; Particle Markov chain Monte Carlo; Jump-diffusion models; Stochastic volatility.

Dedication

À Lorraine et Miko,
Merci infiniment.

Acknowledgements

I would first like to acknowledge and give many thanks to Dr. Jean-François Bégin. His guidance, research ideas, and helpful suggestions made this project possible. He gave me this project and introduced me to many topics in mathematical finance. I am fortunate to work with him.

I would like to thank Dr. Joan Hu, Dr. Liangliang Wang, and Dr. Himchan Jeong for being a part of the examining committee. Thank you for your time and for reviewing this thesis.

I am grateful for the faculty and staff at the Department of Statistics and Actuarial Science of Simon Fraser University for their hard work and dedication to the students.

Also, I would like to thank the following friends for making life more enjoyable: Maxence, Patrick, Sophie, Siggi, Sonny, Gahyun, Zubia, Ahmad, Peter, and Seyeon.

Finally, I would like to give thanks to my family. They've organized video-conferences throughout the lockdowns, are always encouraging, and are pleasant to talk to.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Nobel prize-winning works of Black & Scholes (1973) and Merton (1973) changed finance by attempting to model the market itself rather than the individual investors' actions (Brine & Poovey, 2017). In doing so, they provide a model for asset returns and a formula for derivative pricing using risk-neutralized dynamics. In this model, the only parameter that cannot be directly observed in the market is volatility. This parameter was assumed to be constant across time and independent of the value of the stock. However, the volatility implied by option prices provides evidence that these assumptions are unrealistic (e.g., the volatility smile or the volatility skew).

Stochastic volatility (SV) models relax these assumptions by introducing a latent process enabling volatility to change over time and to depend on the asset price. Heston (1993) shows that this more complex specification can yield a semi-closed form derivative pricing formula. With this additional model component, more sophisticated parameter estimation methods were needed. Taylor (1986) first tried simple moment-matching methods, which lead to the generalized method of moments of Melino & Turnbull (1990). Monte Carlo methods were also developed (e.g., Danielsson & Richard, 1993; Shephard, 1993; Danielsson, 1994), and filtering methods were used in the quasi-likelihood approaches by Scott (1987), Ruiz (1994), and Harvey et al. (1994). While these rely on the Kalman (1960) filter, some applied the discrete nonlinear filter (DNF) of Kitagawa (1987) to obtain likelihood estimates (e.g., Tanizaki & Mariano, 1994; Fridman & Harris, 1998; Watanabe, 1999; Bartolucci & De Luca, 2001, 2003; Langrock et al., 2012).

Although the above approaches rely on frequentist methodology, researchers also developed Bayesian approaches to account for parameter uncertainty. Jacquier et al. (1994), Kim et al. (1998), and Eraker (2001) use a combination of Gibbs and Metropolis-Hastings samplers in order to perform the Bayesian estimation of SV models.

Empirical evidence, particularly during periods of economic crisis, indicates that SV models do not fully capture the dynamics of asset returns even when adding jumps in asset returns (Bakshi et al., 1997; Bates, 2000; Pan, 2002). As a remedy, Duffie et al. (2000) introduce a general affine framework based on jump-diffusions that incorporates stochastic

volatility, instantaneous variance jumps, and jumps in the asset returns.[1] In addition to introducing this class of models, they show that it yields a semi-closed-form option pricing formula. Jump-diffusion models are flexible enough to replicate many market features and, indeed, include most one-factor models used in finance as special cases.

Using Duffie et al.'s framework, Eraker et al. (2003) and Eraker (2004) provide empirical support for the inclusion of both jump components using an extension of the aforementioned Bayesian methods. The addition of these jump components makes Bayesian estimation computationally cumbersome. For efficient Bayesian inference with state-space models, Andrieu et al. (2010) develop a procedure that combines the particle filter of Gordon et al. (1993) and Markov chain Monte Carlo (MCMC) into the particle MCMC (pMCMC) which was recently applied to jump-diffusion models by Jacobs & Liu (2018).

Johannes et al. (2009) note that there are two optimal filters for the latent factors in this framework: the particle filter and the DNF. Recently, Bégin & Boudreault (2020) apply this alternative filter to high-dimensional jump-diffusion models and show that it provides fast and accurate likelihood evaluations when compared to the particle filter.

In the present study, we explore whether the DNF can replace the particle filter in pMCMC and if its advantages for likelihood evaluations carry over to the Bayesian estimation of jump-diffusion models. In fact, we are able to combine the DNF with Markov chain Monte Carlo (dMCMC) by applying a filter-forward-backward-sampling (FFBS) algorithm similar to the one used by Frühwirth-Schnatter (1994) and Carter & Kohn (1994) for the Kalman filter. Throughout our work, we focus on the stochastic volatility with simultaneous and correlated jumps in instantaneous variance and returns (SVCJ) model. We perform a simulation study to compare the dMCMC to the pMCMC and find that the algorithms obtain similar results. We give empirical applications of the dMCMC to estimate SVCJ model parameters of returns time-series from the TSX composite and S&P 500 indices.

Then, we turn to the problem of incorporating option price data for joint estimation with returns. This is an important issue since joint estimation should improve inference as argued by Renault (1997) and Eraker (2004). We show that options prices can easily be introduced as observables into the likelihood evaluations of the DNF. Therefore, we can perform efficient joint Bayesian estimation with the dMCMC algorithm. To assess the accuracy of dMCMC methods for joint Bayesian estimation, we perform another simulation study. Then, we apply the algorithm to a series of S&P 500 index returns and option prices to obtain estimates for both the physical measure and the risk-neutralized parameters as well as option pricing error estimates for the SVCJ model.

[1]Earlier work has studied the inclusion of jump processes in financial models. Merton (1976) gives a derivative pricing formula when volatility is constant and there are jumps in returns. Bates (1996) derives a semi-closed form solution to option pricing with both stochastic volatility and jumps in returns

The remainder of this project report is organized as follows. In Chapter 2, we present the SVCJ model in continuous-time and in discrete-time under the physical and risk-neutral measures. In Chapter 3, we review the filtering problem, introduce the particle filter and the DNF. Chapter 4 provides an overview of Markov chain Monte Carlo methods before giving the pMCMC and dMCMC algorithms. Chapter 5 discusses option pricing with the SVCJ model and highlights how the DNF can efficiently include options in Bayesian estimation. Lastly, Chapter 6 concludes with a review of the project and suggestions for future work.

Chapter 2

Modelling Framework

2.1 Continuous-Time SVCJ Model

In this section we present the stochastic volatility model with simultaneous correlated jumps in the variance and return dynamics (SVCJ). Duffie et al. (2000) introduce the SVCJ model and provide semi-closed option pricing formula for it. This is an important reason for working with the model as we will later incorporate options in our estimation procedure. The SVCJ relies on three features: a stochastic volatility factor, return jumps, and instantaneous variance jumps. Each of these produces a distinct behaviour in the asset price and volatility dynamics. This is the most complex model that we will work with in the present study.

We first set a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ equipped with a filtration $\mathbb{F} = \{\mathcal{F}_t : t \in [0,T]\}$. Let Y_t be the time-t observed financial asset price and X_t be its time-t latent instantaneous variance for $t \in [0,T]$. Under the physical measure $\mathbb{P}$, the dynamics of the model are given by

$$\frac{dY_t}{Y_{t^-}} = (r_t + \delta_t + \gamma_t - \bar{\alpha}\omega)dt + \sqrt{X_{t^-}}dW_t^Y + d\left(\sum_{n=1}^{N_t}\left(e^{Z_n^Y} - 1\right)\right) \quad \text{and} \tag{2.1}$$

$$dX_t = \kappa(\theta - X_{t^-})dt + \sigma\sqrt{X_{t^-}}dW_t^X + d\left(\sum_{n=1}^{N_t} Z_n^X\right), \tag{2.2}$$

where r_t is the time-t risk-free rate of return in the economy, δ_t is the time-t dividend yield, and γ_t is the total risk premium. For the variance process, θ is the unconditional long-run level, κ is the rate of mean reversion, and σ is the volatility of the variance parameter. The processes $W^Y = \{W_t^Y : 0 \leq t \leq T\}$ and $W^X = \{W_t^X : 0 \leq t \leq T\}$ are two correlated Brownian motions with $\text{Corr}(W_t^X, W_t^Y) = \rho$. The number of jumps up to time t, N_t, follows a Poisson process with arrival rate ω. Each variance jump is exponentially distributed with mean ν (i.e., $Z_n^X \sim \text{Exp}(\nu)$, $\forall n \in \{1, 2, \ldots, N_T\}$ where $\sim$ denotes the distribution of a random variable). The return jumps are Gaussian: $Z_n^Y \sim \mathcal{N}(\alpha + \rho_z Z_n^X, \delta^2)$, $\forall n \in \{1, 2, \ldots, N_T\}$. The jump compensator is $\bar{\alpha}\omega$, where $\bar{\alpha} = \frac{\exp(\alpha + 0.5\delta^2)}{1 - \rho_z \nu} - 1$.

Following the work of Jacobs & Liu (2018), we assume that, under the risk neutral measure $\mathbb{Q}$, the model dynamics are

$$\frac{dY_t}{Y_{t^-}} = (r_t + \delta_t - \bar{\alpha}^{\mathbb{Q}}\omega)dt + \sqrt{X_{t^-}}d\widetilde{W}_t^Y + d\left(\sum_{n=1}^{\widetilde{N}_t}\left(e^{\widetilde{Z}_n^Y} - 1\right)\right) \quad \text{and} \tag{2.3}$$

$$dX_t = \tilde{\kappa}(\tilde{\theta} - X_{t^-})dt + \sigma\sqrt{X_{t^-}}d\widetilde{W}_t^X + d\left(\sum_{n=1}^{\widetilde{N}_t}\widetilde{Z}_n^X\right), \tag{2.4}$$

where $\bar{\alpha}^{\mathbb{Q}}$ is the jump compensator under $\mathbb{Q}$. Assuming that the diffusive variance risk premium is $\eta_x X_t$, we have that $\tilde{\kappa} = \kappa - \eta_x$ and $\tilde{\theta} = \frac{\kappa\theta}{\tilde{\kappa}}$. The return and variance jump risk premiums are assumed to be solely dependent on the average size of the return and variance jumps, respectively, with $\eta_{J^Y} = \alpha - \tilde{\alpha}$ and $\eta_{J^X} = \nu - \tilde{\nu}$. Assuming that the diffusive return risk premium, $\eta_y X_t$, is linear in X_t, we get the following total return risk premium: $\gamma_t = \eta_y X_t + \omega(\bar{\alpha} - \bar{\alpha}^{\mathbb{Q}})$. Lastly, σ and ω remain unaffected by the change of measure.

By removing the jump components (e.g., by setting $\omega = 0$), we obtain the stochastic volatility (SV) model similar to that of Heston (1993). This latter model has the latent variance following a square-root diffusion similar to that used by Cox et al. (1985, CIR hereafter) in the interest rate literature. It also incorporates the leverage effect when ρ is negative. If we let $\nu = 0$, we obtain the stochastic volatility with jumps in the return dynamics (SVYJ) model. The SVYJ model resembles the one studied in Bates (1996). This model often appears with the SVCJ in studies on the impact of jumps in the variance process. Indeed, the addition of jumps in both return and volatility is well supported empirically (see Eraker et al., 2003; Eraker, 2004). Jumps are particularly useful to capture the price dynamics during turbulent market periods. Moreover, having the jumps occur simultaneously in both processes helps with parameter identification. Typically, volatility jumps are harder to identify than jumps in returns. Therefore, having simultaneous jumps provides us with more accurate estimates of the variance jump times from our return jump estimation.

2.2 Discrete-Time SVCJ Model

Since our measurements (e.g., daily returns and option prices) are observed in discrete-time, we need to discretize the SVCJ dynamics. Unfortunately, using a simple Euler scheme gives positive probability to negative variance values. Instead, we use the full truncation (FT) scheme proposed in Lord et al. (2010) to circumvent the issue. In a simulation study, the authors compare the Monte Carlo options prices reached by various simulation schemes against the analytical price. They find that FT demonstrates rapid convergence and low bias across option strike prices when compared to the other schemes for both the SV and SVYJ models. Although the SVCJ model was not included in the study, we should expect similar

performance from FT as jumps in variance are not an additional source of discretization error.

In discrete-time, we only have access to the information in the set $\mathbb{G} = \{\mathcal{G}_j : j = 0, 1, \ldots, N\}$ where we set $hN = T$, with N being the number of observations and h the time between consecutive observations. Moreover, let $y_j = \log\left(\frac{Y_{jh}}{Y_{(j-1)h}}\right)$ be the asset returns at time j. Then, FT yields the following discrete-time dynamics:

$$y_j = \left(r_{j-1} + \delta_{j-1} + \gamma_{j-1} - \bar{\alpha}\omega - \frac{x_{j-1}}{2}\right) h + \sqrt{x_{j-1}h}\,\epsilon_j^y + \sum_{i=1}^{n_j} z_{j,i}^y, \tag{2.5}$$

$$\tilde{x}_j = \tilde{x}_{j-1} + h\kappa\left(\theta - x_{j-1}\right) + \sigma\sqrt{x_{j-1}h}\,\epsilon_j^x + \sum_{i=1}^{n_j} z_{j,i}^x, \text{ and} \tag{2.6}$$

$$x_j = \max(\tilde{x}_j, 0). \tag{2.7}$$

We have that both ϵ_j^x and ϵ_j^y are standard normal random variables with correlation ρ. For the jumps, $n_j \sim \text{Poi}(\omega h)$, $z_{j,i}^x \sim \text{Exp}(\nu)$ and $z_{j,i}^y \sim \mathcal{N}(\alpha + \rho_z z_{j,i}^x, \delta^2)$, $i = 1, 2 \ldots n_N$.

Similar discrete-time dynamics can be found in the works of others. Setting $\rho = 0$ and $\omega = 0$, we have a specification similar to those in Danielsson & Richard (1993); Shephard (1993); Danielsson (1994); Harvey et al. (1994); Jacquier et al. (1994); Ruiz (1994); Kim et al. (1998) and Watanabe (1999), among others. If we set $\nu = 0$, we remove the variance jumps and find model dynamics similar to those investigated in Pitt et al. (2014).

Chapter 3

Volatility Filtering

3.1 Background

3.1.1 State-Space Models

State-space models denote a general class of models that are used in different settings such as signal processing or object tracking. In financial applications, we find them in yield detention, dynamic portfolio betas, and stochastic volatility (Rémillard, 2013). These models describe a situation where a latent stochastic process or hidden state $\mathbf{H} = \{\mathbf{H}_j : j = 1, 2, \ldots, N\}$, is related in some way to the measurement or observed process $\mathbf{Z} = \{\mathbf{Z}_j : j = 1, 2 \ldots, N\}$. Because the latent process $\mathbf{H}$ is a Markov process in most applications, these models are also called hidden Markov models. State-space models can be in continuous-time where we would replace $j \in \mathbb{N}$ by $t \in \mathbb{R}$. The relationship of the hidden state and the observations across time is characterized by

$$\mathbf{Z_j} = g_\Theta(\mathbf{H}_{j-1}, \xi_j^z) \text{ and} \tag{3.1}$$

$$\mathbf{H_j} = f_\Theta(\mathbf{H}_{j-1}, \xi_j^h), \tag{3.2}$$

where ξ_j^h and ξ_j^z are random variables (or random vectors) representing the noise in the system while $g_\Theta(\mathbf{H}, \xi^z)$ and $f_\Theta(\mathbf{H}, \xi^h)$ are the measurement and transition equations, respectively. Lastly, Θ is a vector containing the model parameters.

We can see that the discretized version of the SVCJ model is nested within this framework: the asset returns $\mathbf{Z}_j \equiv y_j$ are the observations. Also, by letting $J^{y_j} \equiv \sum_{i=1}^{n_j} z_{j,i}^y$, we get $\xi_j^z \equiv (\epsilon_j^y, J^{y_j})$ as the measurement noise, and $\Theta = \{r, \alpha, \omega, \rho, \rho_z, \delta, \nu, \sigma, \theta, \kappa\}$ the parameters. From Equation (2.5), we get the measurement equation

$$g_\Theta(x_{j-1}, \xi_j^z) = \left(r_{j-1} + \delta_{j-1} + \gamma_{j-1} - \bar{\alpha}\omega - \frac{x_{j-1}}{2}\right) h + \sqrt{x_{j-1} h}\, \epsilon_j^y + J^{y_j}. \tag{3.3}$$

As the volatility process, $x_{1:N}$, is latent, we set it as the hidden state $\mathbf{H_j} \equiv x_j$, define $J^{x_j} \equiv \sum_{i=1}^{n_j} z_{j,i}^x$, and the transition noise is then $\xi_j^h \equiv (\epsilon_j^x, J^{x_j})$. From Equations (2.6) and

(2.7), we get the following transition equation:

$$f_\Theta(x_{j-1}, \xi_j^h) = \max\left(\tilde{x}_{j-1} + h\kappa\left(\theta - x_{j-1}\right) + \sigma\sqrt{x_{j-1}h}\,\epsilon_j^x + J^{x_j}, 0\right). \tag{3.4}$$

Similarly, the continuous-time counterparts of these equations fall within the continuous-time state-space model family. We will now describe the general problem of estimating the model hidden states for our specific process of interest x_j given N return observations $y_{1:N} \equiv \{y_j : j = 1, \ldots, N\}$.

3.1.2 The Filtering Problem

When working with state-space models, there are three common distributions of interest: the filtering distribution $p(x_j \mid y_{1:j})$, the prediction distribution, $p(x_{j+1} \mid y_{1:j})$, and the smoothing distribution $p(x_j \mid y_{1:N})$. We also seek to obtain likelihood evaluations, $p(y_{1:N})$, as these are essential building blocks in Bayesian inference. All of these distributions and quantities of interest are for a particular vector of parameters Θ. For the sake of conciseness, we will specify the parameter set Θ in the various densities only when multiple distinct sets of parameters are considered.

Filters are algorithms that, given a set of observations $y_{1:N}$ and model parameters Θ, provide us with (or possibly approximations to) the posterior distributions of interest for all $j = 1, \ldots, N$. We will also see that with these distributions, we can obtain a likelihood evaluation $p(y_{1:N})$.

For example, the well-known Kalman filter (Kalman, 1960) is a quick and effective tool to evaluate these distributions on simple state-space models. It gives exact solutions when the underlying dynamics of the process of interest are linear and Gaussian. For the models that we will be working with, these assumptions do not hold.

Although Kalman filter-based approaches have shown promise on some nonlinear stochastic volatility models (see Scott, 1987; Ruiz, 1994; Harvey et al., 1994, for more details), these were instances where the model had no jumps and the noise terms were assumed to be uncorrelated (i.e., $\text{Corr}(\epsilon_t^x, \epsilon_t^y) = 0$). Their approach relies on assuming Gaussianity and yields quasi-maximum likelihood estimators (QMLE). As the SVCJ model dynamics include discontinuities and correlated noise terms, they are more complex than the simple SV models where the Kalman-based QMLE method was used. Thus, the approximations of the distributions of interest would probably be very poor when compared to the true ones.

Another approximate solution to the filtering problem in nonlinear settings is the unscented Kalman filter of Julier & Uhlmann (1997). This filter can be readily implemented on the SV model, but it is only accurate to the third order as it relies on Taylor series expansions. Moreover, Tanizaki & Mariano (1994) argue that the use of Taylor approximations in combination with the Kalman filter faces several theoretical issues; namely, there is no guarantee that the errors have an expected value of zero, that they are uncorrelated with

the latent states, that they are Gaussian, and that they keep the right correlation structure between the transition and the measurement errors.

In the present study we will focus on optimal filters. These are filters that do not target approximations of the distributions of interest, but the distributions themselves. Indeed, an optimal filter could provide us with the distributions of interests at any desired level of precision given enough computational resources. There are two main filters capable of converging to the distributions of interest in highly nonlinear settings: the particle filter (PF) and the DNF (Johannes et al., 2009).

Before introducing these filters, we will look at how the measurement and transition equations relate to the distributions of interest and, ultimately, to the likelihood of the model.

It is standard in filtering problems to assume a given distribution $p(x_0)$ for our initial hidden states (Creal, 2012). From there, we can use recursions through Bayes' theorem to obtain our three target distributions. The prediction density depends on the previous time's filtering density, $p(x_{j-1} \mid y_{1:j-1})$, and the transition density, $p(x_j \mid x_{j-1})$ as we can see from

$$\begin{aligned} p(x_j \mid y_{1:j-1}) &= \int p(x_j, x_{j-1} \mid y_{1:j-1})\, dx_{j-1} \\ &= \int p(x_j \mid x_{j-1})\, p(x_{j-1} \mid y_{1:j-1})\, dx_{j-1}, \end{aligned} \tag{3.5}$$

whereas the filtering density is given by

$$\begin{aligned} p(x_j \mid y_{1:j}) &= \frac{p(x_j, y_j \mid y_{1:j-1})}{p(y_j \mid y_{1:j-1})} \\ &= \frac{p(y_j \mid x_j)\, p(x_j \mid y_{1:j-1})}{p(y_j \mid y_{1:j-1})}. \end{aligned} \tag{3.6}$$

Finally, the smoothing density can be obtained via the following integral:

$$\begin{aligned} p(x_j \mid y_{1:N}) &= \int p(x_{j+1}, x_j \mid y_{1:N})\, dx_{j+1} \\ &= \int p(x_j \mid y_{1:j}, x_{j+1})\, p(x_{j+1} \mid y_{1:N})\, dx_{j+1} \\ &= \int \frac{p(x_{j+1} \mid x_j)\, p(x_j \mid y_{1:j})\, p(x_{j+1} \mid y_{1:N})}{p(x_{j+1} \mid y_{1:j})}\, dx_{j+1} \\ &= p(x_j \mid y_{1:j}) \int \frac{p(x_{j+1} \mid x_j)\, p(x_{j+1} \mid y_{1:N})}{p(x_{j+1} \mid y_{1:j})}\, dx_{j+1}. \end{aligned} \tag{3.7}$$

The smoothing density in Equation (3.7) is a function of the filtering density $p(x_j \mid y_{1:j})$ of Equation (3.6), the transition density $p(x_j \mid x_{j-1})$, the next time step's smoothing density $p(x_{j+1} \mid y_{1:N})$, and the prediction density $p(x_{j+1} \mid y_{1:j})$. Note that we can initialize this recursion using the fact that the time-N smoothing density is equivalent to the time-N filtering density (i.e., $p(x_j \mid y_{1:j}) = p(x_j \mid y_{1:N})$ for $j = N$).

The numerator of the filtering density requires the observation density, $p(y_j \mid x_j)$ and the current step's prediction density $p(x_j \mid y_{1:j-1})$. Using the same densities, the denominator in Equation (3.6) can then be obtained through the integral

$$p(y_j \mid y_{1:j-1}) = \int p(y_j \mid x_j) p(x_j \mid y_{1:j-1}) \, dx_j. \tag{3.8}$$

Computing $p(y_j \mid y_{1:j-1})$ at each time step allows us to evaluate our likelihood function for a set of parameters Θ through the following decomposition:

$$p(y_{1:N}) = p(y_1) \prod_{j=2}^{N} p(y_j \mid y_{1:j-1}). \tag{3.9}$$

As is the case for many state-space models, the integrals in Equations (3.7) and (3.8) cannot be evaluated analytically for the SVCJ model. It is here that the main difference between the PF and the DNF appears. They rely on different approaches to handle these integrals. The PF is based on Monte Carlo integration methods while the DNF utilizes deterministic numerical integration.

3.2 The Particle Filter

3.2.1 Sequential Importance Sampling

The idea of importance sampling (IS) is key to sequential importance sampling (SIS). It is a Monte Carlo integration method that enables us to evaluate

$$\mathbb{E}[h(X)] = \int h(x) \, f(x) \, dx \tag{3.10}$$

for a random variable X, where we know the functional form of the density $f(X)$ but cannot sample directly from the associated distribution. Instead, we sample $X^{(1)}, \ldots, X^{(m)}$ from a proposal density $q(X)$ and weigh each draw by the ratio of the target to the proposal density. The importance weight of the i-th draw is $w^{(i)} \equiv \frac{f(X^{(i)})}{q(X^{(i)})}$, and its normalized importance weight is $\hat{w}^{(i)} \equiv \frac{w^{(i)}}{\sum_{k=1}^{m} w^{(k)}}$ for $i \in \{1, \ldots, m\}$. The proposal should be easy to sample from, have the same support as the target density, and be as close as possible to the target density. We then use

$$\mathbb{E}[h(X)] \approx \frac{1}{m} \sum_{i=1}^{m} \hat{w}^{(i)} \, h(X_i)$$

to obtain an estimator that converges to the quantity of interest as $m \to \infty$. This relies on the importance sampling fundamental identity (Robert & Casella, 2013)

$$\mathbb{E}[h(X)] = \int h(x) \, \frac{f(x)}{q(x)} \, q(x) \, dx.$$

In our case, we are interested in the distribution of the latent variance sequences $X_{1:N}$, and the expectation in Equation (3.10) is taken under the posterior $p(x_{1:N} \mid y_{1:N})$. In this context, SIS is favoured over IS as it targets the joint distribution sequentially using the conditional distributions $p(x_j \mid x_{1:j-1}, y_{1:j})$. This allows us to save time. The SIS also bypasses having to find appropriate N-dimensional proposal distributions (Creal, 2012). SIS's conditional proposals are based on

$$q(x_{1:j} \mid y_{1:j}) = q(x_j \mid x_{1:j-1}, y_{1:j})\, q(x_{1:j-1} \mid y_{1:j-1}), \tag{3.11}$$

where $q(x_{1:j-1} \mid y_{1:j-1})$ is a unit probability mass at the previous time's sampled trajectory. Each path generated in this way is called a particle. The time-j unnormalized weights of the i-th particle can be computed recursively by using the following relationship:

$$\begin{aligned}
w_j^{(i)} &\equiv \frac{p(x_{1:j}^{(i)} \mid y_{1:j})}{q(x_{1:j}^{(i)} \mid y_{1:j})} \\
&\propto \frac{p(x_{1:j}^{(i)}, y_j \mid y_{1:j-1})}{q(x_j^{(i)} \mid x_{1:j-1}, y_{1:j})\, q(x_{1:j-1} \mid y_{1:j-1})} \\
&\propto \frac{p(y_j \mid x_j^{(i)})\, p(x_j \mid x_{j-1})\, p(x_{1:j-1}^{(i)} \mid y_{1:j-1})}{q(x_j^{(i)} \mid x_{1:j-1}^{(i)}, y_{1:j})\, q(x_{1:j-1}^{(i)} \mid y_{1:j-1})} \\
&= w_{j-1}^{(i)} \frac{p(y_j \mid x_j^{(i)})\, p(x_j^{(i)} \mid x_{j-1}^{(i)})}{q(x_j^{(i)} \mid x_{1:j-1}^{(i)}, y_{1:j})} \\
&= w_{j-1}^{(i)}\, \tilde{w}_j^{(i)},
\end{aligned} \tag{3.12}$$

where $\tilde{w}_j^{(i)}$ is the incremental importance weight defined by $\tilde{w}_j^{(i)} \equiv \frac{p(y_j|x_j^{(i)})\, p(x_j|x_{j-1}^{(i)})}{q(x_j^{(i)}|x_{1:j-1}^{(i)}, y_{1:j})}$.

As we have seen in Equation (3.12) above, the importance weights can be written as a product of the incremental importance weights at each time index. Thus, if a proposal is poor and has an incremental weight of zero at time j, its importance weight remains at zero in the following time steps. For a long enough series, this will eventually lead to a single particle with weight one, while all the others will have a weight of zero. This issue is called weight degeneracy. Note that this does not mean that the remaining particle is a good estimate in any absolute sense, only that it is relatively good compared to the other particles (Robert & Casella, 2013). The sequential importance resampling (SIR) adds a resampling step to SIS to resolve the problem of weight degeneracy. We discuss the details of this step in the next subsection.

3.2.2 Sequential Importance Resampling

The SIR algorithm (or particle filter) builds on SIS by adding a resampling step. Resampling tends to eliminate lower weighted particles and increase forward propogation of those with higher weights. Although other resampling schemes have been proposed, we present an SIR algorithm with multinomial resampling performed at each time period in Algorithm 1. This scheme relies on the fact that after computing the particles' normalized weights, we have m pairs $\{x_j^{(i)}, \hat{w}_j^{(i)}\}$, where $\sum_{i=1}^m \hat{w}_j^{(i)} = 1$. With these pairs, we can create a multinomial distribution to sample from with replacement. From this multinomial, we draw particles $x_j^{(i)}$ with probability $\hat{w}_j^{(i)}$. Importantly, resampling at every iteration of the algorithm allows for

Algorithm 1 Sequential Importance Resampling

Draw $x_0^{(i)}$ from initial density $q_0(\cdot)$ and set importance weights $w_0^{(i)} = \frac{p(x_0^{(i)})}{q_0(x_0^{(i)})}$ for each $i \in \{1, \ldots, m\}$.
for $j = 1, 2, \ldots, N$ **do**
    for $i = 1, \ldots, m$ **do**
        Draw $x_j^{(i)}$ from $q(x_j \mid x_{1:j-1}, y_{1:j})$.
        Compute the importance weights $w_j^{(i)} = w_{j-1}^{(i)} \, \tilde{w}_j^{(i)}$.
        Compute the normalized importance weights $\hat{w}_j^{(i)} = \frac{w_j^{(i)}}{\sum_{k=1}^m w_j^{(k)}}$.
    end for
    Resample m particles according to their normalized weights $\hat{w}_j^{(i)}$ for each $i \in \{1, \ldots, m\}$.
    Set $w_j^{(i)} = \frac{1}{m}$ for each $i \in \{1, \ldots, m\}$.
end for

the evaluation of the likelihood using the following recursion:

$$\begin{aligned} p(y_j \mid y_{1:j-1}) &= \int p(y_j \mid x_j) \, p(x_j \mid \ y_{1:j-1}) \, dy_{1:j-1} \\ &\approx \frac{1}{m} \sum_{i=1}^{m} \tilde{w}_j^{(i)}. \end{aligned} \tag{3.13}$$

Now that we have a particle filtering framework, we will look at a specific choice of proposal distribution and how it can be applied to the SVCJ model.

3.2.3 The Bootstrap Filter

The bootstrap filter of Gordon et al. (1993) is a PF that is easy to implement as it uses the transition density as its proposal (i.e., we propose according to $q(x_j \mid x_{1:j-1}, y_{1:j}) \equiv p(x_j \mid x_{j-1})$). Choosing the transition density as the proposal density allows for the following

simplification when computing the incremental importance weights:

$$\tilde{w}_j^{(i)} = \frac{p(y_j \mid x_j^{(i)})\, p(x_j \mid x_{j-1}^{(i)})}{q(x_j^{(i)} \mid x_{1:j-1}^{(i)}, y_{1:j})} = \frac{p(y_j \mid x_j^{(i)})\, p(x_j \mid x_{j-1}^{(i)})}{p(x_j \mid x_{j-1}^{(i)})} = p(y_j \mid x_j^{(i)}). \tag{3.14}$$

As the proposal does not use information from the observations $y_{1:j}$, it is "blind." In the case of the SVCJ model, we can sample sequentially from the transition as stated in Equation (3.4) via iterated conditioning on latent factors

$$\begin{aligned} p(x_j^{(i)} \mid x_{j-1}^{(i)}) &= p(x_j^{(i)} \mid x_{j-1}^{(i)}, n_j^{(i)})\, p(n_j^{(i)}) \\ &= p(x_j^{(i)} \mid x_{j-1}^{(i)}, n_j^{(i)}, J^{x_j^{(i)}})\, p(J^{x_j^{(i)}} \mid n_j^{(i)})\, p(n_j^{(i)}). \end{aligned} \tag{3.15}$$

For each particle, $i \in \{1, \ldots, m\}$, we first draw the number of jumps $n_j^{(i)} \sim \text{Poi}(\omega h)$, then get the sum of variance jumps from a gamma distribution, i.e., $J^{x_j^{(i)}} \mid n_j^{(i)} \sim \Gamma(n_j^{(i)}, \nu)$, where $n_j^{(i)}$ is the shape parameter and ν is the scale parameter. Finally, we can draw the instantaneous variance with the following discretization:

$$x_j^{(i)} \mid x_{j-1}^{(i)}, n_j^{(i)}, J^{x_j^{(i)}} \sim \mathcal{N}\left(x_{j-1}^{(i)} + \kappa(\theta - x_{j-1}^{(i)})h + J^{x_j^{(i)}}, \sigma^2 x_{j-1}^{(i)} h\right). \tag{3.16}$$

We now derive the incremental importance weights for the SVCJ model. Under the bootstrap PF, this means having to evaluate the measurement density given in Equation (3.3). As it depends on the previous volatility, the number of jumps, and the size of the jumps in variance which have been simulated, we are looking for $p(y_j \mid x_j^{(i)}, x_{j-1}^{(i)}, n_j^{(i)}, J^{x_j^{(i)}})$. Assuming that $\tilde{x}_j^{(i)} \geq 0$ (if this is not the case in practice, we set it to zero), from the transition Equation (3.4), we can write

$$\epsilon_j^{x_j^{(i)}} = \frac{x_j^{(i)} - x_{j-1}^{(i)} - h\kappa(\theta - x_{j-1}^{(i)}) - J^{x_j^{(i)}}}{\sigma\sqrt{x_{j-1}^{(i)} h}}. \tag{3.17}$$

Then, using the Cholesky decomposition, we write $\epsilon_j^y = \sqrt{1-\rho^2}\epsilon_j^{\perp} + \rho\epsilon_j^{x_j^{(i)}}$, with $\text{Corr}(\epsilon_{\text{j}}^{\perp}, \epsilon_{\text{j}}^{\text{x}_{\text{j}}^{(\text{i})}}) = 0$. This allows us to get the measurement under the following form:

$$y_j = \left(r_{j-1} + \delta_{j-1} + \gamma_{j-1} - \bar{\alpha}\omega - \frac{x_{j-1}^{(i)}}{2}\right) h + \sqrt{x_{j-1}^{(i)} h}\left(\sqrt{1-\rho^2}\epsilon_j^{\perp} + \rho\epsilon_j^{x_j^{(i)}}\right) + J^{y_j}. \tag{3.18}$$

Conditional on the latent factors produced by the proposal (the number of jumps, the variance jump size, and the instantaneous variances up to time j), every remaining random variable in Equation (3.18) is Gaussian and uncorrelated. Thus, we find that

$$y_j \mid x_j^{(i)}, x_{j-1}^{(i)}, n_j^{(i)}, J^{x_j^{(i)}} \sim N(\mu_j^{(i)}, \sigma_j^{(i)\,2}), \tag{3.19}$$

where

$$\mu_j^{(i)} = \left(r_{j-1} + \delta_{j-1} + \gamma_{j-1} - \bar{\alpha}\omega - \frac{x_{j-1}^{(i)}}{2} \right) h + \sqrt{x_{j-1}^{(i)} h}\, (\rho \epsilon_j^{x_j^{(i)}}) + n\alpha + \rho_z J^{x_j^{(i)}}, \tag{3.20}$$

with the value of $\epsilon_j^{x_j^{(i)}}$ coming from Equation (3.17), and

$$\sigma_j^{(i)\,2} = x_{j-1}^{(i)}(1-\rho^2)h + n\delta^2. \tag{3.21}$$

A popular alternative to the boostrap PF and its blind proposal is the auxiliary particle filter (APF) of Pitt & Shephard (1999). The APF incorporates information from the data into the filter's proposals by resampling before drawing new particles from a joint density that includes an auxiliary variable. The APF is implemented for the SVCJ model in Johannes et al. (2009). For the Bayesian estimation of the SVCJ model, Jacobs & Liu (2018) find that the APF yields more variable likelihood evaluations and is therefore less suitable than the bootstrap filter. Pitt et al. (2014) also opt for the simpler filter arguing that single observations are not particularly informative in the context of stochastic volatility models. Thus, including them in the proposal should not lead to significant benefits. So, we will be using the bootstrap filter in future sections.

We would also like to note in passing that there exists many other extensions to the PF presented here. We will briefly discuss a few of them in the remainder of this subsection.

Doucet et al. (2006) presents a blocked sampling scheme that allows for a higher number of unique particles throughout the series while reducing the number of resampling steps. This solves an issue known as "particle thinning." As resampling implies that we have fewer and fewer unique particles at earlier times, we can get relatively poor approximations of the target distributions for values of j much lower than N. However, the block sampling strategy is computationally costly and requires the selection of a good joint proposal.

Another issue relevant to the PF is the randomness in its likelihood evaluations. Since the PF uses the weights from a stochastic discrete distribution to evaluate likelihood contributions, there can be jumps in the likelihood as a function of the parameter values. Noise in the likelihood evaluations makes its maximization and Bayesian posterior distribution estimation more difficult. Although using more particles can help smooth the likelihood, it can be more effective to use the continuous resampling scheme of Malik & Pitt (2011). Continuous resampling utilizes interpolation between the steps of the empirical distribution function (ECDF) produced by the particles to allow midpoints to be sampled. While most simple sampling schemes (including multinomial resampling) have a computational burden

of $\mathcal{O}(m)$ at each iteration, this scheme has a higher order of operations, i.e., $\mathcal{O}(m \log m)$. The more expensive scheme also introduces a bias of order $\frac{1}{m}$ in the likelihood estimates which gives us a classic bias-variance trade-off situation. In the next section, we will see that we can have a precise likelihood estimate by using a deterministic filter (i.e., with a nil variance).

3.3 The Discrete Nonlinear Filter

3.3.1 Framework

Kitagawa (1987) introduces the discrete nonlinear filter (DNF) to obtain filtering, prediction, and smoothing distributions as well as likelihood evaluations for general state-space models. The method solves the filtering problem by iterated numerical integrations while using the models' transition and measurement equations. This approach has been applied (and improved upon) by Pole & West (1990) in the context of conditionally Gaussian models and by Watanabe (1999), Langrock et al. (2012), and Bégin & Boudreault (2020) for modelling financial assets.

With Equations (3.5), (3.6), (3.8) and (3.9) presented in Section 3.1.2, we can compute all of the filtering and prediction densities of interest using the forward recursion of Algorithm 2.

Algorithm 2 Discrete Nonlinear Filter

1: Initialize by selecting density $p(\mathbf{x}_0)$
2: **for** $j = 1, 2, \ldots, N$ **do**
3: Compute $p(x_j \mid y_{1:j-1})$ by numerical integration using Equation (3.5).
4: Compute $p(x_j \mid y_{1:j})$ via Equation (3.6) and numerical integration of the
5: denominator using Equation (3.8).
6: **end for**
7: Compute $p(y_{1:N})$ using Equation (3.9).

With the filtering densities computed, we can obtain the smoothing densities through backwards recursion using the fact that the time-N filtering density equals the time-N smoothing density and Equation (3.7).

3.3.2 Implementation for the SVCJ Model

The DNF has been applied to stochastic volatility models more complex than the SVCJ model in Bégin & Boudreault (2020). They include a stochastic jump arrival intensity component to get the SVCJSI model (stochastic volatility with correlated jumps in returns and variance with stochastic jump arrival intensity). Following the methodology outlined in this work, we apply the DNF to the simpler SVCJ model.

First, we can perform the prediction step of the DNF (i.e., Step 3 of Algorithm 2) by integrating out the hidden factors

$$\begin{aligned} p(x_j \mid y_{1:j-1}) &= \int_{\mathbb{R}_+} p(x_j \mid x_{j-1})\, p(x_{j-1} \mid y_{1:j-1})\, dx_{j-1} \\ &= \iint_{\mathbb{R}_+^2} p(x_j \mid x_{j-1}, J^{x_j})\, p(x_{j-1} \mid y_{1:j-1})\, p(J^{x_j})\, dJ^{x_j}\, dx_{j-1} \\ &= \sum_{n_j=0}^{\infty} \iint_{\mathbb{R}_+^2} p(x_j \mid x_{j-1}, J^{x_j}, n_j)\, p(J^{x_j} \mid n_j)\, p(n_j) \\ &\qquad\qquad p(x_{j-1} \mid y_{1:j-1})\, dJ^{x_j}\, dx_{j-1}, \end{aligned} \tag{3.24}$$

where $p(x_{j-1} \mid y_{1:j-1})$ is the previous iterations filtering distribution and all other densities are identical to those found in Equations (3.15) and (3.16).

Second, we obtain the likelihood contribution by computing the following:

$$\begin{aligned} p(y_j \mid y_{1:j-1}) &= \int_{\mathbb{R}_+} p(y_j \mid x_j)\, p(x_j \mid y_{1:j-1})\, dx_j \\ &= \iint_{\mathbb{R}_+^2} p(y_j \mid x_j, x_{j-1})\, p(x_j \mid x_{j-1})\, p(x_{j-1} \mid y_{1:j-1})\, dx_j\, dx_{j-1} \\ &= \iiint_{\mathbb{R}_+^3} p(y_j \mid x_j, x_{j-1}, J^{y_j})\, p(J^{y_j}) \\ &\qquad\qquad p(x_j \mid x_{j-1})\, p(x_{j-1} \mid y_{1:j-1})\, dx_j\, dx_{j-1}\, dJ^{y_j} \\ &= \sum_{n_j=0}^{\infty} \iiint_{\mathbb{R}_+^3} p(y_j \mid x_j, x_{j-1}, J^{x_j}, n_j)\, p(J^{x_j} \mid n_j) \\ &\qquad\qquad p(n_j)\, p(x_j \mid x_{j-1}, J^{x_j}, n_j) \\ &\qquad\qquad p(x_{j-1} \mid y_{1:j-1})\, dx_j\, dx_{j-1}\, dJ^{x_j}, \end{aligned} \tag{3.25}$$

where we have the previous iteration's filtering density, $p(x_{j-1} \mid y_{1:j-1})$, the measurement density, $p(y_j \mid x_j, x_{j-1}, J^{x_j}, n_j)$, given by Equation (3.19), and the transition densities found in Equations (3.15) and (3.16).

Finally, we can get the filtering density by using the following equation:

$$\begin{aligned} p(x_j \mid y_{1:j}) &= \sum_{n_j=0}^{\infty} \iint_{\mathbb{R}_+^2} p(x_j, x_{j-1}, J^{x_j}, n_j \mid y_{1:j})\, dx_j\, dx_{j-1}\, dJ^{x_j} \\ &= \frac{1}{p(y_j \mid y_{1:j-1})} \Bigg(\sum_{n_j=0}^{\infty} \iint_{\mathbb{R}_+^2} p(y_j \mid x_j, x_{j-1}, J^{x_j}, n_j) \\ &\qquad\qquad p(x_j, x_{j-1}, J^{x_j}, n_j \mid y_{1:j-1})\, dx_j\, dx_{j-1}\, dJ^{x_j} \Bigg), \end{aligned} \tag{3.26}$$

where $p(x_j, x_{j-1}, J^{x_j}, n_j \mid y_{1:j-1})$ can be decomposed through iterated conditioning into the components of the prediction density in Equation (3.24). Note that if we are only interested

in obtaining likelihood evaluations, $p(y_{1:N})$, we do not need to compute the prediction densities explicitly at each step. We can obtain the likelihood by computing only the time-j filtering density and the likelihood contributions for $j = 1, \ldots, N$ as was done in Bégin & Boudreault (2020).

3.3.3 Numerical Implementation

When performing the numerical integrations needed for the DNF, there are two main considerations: the grid of nodes used (i.e., the points at which we evaluate the densities) and the choice of the integration rule. These choices impact the DNF integration errors. Kitagawa (1987) highlights the three sources of error for the DNF and explains how they are affected by the user's decisions: (1) the errors that are usually associated with numerical methods such as rounding error, (2) the errors that occur from truncating the target density (determined by the range of the node grid), and (3) the errors related to the target density approximation between the nodes. These errors are decided by the integration rule, the node placement, and the number of nodes used.

Both Kitagawa (1987) and Bégin & Boudreault (2020) fix a sequence of nodes to be used for numerical integration throughout the time series. Kitagawa (1987) mentions that the effectiveness of the DNF depends on the careful selection of the node placement. Indeed, more nodes should be found in regions with high nonlinear probability density as they could lead to important approximation errors.

The nodes for discrete latent variables should be set to natural numbers. Therefore, for the number of jumps, we simply use the following grid: $\mathbf{R} = [0, 1, \ldots, R]$.

For continuous latent variables, we can obtain the boundaries of a latent factor H by using

$$E_H \pm \delta_L \sqrt{V_H}, \tag{3.27}$$

where $E_H = \lim_{j\to\infty} \mathbb{E}\left[H_j \mid \mathcal{F}_0\right]$ and $V_H = \lim_{j\to\infty} \mathbb{V}\left[H_j \mid \mathcal{F}_0\right]$ are the long-run expected value and variance of H, respectively. The function δ_L depends on the number of nodes, L. To ensure convergence of the DNF, δ_L must satisfy two conditions. First, we require that the boundaries cover the domain of integration as we increase the number of nodes used. Therefore, we need $\lim_{L\to\infty} \delta_L = \infty$. Second, the space between the nodes needs to go to zero so that we have a finer partition as L increases. It is then necessary that $\lim_{L\to\infty} \frac{\delta_L}{L} = 0$. There are many possible choices of δ_L that satisfy the above conditions. We set $\delta_L = 3 + \log(L)$ as done in Bégin & Boudreault (2020).

The grids for the instantaneous variance and the variance jumps are defined as $\mathbf{X} = \left[x^{(1)}, \ldots, x^{(M)}\right]$ and $\mathbf{J} = \left[J^{(1)}, \ldots, J^{(S)}\right]$, respectively. The grids' boundaries are given by

$$
\begin{aligned}
x^{(1)} &= \max\left(\theta - (3 + \log(M))\sqrt{\frac{0.5\,\theta\,\sigma^2}{\kappa}}, 0\right), \\
x^{(M)} &= \max\left(\theta + (3 + \log(M))\sqrt{\frac{0.5\,\theta\,\sigma^2}{\kappa}}, \sqrt{B_x}\right), \\
J^{(1)} &= \max\left(\nu - (3 + \log(S))\sqrt{\nu^2}, 0\right) \quad \text{and} \\
J^{(S)} &= \max\left(\nu R + (3 + \log(S))\sqrt{R\nu^2}, B_J\right).
\end{aligned}
$$

We then get the sequence of intermediate nodes by using

$$
\begin{aligned}
x^{(i)} &= \sqrt{x^{(1)}} + \left(\frac{i-1}{M-1}\right)\left(\sqrt{x^{(M)}} - \sqrt{x^{(1)}}\right)^2, && i = 2, \ldots, M-1, \\
J^{(l)} &= J^{(1)} + \left(\frac{l-1}{S-1}\right)\left(J^{(S)} - J^{(1)}\right), && l = 2, \ldots, S-1,
\end{aligned}
$$

where the intervals between the variance nodes get larger in i as they are uniformly distributed in the volatility (i.e., square root of the variance). The variance and jump size end points, $x^{(M)}$ and $J^{(S)}$, have lower bounds determined by B_x and B_J in order to guarantee that we have a large enough grid. These values will change depending on the context in which we apply the DNF. Here, we set $B_x = 0.15$ and $B_J = 0.015$

Dynamic grid specification schemes have been proposed by Pole & West (1990) and Watanabe (1999). These methods try to lower the number of nodes needed by dynamically improving their placement. Unlike the fixed methods, dynamic ones incorporate information from the data being filtered to place their nodes. Watanabe (1999) relies on using a non-optimal filter first (e.g., the Kalman filter) to get estimates of the mean and variance for the prediction and filtering distributions: $\hat{\mathbb{E}}[x_j \mid y_{1:j-1}]$, $\hat{\mathbb{E}}[x_j \mid y_{1:j}]$, $\hat{\mathbb{V}}[x_j \mid y_{1:j-1}]$, and $\hat{\mathbb{V}}[x_j \mid y_{1:j}]$. Then, he draws the nodes from the Gaussian mixture, i.e.,

$$
\frac{1}{2}\mathcal{N}\left(\hat{\mathbb{E}}[x_j \mid y_{1:j-1}],\, v\,\hat{\mathbb{V}}[x_j \mid y_{1:j-1}]\right) + \frac{1}{2}\mathcal{N}\left(\hat{\mathbb{E}}[x_j \mid y_{1:j}],\, v\,\hat{\mathbb{V}}[x_j \mid y_{1:j}]\right).
$$

The smoothing distributions obtained by the non-optimal filter can also be added to the Gaussian mixture from which the nodes are drawn. The hyperparameter v is set to 4 in Watanabe (1999) with good results when applied to an SV model without leverage (i.e., similar to our model with $\omega = \rho = 0$).

Pole & West (1990) incorporate dynamic grid updates in the optimal filter itself. At each iteration, they move the grid points using the current grid's filtered mean and variance estimates. They repeat the process until the changes in filtered moments of interest are insignificant. It is unclear how both of these methods would fare when applied to the SVCJ

model and compared to the static grid we define above. We leave this investigation for future research.

We now present several numerical integration rules that have been used in the literature for the DNF. Kitagawa (1987) and Watanabe (1999) both use the trapezoid rule. This rule states that for two functions, f_1 and f_2 for example, we have

$$\int_a^b f_1(x)\, f_2(x)\, dx \approx \frac{1}{2}(b-a)\left[f_1(a)\, f_2(a) + f_1(b)\, f_2(b)\right] \tag{3.28}$$

if a and b are close enough. Pole & West (1990) use a Gauss-Hermite quadrature while Fridman & Harris (1998) opt for the Gauss-Legendre quadrature. A rule based on a midpoint c,

$$\int_a^b f_1(x)\, f_2(x)\, dx \approx (b-a)\, f_1(c)\, f_2(c) \tag{3.29}$$

has also been used in the econometric literature by Bartolucci & De Luca (2001, 2003) (as cited in Langrock et al., 2012). In this study, we use the numerical integration rule from Langrock et al. (2012) and Bégin & Boudreault (2020),

$$\int_a^b f_1(x)\, f_2(x)\, dx \approx f_1(c) \int_a^b f_2(x)\, dx.$$

If f_2 is a probability density function and F_2 is the associated cumulative distribution function, then this rule implies that $\int_a^b f_1(x)\, f_2(x)\, dx \approx f_1(c)(F_2(b) - F_2(a))$.

Using the above nodes as midpoints, we define the intervals for the integrations as in Bégin & Boudreault (2020). That is, we let

$$\mathbf{X}^{(i)} = \left[\frac{x^{(i-1)} + x^{(i)}}{2}, \frac{x^{(i)} + x^{(i+1)}}{2}\right), \quad i = 1, \ldots, M,$$
$$\mathbf{J}^{(l)} = \left[\frac{J^{(l-1)} + J^{(l)}}{2}, \frac{J^{(l)} + J^{(l+1)}}{2}\right), \quad l = 1, \ldots, S,$$

where $x^{(0)} = -x^{(1)}$, $j^{(0)} = -j^{(1)}$ and $x^{(M+1)} = j^{(S+1)} = \infty$.

Combining the chosen integration rule with Equation (3.24), we can approximate the prediction density at each time with

$$\hat{p}(x^{(i_j)} \mid y_{1:j-1}) \approx \sum_{n_j=0}^{R} \sum_{i_{j-1}=1}^{M} \sum_{l_j=1}^{S} p(x_j \in \mathbf{X}^{(i_j)} \mid x^{(i_{j-1})}, J^{(l_j)}, n_j)$$
$$p(J^{x_j} \in \mathbf{J}^{(l_j)} \mid n_j)\, p(n_j) p(x^{(i_{j-1})} \mid y_{1:j-1}).$$

Then, from Equation (3.25), we get the following approximation for the time-j likelihood contribution:

$$\hat{p}(y_j \mid y_{1:j-1}) \approx \sum_{n_j=0}^{R} \sum_{i_j=1}^{M} \sum_{i_{j-1}=1}^{M} \sum_{l_j=1}^{S} p(y_j \mid x^{(i_j)}, x^{(i_{j-1})}, J^{(l_j)}, n_j) \\ p(x_j \in \mathbf{X}^{(i_j)} \mid x^{(i_{j-1})}, J^{(l_j)}, n_j) \\ p(J^{x_j} \in \mathbf{J}^{(l_j)} \mid n_j)\, p(n_j) p(x^{(i_{j-1})} \mid y_{1:j-1}).$$

Lastly, using Equation (3.26), we estimate the filtering density with the following approximation:

$$\hat{p}(x^{(i_j)} \mid y_{1:j}) \approx \frac{1}{\hat{p}(y_j \mid y_{1:j-1})} \sum_{n_j=0}^{R} \sum_{i_{j-1}=1}^{M} \sum_{l_j=1}^{S} p(y_j \mid x^{(i_j)}, x^{(i_{j-1})}, J^{(l_j)}, n_j) \\ p(x_j \in \mathbf{X}^{(i_j)} \mid x^{(i_{j-1})}, J^{(l_j)}, n_j) \\ p(J^{x_j} \in \mathbf{J}^{(l_j)} \mid n_j)\, p(n_j) p(x^{(i_{j-1})} \mid y_{1:j-1}).$$

We see that the operations are of $\mathcal{O}((R+1)M^2S)$ at time j for the likelihood contribution. The order of operations increases rapidly with the number of latent factors in the model. Bégin & Boudreault (2020) perform a simulation study to evaluate the accuracy of the DNF likelihood evaluations and how they compare to the bootstrap particle filter. They found accurate (i.e., below 0.1% mean absolute percentage error or MAPE) likelihood evaluations for the SVCJ model using $R = 1$, M between 50 and 60, and $S = M/2.5$. Moreover, for the same computing budget, the DNF is more accurate and reliable than the basic particle filter. These empirical results hold even for the SVCJSI model which has two additional factors that need to be integrated out. Although the DNF would be uncompetitive at higher dimensions, it is a promising choice of filter for the models we investigate in this report.

Chapter 4

Bayesian Estimation

4.1 Background

4.1.1 Markov Chains

In the Bayesian paradigm, we treat model parameters as random variables. In the context of stochastic volatility models, this means estimating the joint posterior distribution of both the instantaneous variances and the model parameters, $p(x_{1:N}, \Theta \mid y_{1:N})$. As this posterior distribution cannot be obtained in closed form, Markov chain Monte Carlo (MCMC) methods are often used to approximate it (Jacquier et al., 1994; Eraker, 2001; Eraker et al., 2003; Eraker, 2004; Jacobs & Liu, 2018). In this subsection, we will formally introduce Markov chains. After defining Markov chains, we present the properties necessary to understand ergodicity and stationary distributions as these are key concepts for MCMC methods. For a more in-depth presentation of Markov chains on continuous spaces, we refer the reader to Chapter 6 of Robert & Casella (2013) as well as Chapters 2 and 3 of Dobrow (2016) for the discrete case.

Definition 1 (Markov Chain)**.** *A discrete-time Markov chain is a discrete-time stochastic process* $\mathbf{X} = \{\mathbf{X}_j \in \mathcal{X} : j = 1, 2, 3, \ldots\}$*, where*

$$p(\mathbf{X}_{j+1} \mid \mathbf{X}_{1:j}) = p(\mathbf{X}_{j+1} \mid \mathbf{X}_j), \quad \forall j = 1, 2, 3, \ldots \tag{4.1}$$

Equation (4.1) describes the so-called Markov property. It implies that once we know a Markov chain's current state, its history does not give additional information about the next state's distribution. In our case, the domain of the Markov chain, $\mathcal{X}$, is either $\mathbb{R}$ or $\mathbb{R}_+$. To properly deal with Markov chains on measurable spaces, measure theoretic probability theory beyond the scope of this report is necessary. It can be found in Robert & Casella (2013). We hope to provide an intuitive understanding of the concepts being presented.

Definition 2 (Stationary Distribution)**.** *We say that* π *is a stationary distribution for the Markov chain* $\mathbf{X}$ *if*

$$\mathbf{X}_j \sim \pi \rightarrow \mathbf{X}_{j+1} \sim \pi, \tag{4.2}$$

Once a Markov chain reaches a stationary distribution, it is in equilibrium as it will remain in the stationary distribution for all subsequent steps.

Definition 3 (Irreducibility)**.** *We say that a Markov chain* $\mathbf{X}$ *is irreducible if there exists* $j \in \mathbb{N}$ *such that* $p(\mathbf{X}_j \in \mathbf{A} \mid \mathbf{X}_0 = \mathbf{x}_0) > 0$ *for all sets* $\mathbf{A} \in \mathcal{X}$ *such that* $p(\mathbf{X} \in \mathbf{A}) > 0$*.*

Irreducibility ensures that the regions available to the chain are not sensitive to its initial condition, $\mathbf{X_0}$. Indeed, if a set can be reached by an irreducible Markov chain, then it can be reached from any inital value.

Definition 4 (Recurrence)**.** *We say that the Markov chain* $\mathbf{X}$ *is recurrent if:*

(1) $\mathbf{X}$ *is irreducible.*

(2) $\forall \mathbf{A} \in \mathcal{X}$ *such that* $p(\mathbf{X} \in \mathbf{A}) > 0$*, the expected number of times the chains visits* $\mathbf{A}$ *is infinite.*

Informally, we now define the concept of recurrence and transience for the states of Markov chains. A Markov chain's states can be recurrent or transient. If they are recurrent, it will always be possible for the Markov chain to reach these states. If they are not recurrent, they are transient (i.e., the chain might not be able to reach them after some point). If all the states of an irreducible Markov chain are recurrent, then the chain is recurrent. If an irreducible Markov chain is not recurrent, we say that the chain is transient.

There is a stronger concept of recurrence in the literature around Markov chains called Harris recurrence. It requires that the probability that the chain eventually returns to a region is one. This concept is used to ensure that MCMC chains converge for all rather than almost all initial values (Robert & Casella, 2013).

Periodicity is the last concept needed before we can define ergodic Markov chains. A chain is periodic if there is a deterministic pattern in its behaviour. This means that it is somehow constrained in its transitions. The number of steps required for a periodic Markov chains to move from one region to another is a multiple of an integer. The concept of periodicity is essential for MCMC. As we will see, the Metropolis-Hastings algorithm relies on using an aperiodic and irreducible transition density on the same support as its target density as a proposal density in the spirit of an acceptance-rejection method (Smith & Roberts, 1993).

Definition 5 (Ergodicity)**.** *A Markov chain with stationary distribution* π *is ergodic if it is aperiodic, irreducible, and Harris recurrent.*

There exist different types of ergodicity such as uniform ergodicity and geometric ergodicity. We omit the details of these concepts and, again, refer the interested reader to Chapter 6 of Robert & Casella (2013). Ergodic Markov chains are necessary for MCMC methods as they eventually converge (the strength of this convergence depends on the type of ergodicity) to the unique stationary distribution π regardless of initial conditions. The

key idea of MCMC is to use an algorithm that generates an ergodic Markov chain with the target distribution as its stationary distribution.

Definition 6 (Detailed Balance Equation)**.** *A Markov chain and a probability density q satisfy the detailed balance equation if*

$$q(\mathbf{x})p(\mathbf{x}' \mid \mathbf{x}) = q(\mathbf{x}')p(\mathbf{x} \mid \mathbf{x}'), \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \tag{4.3}$$

In other words, the detailed balance equation is satisfied if the probability of the Markov chain being in state $\mathbf{x}$ and moving to state $\mathbf{x}'$ is the same as the probability of being in state $\mathbf{x}'$ and moving to state $\mathbf{x}$ for all pairs of states $\mathbf{x}$ and $\mathbf{x}'$. This concept is useful because if an ergodic Markov chain satisfies Equation (4.3) for some distribution q, then it can be shown that q is its stationary distribution (Robert & Casella, 2013).

4.1.2 Markov Chain Monte Carlo

In this subsection, we introduce two MCMC algorithms: the Metropolis-Hastings sampler (Metropolis et al., 1953; Hastings, 1970) and the adaptive Metropolis-Hastings sampler. In general, a Markov chain Monte Carlo method is an algorithm that generates an ergodic Markov chain with stationary distribution π. We can then take the chain's states as samples to estimate quantities of interest such as the moments of the target distribution. Because the chain does not start in its stationary distribution, we *burn in* (i.e., throw away) the first B observations.

It is common practice to set $B = \frac{G}{5}$, where G is the total number of steps in the Markov chain. So, the algorithm provides us with $\mathbf{X}_{B+1}, \mathbf{X}_{B+2}, \ldots, \mathbf{X}_G \sim \pi$. The states after the burn-in period are fundamentally identical to independent and identically distributed samples from π if the chain has converged. Thanks to central limit theorems and convergence results (Robert & Casella, 2013), we get estimators using

$$\hat{\mathbb{E}}\left[h(\mathbf{X})\right] = \frac{1}{G-B} \sum_{j=B+1}^{G} h(\mathbf{X}_j),$$

where

$$\hat{\mathbb{E}}\left[h(\mathbf{X})\right] \to E(h(\mathbf{X})) \text{ as } G \to \infty,$$

which are similar to classic, naive Monte Carlo estimators. The above estimator converges almost surely (i.e., with probability one).

To ensure that the chain reached its stationary distribution, convergence diagnostic tools such as Gelman-Rubin tests (Gelman & Rubin, 1992), Geweke convergence tests (Geweke, 1992), or trace plots are necessary. To perform Gelman-Rubin diagnostics, we run the same MCMC method on the same data using different starting values and compare the variance between and within the chains. If the chains have converged, there should not be important

differences between those variances. Like other tools that require multiple chains, the speed of convergence gets determined by the slowest chain. Then, the starting values for the chains depend on our prior knowledge of the target distribution (i.e., what are reasonable values for the chain to begin in).

An alternative diagnostic tool is the Geweke convergence test (Geweke, 1992). This convergence test compares the mean at the beginning of the post-burn chain with the mean at the end. If the chain has reached its stationary distribution, the mean should not change significantly. It is standard practice to compare the first 10% with the last 50% of the chain after the burn period. One issue is that the test results can be sensitive to this choice of proportions (Cowles & Carlin, 1996). For example, Cowles et al. (1999) finds less bias by using 25% for the first proportion, although both discard more samples than they theoretically should for their simulation study.

The Geweke convergence test has the computational advantage of being a single-chain method. This allows us to evaluate convergence without having run multiple chains on the same data as with Gelman-Rubin diagnostics. The disadvantage of the single-chain approaches is that they cannot distinguish between a chain that has converged and a chain stuck in a local optima because it has not explored enough of the parameter space. More information about the Gelman-Rubin diagnostic, the Geweke diagnostic, and other convergence tests can be found in Chapter 12 of Robert & Casella (2013).

Trace plots represent an alternative approach. This plot is a useful tool that is not a formal test for convergence. The trace plot shows the path of a particular element of the Markov chain as a function of the MCMC iterations. If the chain gets stuck in a particular region, moves slowly, or does not leave its starting values, it could indicate that there is a problem with the algorithm (Johannes & Polson, 2010). Unfortunately, these tools do not always provide conclusive evidence. Johannes & Polson (2010) recommend the use of simulation studies to verify the accuracy of MCMC methods.

The Metropolis-Hastings Algorithm

The steps of the Metropolis-Hastings sampler are given in Algorithm 3. In short, we begin by drawing a candidate move $\mathbf{x}'$ for the Markov chain conditional on $\mathbf{x}$ from the proposal distribution with density $q(\mathbf{x}' \mid \mathbf{x})$. This distribution is chosen by the user and should ideally be easy to sample from and close to the target. The proposal distribution needs to be aperiodic and irreducible on the same support as the target distribution π for the algorithm to converge (Smith & Roberts, 1993). Fortunately, the aperiodicity and irreducibility conditions are usually satisfied if the Markov chain's transition density has the same support as the target distribution (Chib & Greenberg, 1995).

Then, we accept the j-th proposal with acceptance probability $\alpha(\mathbf{x}'_j, \mathbf{x}_{j-1})$. The ratio $\frac{\pi(\mathbf{x}'_j)}{\pi(\mathbf{x}_{j-1})} \frac{q(\mathbf{x}_j|\mathbf{x}'_{j-1})}{q(\mathbf{x}'_{j-1}|\mathbf{x}_j)}$ in the acceptance probability is sometimes called the Hastings ratio. If a symmetric proposal is used (i.e., $q(\mathbf{x}' \mid \mathbf{x}) = q(\mathbf{x} \mid \mathbf{x}')$, $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$), then the Hastings ratio

Algorithm 3 Metropolis-Hastings

Initialize by selecting $\mathbf{x}_0$.
for $j = 1, 2, \ldots, G$ **do**
    Draw $\mathbf{x}'_j \sim q(\mathbf{x}' \mid \mathbf{x}_{j-1})$ and $u \sim \mathcal{U}(0, 1)$.
    Compute $\alpha(\mathbf{x}'_j, \mathbf{x}_{j-1}) = \min\left\{\frac{\pi(\mathbf{x}'_j)}{\pi(\mathbf{x}_{j-1})}\frac{q(\mathbf{x}_{j-1}|\mathbf{x}'_j)}{q(\mathbf{x}'_j|\mathbf{x}_{j-1})}, 1\right\}$.
    if $u < \alpha(\mathbf{x}'_j, \mathbf{x}_{j-1})$ **then**
        Set $\mathbf{x}_j = \mathbf{x}'_j$.
    else
        Set $\mathbf{x}_j = \mathbf{x}_{j-1}$.
    end if
end for

simplifies to a likelihood ratio as

$$\frac{\pi(\mathbf{x}'_j)}{\pi(\mathbf{x}_{j-1})}\frac{q(\mathbf{x}_{j-1} \mid \mathbf{x}'_j)}{q(\mathbf{x}'_j \mid \mathbf{x}_{j-1})} = \frac{\pi(\mathbf{x}'_j)}{\pi(\mathbf{x}_{j-1})}.$$

We can show that the Metropolis-Hastings algorithm converges to π as it satisfies the detailed balance equation in Equation (4.3). First, note that the probability of the chain going from $\mathbf{x}$ to $\mathbf{x}'$ is the probability that $\mathbf{x}'$ is proposed and accepted (i.e., the transition can be written as $p(\mathbf{x}' \mid \mathbf{x}) = q(\mathbf{x}' \mid \mathbf{x})\,\alpha(\mathbf{x}', \mathbf{x})$). Without loss of generality, assume $\pi(\mathbf{x}')\,q(\mathbf{x}' \mid \mathbf{x}) > \pi(\mathbf{x})\,q(\mathbf{x} \mid \mathbf{x}')$. Then, we have

$$\begin{aligned}
\pi(\mathbf{x})\,p(\mathbf{x}' \mid \mathbf{x}) &= \pi(\mathbf{x})\,q(\mathbf{x}' \mid \mathbf{x})\,\alpha(\mathbf{x}', \mathbf{x}) \\
&= \pi(\mathbf{x})\,q(\mathbf{x}' \mid \mathbf{x})\,\frac{\pi(\mathbf{x}')}{\pi(\mathbf{x})}\frac{q(\mathbf{x} \mid \mathbf{x}')}{q(\mathbf{x}' \mid \mathbf{x})} \\
&= \pi(\mathbf{x}')\,p(\mathbf{x} \mid \mathbf{x}'),
\end{aligned}$$

where the last equality holds because $q(\mathbf{x} \mid \mathbf{x}') = p(\mathbf{x} \mid \mathbf{x}')$ as $\alpha(\mathbf{x}, \mathbf{x}') = 1$. If $\pi(\mathbf{x}')\,q(\mathbf{x}' \mid \mathbf{x}) = \pi(\mathbf{x})\,q(\mathbf{x} \mid \mathbf{x}')$, then $\alpha(\mathbf{x}', \mathbf{x}) = \alpha(\mathbf{x}, \mathbf{x}') = 1$ and detailed balance is immediately implied. Now that it is clear that π is the stationary distribution for the Markov chain produced by the Metropolis-Hastings algorithm, we turn to the issue of choosing the proposal distribution, $q(\mathbf{x}' \mid \mathbf{x})$.

Like standard acceptance-rejection methods, the choice of proposal distribution influences the rate at which the Metropolis-Hastings algorithm converges. As we mentioned earlier, the proposal should be close to the target distribution. But, in some situations—particularly in high dimensions—this can be a difficult problem.

A practical solution is to use a random walk proposal. That is, for a chain at state $\mathbf{x}$, we draw proposals $\mathbf{x}' \sim \mathcal{N}(\mathbf{x}, s^2)$. As this proposal is irreducible on $\mathbb{R}$ and aperiodic, it converges for target distributions that lie on the real numbers. In sum, Gaussian proposals

centred at the current state are symmetric, easy to implement in high dimension by using multivariate Gaussian proposals, and converge for a wide range of target distributions.

Using random-walk Metropolis-Hastings, we might encounter the issue of having a target with a support that is a subset of the real line (e.g., $\mathbb{R}_+$). In these cases, one could simply reject all of the proposals that fall outside the target's support (Johannes & Polson, 2010), propose conditional on the regional constraints (Gelman & Rubin, 1992), or work with a transformation of $\mathbf{x}$.

Another issue with random walk proposals is that they do not converge rapidly. They can only be geometrically ergodic rather than uniformly ergodic according to Mengersen et al. (1996). In this case, the proposal variance s^2 can be seen as a tuning parameter that can be changed to optimize the performance of the algorithm. To avoid an expensive search for good values of s^2, we turn to adaptive MCMC methods.

The Adaptive Metropolis-Hastings Algorithm

The adaptive Metropolis-Hastings algorithm avoids the problem of specifying the covariance matrix of the random walk proposal distribution. Adaptive MCMC methods aim to improve the proposal parameters while they run. We present the adaptive Metropolis-Hastings method given in Roberts & Rosenthal (2009) as this was the approach used in Jacobs & Liu (2018) to estimate the posterior parameter distributions of the SVCJ model. For a d-dimensional target, the proposal $\mathbf{x}'$ from the adaptive Metropolis-Hastings chain at $\mathbf{x}$ is drawn for the first G_0 steps using a fixed initial Gaussian random walk proposal and then using a Gaussian mixture:

$$\mathbf{x}'_j \sim \mathcal{N}\left(\mathbf{x},\, \mathbf{v}_0\, 0.1^2\, \frac{I_d}{d}\right), \quad \text{if } j \leq G_0 \text{ and} \tag{4.4}$$

$$\mathbf{x}'_j \sim (1-\beta)\,\mathcal{N}\left(\mathbf{x},\, 2.38^2\, \frac{\Sigma_{j-1}}{d}\right) + \beta\,\mathcal{N}\left(\mathbf{x},\, \mathbf{v}_j\, 0.1^2\, \frac{I_d}{d}\right), \quad \text{if } G_0 < j \leq G, \tag{4.5}$$

where Σ_{j-1} is the previous iteration's empirical covariance matrix estimate based on the Markov chain's past states and I_d is a d-dimensional identity matrix. The second component of the mixture in Equation (4.5) is fixed to ensure that the variance of the proposal distribution has a lower bound so that the chain does not stop exploring. Following Roberts & Rosenthal (2009), we take $\beta = 0.05$. Variables $\mathbf{v}_0$ and $\mathbf{v}_j$ are d-dimensional vectors to scale the variances of the initial proposal distribution and the fixed component of the Gaussian mixture proposal, respectively.

The user must also choose a period p. At every p steps, we compute Σ_j based on $\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}$ and use it for the next p steps. If $p = G$, we use Σ_0 throughout the proposals and the adaptive algorithm reduces to the standard random-walk Metropolis-Hastings sampler. We present the pseudo-code for the adaptive Metropolis-Hastings sampler in Algorithm 4.

Algorithm 4 Adaptive Metropolis-Hastings

Initialize by selecting $\mathbf{x}_0$, $\mathbf{v}_0$ and G_0.
for $j = 1, 2, \ldots, G$ **do**
    Draw $\mathbf{x}'_j$ from Equation (4.4) if $j \leq G_0$ or Equation (4.5) if $G_0 < j$ and $u \sim \mathcal{U}(0, 1)$.
    Compute $\alpha(\mathbf{x}'_j, \mathbf{x}_{j-1}) = \min\left\{\frac{\pi(\mathbf{x}'_j)}{\pi(\mathbf{x}_{j-1})}, 1\right\}$.
    if $u < \alpha(\mathbf{x}'_j, \mathbf{x}_{j-1})$ **then**
        Set $\mathbf{x}_j = \mathbf{x}'_j$.
    else
        Set $\mathbf{x}_j = \mathbf{x}_{j-1}$.
    end if
    if $j/p \in \mathbb{N}$ and $j > G_0$ **then**
        Set $\Sigma_j = \mathrm{Var}(\mathbf{x}_{1:j})$.
    else
        Set $\Sigma_j = \Sigma_{j-1}$.
    end if
end for

4.2 Particle Markov Chain Monte Carlo

In this section we describe a popular particle Markov chain Monte Carlo (pMCMC) method presented in Andrieu et al. (2010): the particle marginal Metropolis-Hastings sampler. The algorithm combines the particle filter with MCMC to allow for the Bayesian estimation of state-space models. The particle filter is used to obtain proposals of the latent variables.

The first use of pMCMC methods can be found in the economics literature around dynamic stochastic general equilibrium (DSGE) models (An & Schorfheide, 2007; Fernández-Villaverde & Rubio-Ramírez, 2007). They use the particle filter to obtain approximate likelihood evaluations of complex DSGE models within a random-walk Metropolis-Hastings sampler. Andrieu et al. (2010) provide theoretical validity to this approach by proving that the algorithm converges to the joint posterior of the latent variables and parameters under mild regularity conditions. pMCMC methods have been since applied to many areas outside of DSGE models as they are suitable in any state-space context. In this report, we are especially interested in their use for the SVCJ model (see Jacobs & Liu, 2018).

Let $\mathbf{x}_{1:N}$ be the sequence of latent variables, $\mathbf{y}_{1:N}$ be the observations, and Θ be a vector containing the parameters of the state-space model. The pMCMC algorithm relies on decomposing the target distribution as follows:

$$p(\mathbf{x}_{1:N}, \Theta \mid \mathbf{y}_{1:N}) = p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N}, \Theta)\, p(\Theta \mid \mathbf{y}_{1:N}). \tag{4.6}$$

A similar decomposition can be applied to the proposal distribution in the particle marginal Metropolis-Hastings sampler. If the Markov chain of interest is $\{\mathbf{x}_{1:N}, \Theta\}$, then Andrieu et

al. (2010) suggest using a proposal of the form

$$q(\mathbf{x}'_{1:N},\, \Theta' \mid \mathbf{x}_{1:N},\, \Theta,\, \mathbf{y}_{1:N}) = q(\mathbf{x}'_{1:N} \mid \mathbf{y}_{1:N},\, \Theta')\, q(\Theta' \mid \Theta).$$

This proposal is performed in two steps: (1) given that the current value of the parameters in the chain is at Θ, sample a new set of parameters using a user-chosen proposal distribution $q(\Theta' \mid \Theta)$, and (2) sample from the latent variable space using the particle filter conditional on the sampled parameter vector Θ'.

We can sample $\mathbf{x}'_{1:N}$ using a similar rationale as that of the multinomial sampling done at every step of SIR in Algorithm 1. The particle filter provides us with m (the total number of particles) pairs of particles and their normalized weights $(\mathbf{x}^{(i)}_{1:N}, \hat{w}^{(i)}_N)$ for each $i = 1, ..., m$. Since $\sum_{i=1}^{m} \hat{w}^{(i)}_N = 1$, we can draw a specific $\mathbf{x}^{(i)}_{1:N}$ with probability $\hat{w}^{(i)}_N$. A naive application of the Metropolis-Hastings theory explained in Section 4.1.2 gives the following Hastings ratio:

$$\frac{p(\mathbf{x}'_{1:N} \mid \mathbf{y}_{1:N},\, \Theta')\, p(\Theta')\, q(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N},\, \Theta)\, q(\Theta \mid \Theta')}{p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N},\, \Theta)\, p(\Theta)\, q(\mathbf{x}'_{1:N} \mid \mathbf{y}_{1:N},\, \Theta')\, q(\Theta' \mid \Theta)},$$

where $p(\Theta)$ is the parameter prior distribution. To compute $q(\mathbf{x}'_{1:N} \mid \mathbf{y}_{1:N},\, \Theta')$, the probability of proposing a particular path with the particle filter, we marginalize out all possible paths that are sampled. We cannot do this directly as this density is generally intractable. Fortunately, this issue can be solved by defining an extended proposal distribution that includes the particles generated at every time j, the parent of the particles, and their entire trajectory as auxiliary variables. These variables have statistical properties that can be precisely defined through auxiliary distributions. With the joint density for all of the components necessary in generating a particle's trajectory, Andrieu et al. (2010) show that we can use the likelihood $p(\mathbf{y}_{1:N} \mid \Theta')$. Then, the acceptance probability can be written as

$$\alpha(\{\mathbf{x}'_{1:N}, \Theta'\},\, \{\mathbf{x}_{1:N}, \Theta\}) = \min\left[1,\, \frac{p(\mathbf{y}_{1:N} \mid \Theta')\, p(\Theta')\, q(\Theta \mid \Theta')}{p(\mathbf{y}_{1:N} \mid \Theta)\, p(\Theta)\, q(\Theta' \mid \Theta)}\right] \tag{4.7}$$

where the likelihood evaluations $p(\mathbf{y}_{1:N} \mid \Theta)$ and $p(\mathbf{y}_{1:N} \mid \Theta')$ can be approximated using Equations (3.8) and (3.13).

The particle marginal Metropolis-Hastings algorithm is presented below in Algorithm 5 where we use $\mathbf{x} \equiv \mathbf{x}_{1:N}$ to reduce the notational burden. Unlike standard SMC estimation of the latent variables' posterior distributions, pMCMC methods are less liable to suffer from particle thinning when estimating the latent states (see Section 3.2.2). Because we sample a full particle path for each accepted proposal, we almost surely obtain a unique particle throughout $j = 1, \ldots, N$ for each acceptance.

Andrieu et al. (2010) show that this algorithm converges to the target posterior distribution under mild regularity conditions despite using approximate likelihood evaluations from the particle filter. Similarly to the Metropolis-Hastings algorithm, the proposal distribution along the parameter space should be aperiodic and irreducible on the same support as the

Algorithm 5 Particle Marginal Metropolis-Hastings

1: Initialize by selecting $\mathbf{x}_0$ and Θ_0.
2: **for** $j = 1, 2, \ldots, G$ **do**
3: Draw $\Theta'_j \sim q(\Theta' \mid \Theta_{j-1})$ and $u \sim \mathcal{U}(0, 1)$.
4: Run the particle filter with Θ'_j.
5: Compute approximate likelihoods $\hat{p}(\mathbf{y}_{1:N} \mid \Theta'_j)$ using Equations (3.8) and (3.13).
6: Estimate $\alpha(\{\mathbf{x}'_j, \Theta'_j\}, \{\mathbf{x}_{j-1}, \Theta_{j-1}\})$ with Equation (4.7) and $\hat{p}(\mathbf{y}_{1:N} \mid \Theta'_j)$ from Step
7: 5.
8: **if** $u < \alpha(\{\mathbf{x}'_j, \Theta'_j\}, \{\mathbf{x}_{j-1}, \Theta_{j-1}\})$ **then**
9: Sample $\mathbf{x}'_j$ from the particles in Step 4 according to their time-N normalized
10: weights.
11: Set $\mathbf{x}_j = \mathbf{x}'_j$ and $\Theta_j = \Theta'_j$.
12: **else**
13: Set $\mathbf{x}_j = \mathbf{x}_{j-1}$ and $\Theta_j = \Theta_{j-1}$.
14: **end if**
15: **end for**

target distribution. Moreover, the estimation of the likelihood function should be unbiased for the convergence to the target distribution to be theoretically guaranteed. Many sampling schemes are unbiased (e.g., multinomial or stratified), but the continuous resampling scheme of Malik & Pitt (2011) for instance is not. As the bias of the scheme is inversely related to the number of particles used, it is unclear how it would perform. In practice, the bias can be made arbitrarily small by increasing the number of particles.

When implementing pMCMC methods, the number of particles in the filter is an important tuning parameter. As we perform an SMC step every time we sample new parameters, using a large number of particles substantially increases the computational burden of the algorithm. Nonetheless, a large number of particles reduces the variance of the likelihood evaluations, which can speed up the convergence of the Markov chain. Thus, there is a trade-off between the speed of an iteration and the precision of its likelihood estimate. For our simulation study, we are able to run an adaptive particle marginal Metropolis-Hastings sampler with 10,000 iterations with 10,000 particles per bootstrap filter in under 48 hours.

Other fine-tuning options that could impact the algorithm performance are: (1) the parameter proposal and prior distributions and (2) the use of another particle filter than the bootstrap filter (e.g., different proposals distributions for the particle generation or sampling schemes within the filter).

In sum, pMCMC methods use a particle filter to propose latent variable samples in a very efficient way. This allows for efficient Bayesian estimation of complex state-space models. Nonetheless, the evaluation of the acceptance probability in Equation (4.7) depends on the particle filter's noisy likelihood estimates and can be computationally intensive.

4.3 DNF-Markov Chain Monte Carlo

In this section, we introduce a discrete nonlinear filter Markov chain Monte Carlo (dM-CMC) method. The algorithm uses the DNF as a proposal within a MCMC method as pMCMC utilizes the particle filter. Algorithm 5 shows that pMCMC uses the particle filter in two different steps: (1) to obtain likelihood estimates $\hat{p}(\mathbf{y}_{1:N} \mid \Theta'_j)$ for the iteration j proposed parameters Θ'_j in Step 5 and (2) to sample $\mathbf{x}'_{1:N} \sim p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N}, \Theta'_j)$, the joint smoothing distribution in Step 9. The DNF given in Algorithm 2 yields an estimate of the likelihood, but does not provide us with the samples from the joint smoothing distribution directly. In this section, we develop a recursive algorithm to sample from the joint smoothing distribution constructed from the DNF.

4.3.1 Forward Filtering Backward Sampling for the DNF

Forward-filtering-backward-sampling (FFBS) algorithms have been used to sample from the joint smoothing distribution of various state-space models. As the name suggests, FFBS relies on first running a filter on the data to obtain some quantities or distributions, and on a backward recursion to sample from $p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N}, \Theta)$ using those quantities.

The first uses of the FFBS algorithm can found Carter & Kohn (1994) and Frühwirth-Schnatter (1994), where the Kalman filter is applied to sample from the joint smoothing distribution within an MCMC method to perform the Bayesian estimation of certain state-space models. The Kalman filter is used to determine the first two moments of the filtering densities. Then, the FFBS algorithm uses these moments to recursively sample the joint smoothing distribution.

We now introduce the sampling recursion and demonstrate how it can be carried out in the context of the DNF. All distributions in the FFBS algorithm are conditional on a particular set of parameters, Θ, which we omit for the remainder of this section. To obtain the FFBS recursion for the DNF, we start by conditioning $p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N})$ on the last state, $\mathbf{x}_N$:

$$\begin{aligned} p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N}) &= p(\mathbf{x}_{1:N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N})\, p(\mathbf{x}_N \mid \mathbf{y}_{1:N}) \\ &= p(\mathbf{x}_{1:N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N-1})\, p(\mathbf{x}_N \mid \mathbf{y}_{1:N}). \end{aligned}$$

To initiate the FFBS algorithm, we sample from the time-N filtering distribution, $p(\mathbf{x}_N \mid \mathbf{y}_{1:N})$, which is given by the DNF. Then, conditioning the first term, $p(\mathbf{x}_{1:N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N-1})$, on $\mathbf{x}_{N-1}$, we have:

$$\begin{aligned} p(\mathbf{x}_{1:N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N-1}) &= p(\mathbf{x}_{1:N-2} \mid \mathbf{x}_{N-1:N}, \mathbf{y}_{1:N-2})\, p(\mathbf{x}_{N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N-1}) \\ &= p(\mathbf{x}_{1:N-2} \mid \mathbf{x}_{N-1}, \mathbf{y}_{1:N-2})\, p(\mathbf{x}_{N-1} \mid \mathbf{x}_N, \mathbf{y}_{1:N-1}). \end{aligned}$$

If we continue conditioning the first term on $\mathbf{x}_{N-2}$, $\mathbf{x}_{N-3}$, $\ldots$, $\mathbf{x}_1$, we obtain:

$$p(\mathbf{x}_{1:N} \mid \mathbf{y}_{1:N}) = p(\mathbf{x}_N \mid \mathbf{y}_{1:N}) \prod_{j=1}^{N-1} p(\mathbf{x}_j \mid \mathbf{x}_{j+1}, \mathbf{y}_{1:j}). \tag{4.8}$$

With Bayes' theorem, each $p(\mathbf{x}_j \mid \mathbf{x}_{j+1}, \mathbf{y}_{1:j})$ term in the product of Equation (4.8) can be written as:

$$\begin{aligned} p(\mathbf{x}_j \mid \mathbf{x}_{j+1}, \mathbf{y}_{1:j}) &= \frac{p(\mathbf{x}_{j+1} \mid \mathbf{x}_j, \mathbf{y}_{1:j})\, p(\mathbf{x}_j \mid \mathbf{y}_{1:j})}{p(\mathbf{x}_{j+1} \mid \mathbf{y}_{1:j})} \\ &= \frac{p(\mathbf{x}_{j+1} \mid \mathbf{x}_j)\, p(\mathbf{x}_j \mid \mathbf{y}_{1:j})}{p(\mathbf{x}_{j+1} \mid \mathbf{y}_{1:j})} \\ &\propto p(\mathbf{x}_{j+1} \mid \mathbf{x}_j)\, p(\mathbf{x}_j \mid \mathbf{y}_{1:j}), \end{aligned}$$

where the filtering distribution $p(\mathbf{x}_j \mid \mathbf{y}_{1:j})$ is computed from the DNF and the transition density $p(\mathbf{x}_{j+1} \mid \mathbf{x}_j)$ is specified by the state-space model. The transition density should be computed for all possible pairs of M latent variable nodes that are used in the DNF. Thus, the sampling is of $\mathcal{O}(M^2)$, which is relatively inexpensive when compared to the computational cost of filtering. For the SVCJ model, the transition density can be evaluated using Equations (3.15) and (3.16).

Algorithm 6 Forward Filtering Backward Sampling for the DNF

1: Run the DNF from Algorithm 2 to obtain the filtering distributions for all $j = 1, \ldots, N$.
2: Sample a node $\mathbf{x}_N$ in $\mathbf{X}$ from a multinomial distribution by normalizing the filtering density $p(\mathbf{x}_N \mid \mathbf{y}_{1:N})$ obtained from the DNF.
3: Compute the transition densities $p(\mathbf{x}' \mid \mathbf{x})$ for all $\mathbf{x}', \mathbf{x} \in \mathbf{X}$
4: **for** $j = N-1, N-2, \ldots, 1$ **do**
5: Compute $p_j \equiv p(\mathbf{x}_{j+1} \mid \mathbf{x}_j)\, p(\mathbf{x}_j \mid \mathbf{y}_{1:j})$ for all $\mathbf{x}_j \in \mathbf{X}$.
6: Sample a node $\mathbf{x}_j$ in $\mathbf{X}$ from a multinomial with probabilities given by normalizing
7: p_j.
8: **end for**

The steps to perform FFBS with the DNF are given in Algorithm 6. We can then slightly modify the particle marginal Metropolis-Hastings algorithm to perform dMCMC. Most steps are identical to those in Algorithm 5, but we run the DNF instead of the particle filter in Step 5 and sample from the joint smoothing distribution with FFBS in Step 9.

Directly using the FFBS in combination with the DNF could result in noticeably discrete sample paths when using a small number of nodes. For example, the posterior instantaneous variance sample can only take 50 different values if $M = 50$. This is less of an issue over the course of the dMCMC since we almost surely use a different set of M nodes at for each parameter combination. Thus, the posterior instantaneous variance sample can take different sets of M values at each accepted proposal. For less crude samples, a methodology

similar to the continuous sampling scheme in essence could be implemented by interpolating between the nodes as in Malik & Pitt (2011).

With dMCMC, we get an algorithm similar to the particle marginal Metropolis-Hastings algorithm, but with deterministic likelihood evaluations. These evaluations have an error that can be made arbitrarily small and can reach high accuracy at relatively low computational costs (Bégin & Boudreault, 2020). As the likelihood is a key component in the particle marginal Metropolis-Hastings, we believe that this can have a significant impact on the computational costs needed for Bayesian estimation in the context of jump-diffusion models.

4.4 Simulation Study

In this section, we perform a simulation study to compare the dMCMC to the pMCMC in the Bayesian estimation of the SVCJ model. We generate 100 paths of 2,520 observations (i.e., ten-year series) from the SVCJ model with a realistic set of parameter values. These parameters are given in the first column of Table 4.1. They are based on the SVCJ maximum likelihood parameter estimates of Bégin & Boudreault (2020) for the S&P 500 index daily returns from January 1990 to September 2018.

We assume the risk-free rate of return r_j to be constant and equal to 1% annually. We also assume that $\rho_z = 0$ and that it is a known constant as this parameter is difficult to estimate when using exclusively returns data (Johannes et al., 2009). We relax this last assumption in Section 5.3, where we perform a simulation study with both returns and option prices. We assume that $\bar{\alpha} - \bar{\alpha}^{\mathbb{Q}} = 0$ throughout the returns-only estimation for reasons of parameter identifiability.

The pMCMC algorithm uses the bootstrap filter described in Section 3.2.3 with 10,000 particles while the dMCMC uses a DNF with $M = 50$, $S = 20$, and $R = 1$. Both methods use the adaptive Metropolis-Hastings sampler described in Algorithm 4 to generate parameter proposals. We update the covariance matrix estimate of these proposals every $p = 100$ iterations after the first $G_0 = 500$ iterations. Recall that, during these first G_0 iterations, we use a diagonal matrix with initial proposal variances $\mathbf{v}_0$ to sample new parameter sets. The initial parameters in the Markov chain are drawn from a Gaussian distribution with a mean equal to the true parameters and a standard deviation that is 10% of the true parameter's magnitude. The initial standard deviation of the parameter proposals $\sqrt{\mathbf{v}_0}$ is the absolute value of 10% of the initial parameter values. We set the mixture component's scale vector $\mathbf{v}_j = \mathbf{v}_0$, for $j = 1, \ldots, N$. We run the adaptive Metropolis-Hastings for $G = 10{,}000$ iterations and burn the first $B = 2{,}000$ iterations. The pMCMC algorithm requires roughly 30 hours to run, while the dMCMC is 20% faster.

The SVCJ prior distributions are set to $\eta_y \sim \Gamma(3, 1)$, $\alpha \sim \mathcal{N}(0, 1)$, $\delta \sim \mathcal{N}(0, 1)$, $\rho \sim \mathcal{U}(-1, 1)$, $\omega \sim \Gamma(5, 1)$, and $\nu \sim \Gamma(1.25, 0.01)$, while κ, θ, and σ are uniform over the positive

Table 4.1: Posterior parameter mean comparison.

Parameter	**True Value**	**pMCMC Mean** (RMSE)	**dMCMC Mean** (RMSE)
η_y	3.000	2.823	2.712
		(0.780)	(0.748)
ρ	-0.745	-0.750	-0.742
		(0.043)	(0.042)
$100\,\theta$	3.200	2.852	2.972
		(0.603)	(0.503)
σ	0.446	0.401	0.399
		(0.052)	(0.054)
κ	3.689	3.876	3.829
		(0.831)	(0.640)
ω	5.125	3.914	3.394
		(1.361)	(1.807)
$1000\,\nu$	4.000	6.467	6.202
		(3.544)	(3.251)
$1000\,\delta$	3.000	7.164	7.614
		(5.10)	(5.54)
$1000\,\alpha$	-7.000	-3.962	-2.548
		(5.151)	(6.490)

real line. As it is difficult to distinguish between few large jumps and frequent small jumps, we choose prior distributions that favour the former as done in Eraker et al. (2003) and Jacobs & Liu (2018).

The average posterior parameter means and their root-mean-square errors (RMSEs) are given in the second and third columns of Table 4.1. We can see that the dMCMC algorithm has a similar performance to the pMCMC algorithm. Specifically, the dMCMC leads to comparable or lower RMSEs for the parameters that are not associated with jumps; that is, η_y, ρ, θ, σ, and κ. However, the dMCMC RMSEs are higher for the all jump parameters besides ν. These parameters are known to be more difficult to estimate as we expect to have only 51.25 jumps for the whole ten year time series used, on average. Moreover, it is often difficult to distinguish between fewer large jumps and more frequent smaller jumps as we mentioned above. The median values and modes for the parameter posterior distributions are given in Table A.1 and Table A.2 of Appendix A, respectively. These statistics report a similar general pattern than that obtained with posterior means.

The dMCMC simulations have a proposal acceptance probability of 14.7%, on average, while the pMCMC average acceptance probability is 14.2%.

Table 4.2 gives the percentage of rejections for the Geweke convergence tests for each parameter. According to the Geweke tests, the dMCMC converged less often than the pMCMC for the jump parameters. For the rest of the parameters, it has similar convergence

Table 4.2: Geweke convergence test rejection in percentage.

Parameter	**pMCMC 95%** (99%)	**dMCMC 95%** (99%)
η_y	14%	8%
	(7%)	(5%)
ρ	9%	10%
	(5%)	(5%)
θ	18%	16%
	(7%)	(6%)
σ	12%	7%
	(3%)	(3%)
κ	12%	19%
	(4%)	(9%)
ω	14%	20%
	(5%)	(10%)
ν	18%	24%
	(11%)	(8%)
δ	18%	24%
	(11%)	(17%)
α	13%	15%
	(4%)	(6%)

percentages with the exceptions of κ and η_y. The results also tell us that more iterations should be used to ensure that the chains attain the stationary distribution for each parameter. Finally, the dMCMC performs well in terms of RMSE for parameters with high convergence rates. This suggests that when the algorithm converges, it reaches a similar distribution to the pMCMC.

Figures C.1 and C.2 in Appendix C give the posterior mean volatilities for a particular path along with 95% confidence intervals (CI) around these means for each method. Both algorithms were run for 10,000 iterations and the first 2,000 were burned. Although parameter estimates may differ, the estimated instantaneous variance trajectories display similar trends. The pMCMC has wigglier CIs which can be explained by the discrete nature of the sampling of the dMCMC.

4.5 Empirical Application

In this section, we apply the dMCMC algorithm to estimate the parameters of the SVCJ model using market data. We use returns from the TSX Composite index between January 2005 and February 2021 and returns from the S&P 500 index from January 2000 to December 2019 (excluding dividends). Both returns series are plotted in Appendix B. The dMCMC has the same tuning parameter settings and priors as in Section 4.4, and uses

$G = 50{,}000$ iterations. Additionally, we estimate ρ_z and following Eraker et al. (2003), we set the prior for this parameter to $\rho_z \sim N(0, 4)$. Table 4.3 provides the posterior parameter means and standard errors for each series. All parameters have the expected sign for both. We obtain an acceptance probability of about 10% for the TSX composite index and 15%

Table 4.3: Posterior parameter mean comparison.

Parameter	**TSX Composite** (Standard error)	**S&P 500** (Standard error)
η_y	0.875	0.797
	(0.483)	(0.418)
ρ	-0.634	-0.813
	(0.049)	(0.024)
$100\,\theta$	4.186	3.862
	(0.481)	(0.367)
σ	0.282	0.448
	(0.024)	(0.025)
κ	1.574	3.275
	(0.283)	(0.483)
ω	0.817	4.290
	(0.384)	(1.450)
$1000\,\nu$	38.640	2.976
	(15.268)	(1.677)
$1000\,\delta$	48.088	3.167
	(62.639)	(2.344)
$1000\,\alpha$	-48.289	-9.734
	(67.093)	(3.599)
ρ_z	-0.021	-1.474
	(0.782)	(1.138)

for the S&P 500 index's. For the TSX, the Geweke convergence test rejects both α and ρ_z at the 1% level. For the S&P 500, convergence in κ is rejected at the 1% level, while θ and ρ_z are rejected at the 5% level. We provide trace plots for these parameter in Appendix D. The trace plots for κ and θ do not appear to be too problematic as they have relatively stable means and variances. Moreover, ρ_z does not converge for both series and this is not surprising as this parameter is difficult to estimate.

For the TSX Composite series, we find much fewer jumps, but they are significantly more important than those found with the S&P 500 index. The TSX jump parameter values resemble those found in Jacobs & Liu (2018) for the S&P 500 index (given in the first column of Table 5.2). This shows that our prior distributions on ω, ν, δ and α do not overly constrain these parameters. We find similar diffusive equity risk premiums η_y for both indices. Moreover, the S&P 500 parameter estimates are aligned with previous studies using the SVCJ model (summary tables can be found in Jacobs & Liu, 2018).

Chapter 5

Joint Bayesian Estimation with Returns and Options Data

Including options as observables in econometric models has been a long-standing goal of researchers in the field (Renault, 1997). These contracts contain additional information about the underlying asset dynamics and its latent factors that is not found in return time series. Eraker (2004) argues that there are three main benefits of including options: (1) we can estimate both risk neutral and physical parameters, (2) we should obtain more accurate estimates for certain parameters, and (3) the latent volatility factor estimates should be more accurate. The estimation of stochastic volatility financial models using both returns and option prices has been approached under the frequentist paradigm by Hurn et al. (2015), among others. These approaches rely on particle filters which are computationally cumbersome for joint estimation. Jacobs & Liu (2018) offer an estimation methodology based on joint likelihoods via approximations of the option implied volatility. Their algorithm saves much of the costs associated with joint estimation when using the particle filter and makes joint Bayesian estimation feasible.

In this chapter, we demonstrate that the DNF can be used to efficiently include options into the Bayesian inference of jump-diffusion models without introducing additional sources of error. The new method allows for flexibility when specifying the error distribution associated with the option prices.

5.1 Option Pricing

Duffie et al. (2000) provide a closed-form option pricing formula for the risk-neutralized SVCJ model of Equations (2.3) and (2.4). Particularly, they present the moment generating function (mgf) for $\log(Y_{T_{mat}})$ conditional on the latent variables and the model parameters, where T_{mat} is the time at which the option expires in days. Letting $\tau = (T_{mat} - j)h$ be the time to maturity in years and $\Theta^{\mathbb{Q}}$ be the set of model parameters under the risk neural

measure, we have the following mgf for $\log(Y_{T_{mat}})$:

$$\phi(u,\, Y_j,\, x_j,\, r_j,\, \tau,\, \Theta^{\mathbb{Q}}) = \exp\left\{\mathcal{A}(u,\tau) + u\,\log(Y_j) + \mathcal{B}(u,\tau)\,x_j\right\},$$

where the coefficients

$$\begin{aligned}
\mathcal{B}(u,\tau) &= -\frac{a\,(1-e^{-\gamma\,\tau})}{2\,\gamma - (\gamma+b)\,(1-e^{-\gamma\,\tau})},\\
\mathcal{A}(u,\tau) &= \alpha_0 - \omega\,\tau\,(1+u\,\bar{\alpha}) + \omega\,\mathcal{C}(u,\mathcal{B}(u,\tau)) \quad \text{with}\\
\alpha_0 &= (r_j-\delta_j)\,u\,\tau - \tilde{\kappa}\,\tilde{\theta}\,\left(\tau\,\frac{\gamma+b}{\sigma^2} + \frac{2}{\sigma^2}\,\log\left(1-\frac{(\gamma+b)}{2\,\gamma}\,(1-e^{-\gamma\,\tau})\right)\right), \quad \text{and}\\
\mathcal{C}(u,\mathcal{B}(u,\tau)) &= e^{\tilde{\alpha}\,u+\delta^2\,\frac{u^2}{2}}\,d
\end{aligned}$$

rely on the following constants: $a = u\,(1-u)$, $b = \sigma\,\rho\,u - \tilde{\kappa}$, $c = 1-\rho_z\,\tilde{\nu}\,u$, $\gamma = \sqrt{b^2+a\,\sigma^2}$, and

$$d = \frac{\tau\,(\gamma-b)}{(\gamma-b)\,c+\tilde{\nu}\,a} - \frac{2\;\tilde{\nu}\,a}{(\gamma\,c)^2-(b\,c-\tilde{\nu}\,a)^2}\,\log\left(1-\frac{(\gamma+b)\,c-\tilde{\nu}\,a}{(2\,\gamma\,c)}(1-e^{-\gamma\,\tau})\right).$$

The moneyness of an option with strike price K, which is defined as $\mathcal{M}_j \equiv \frac{K}{Y_j}$, is an important quantity for option pricing. It is possible to write the pricing formula as a product of a function of moneyness and the spot price, Y_j. Indeed, the mgf of the log terminal price can be written as

$$\phi(u,\, Y_j,\, x_j,\, \tau,\, \Theta^{\mathbb{Q}}) = e^{K}\,\phi(u,\, \mathcal{M}_j^{-1},\, x_j,\, \tau,\, \Theta^{\mathbb{Q}}). \tag{5.1}$$

Then, we can rewrite the formula to price a European call option from the Appendix of Bégin & Gauthier (2020) as

$$\begin{aligned}
C_j(Y_j,\, x_j,\, K,\, \tau,\, \Theta^{\mathbb{Q}}) &= e^{-r_j\tau}\mathbb{E}^{\mathbb{Q}}\left[\max\left(Y_j-K\right),0\right]\\
&= Y_j\left(e^{-\delta_j\,\tau}P_1 - \mathcal{M}_j e^{-r_j\,\tau}P_2\right),
\end{aligned}$$

where

$$\begin{aligned}
P_1 &= \frac{1}{2} + \frac{\mathcal{M}_j}{\pi}\int_0^\infty e^{-r_j\,\tau}\,\mathrm{Re}\left(\frac{\phi(ui+1,\,\mathcal{M}_j^{-1},\,x_j,\,r_j,\,\tau,\,\Theta^{\mathbb{Q}})}{ui}\right)du, \quad \text{and}\\
P_2 &= \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \mathrm{Re}\left(\frac{\phi(ui,\,\mathcal{M}_j^{-1},\,x_j,\,r_j,\,\tau,\,\Theta^{\mathbb{Q}})}{ui}\right)du.
\end{aligned}$$

This formulation is practical as it can greatly reduce the computational burden of the joint estimation. If we select options that have the same maturity and moneyness for all

$j = 1, \ldots, N$, we need only compute the quantity $e^{-r_j \tau} P_1 - \mathcal{M}_j e^{-\tau} P_2$ for the different instantaneous variance values used to obtain likelihood evaluations.

With the DNF, the M instantaneous variances nodes are also fixed across time. Then, we need to evaluate the normalized option price only MN_o times, where N_o is the number of options we include at each time step. Finally, multiplying this normalized option price by the appropriate price Y_j gives the theoretical option prices $C_j(Y_j, x_j, K, \tau, \Theta^{\mathbb{Q}})$ necessary for joint likelihood evaluations.

The same approach with the particle filter would require computing $C_j(Y_j, x_j, K, \tau, \Theta^{\mathbb{Q}})$ for each of the m particles at every time j as the instantaneous variance used to evaluate the option price is almost surely distinct from that of other particles (past and present). This requires the pricing of mTN_o options where both T and m are usually much larger than M.

Although computing the price of a single option does not take much computing time, the difference between the approaches can be significant. For example, to incorporate a single option in an iteration of the pMCMC simulation study performed in Section 4.4 with 10,000 particles and ten-years series requires 25,200,000 option price evaluations while the dMCMC method requires only 50.

5.2 Joint Likelihood Contributions

For the joint estimation of jump-diffusion models, it is necessary to specify a likelihood function that includes the returns y_j and the options $c_{k,j}$, where $j = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N_o$. We use a weighted likelihood approach. The weights given to the observations can have a large influence on the likelihood as the option prices will typically outnumber the single return that we observe each day (Jacobs & Liu, 2018). Our time-j, weighted likelihood contribution therefore takes the following form:

$$p(y_j, c_{1:N_o,j} \mid y_{1:j-1}) = p(y_j \mid y_{1:j-1}) \left(\prod_{k=1}^{N_o} p(c_{k,j} \mid \Theta^{\mathbb{Q}}) \right)^{\frac{1}{N_o}}.$$

In the latter, the option prices and the returns are independent and given equal weight in the likelihood. It is similar to the likelihood used in Jacobs & Liu (2018), who warn that it "is not entirely guided by theory and therefore, to some extent, ad-hoc."

There are different potential distributions $p(c \mid \Theta^{\mathbb{Q}})$ that can be used for the option pricing errors. Hurn et al. (2015) investigate four different error specifications (ES). Their second specification (ES2) is used in Jacobs & Liu (2018). Under this specification, the observed option prices are related to the theoretical ones using an additive Gaussian noise with constant variance σ_c^2:

$$c \sim \mathcal{N}(C, \sigma_c^2).$$

ES2 has some drawbacks: it gives positive probability to negative options prices, and the size of the error is independent of the theoretical price. Both drawbacks make the specification less realistic when compared to other error distributions. Hurn et al.'s third specification (ES3), on the other hand, avoids these problems by assuming that

$$\log\left(\frac{c}{C}\right) \sim \mathcal{N}(0, \sigma_c^2).$$

ES3 suffers from the fact that the mean of the observed option distribution is not the theoretical option price. Hurn et al. (2015) remedy this in their fourth specification (ES4), but note that this last specification did not significantly affect parameter estimates and comes at an increased computational cost. For these reasons, we use ES3 in our simulation study and empirical applications.

5.3 Simulation Study

We now perform a simulation study similar to that of Section 4.4 where we estimate SVCJ model parameters from option prices as well as returns. The observations for our 100 paths are drawn from the SVCJ model with parameters given in the first column of Table 5.1. We generate three European call options: one with a moneynesses of 0.9 and a maturity of 30 days, one with a moneyness of 1 and a maturity of 90 days, and one with a moneyness of 1.1 and a maturity of 120 days. We assume that on the option price errors are distributed according to ES3.

In addition to estimating the risk-neutral parameters, ρ_z is assumed to be unknown. We apply the dMCMC algorithm to estimate the parameters of the model with both sources of information and use the correct error specification (i.e., ES3) to compute the joint likelihood contributions.

We keep the same prior distributions as those used in the simulation study of Section 4.4 for the common parameters and assume that $\rho_z \sim \mathcal{N}(0, 4)$, $\eta_{J^Y} \sim \mathcal{N}(0, 4)$, $\tilde{\kappa} = \kappa - \eta_x > 0$, $\sigma_c > 0$, and that $\tilde{\nu} = \nu - \eta_{JX} > 0$. With $p = 100$, $G_0 = 500$, $M = 50$, $S = 20$, $R = 1$, and $G = 10,000$ iterations, we obtain the results shown in Table 5.1. Table A.3 in Appendix A gives the results for a similar study with nine options.

With three options and this choice of grid node density, the dMCMC speed is similar to that of the pMCMC algorithm using 10,000 particles without options. With the options, the dMCMC obtains lower RMSEs than the pMCMC across the jump parameters despite having to estimate ρ_z. The dMCMC gets lower RMSEs for κ and θ, but has higher errors for η_y, ρ, and σ. We also get estimates for the risk premiums and the option price errors. The risk premiums associated with the risk-neutral measure are η_x, η_{J^Y}, and η_{JX} . These are typically difficult to estimate. Indeed, we obtain a large RMSE on η_x. On the other

Table 5.1: Parameter estimates with returns and options.

	True Value	**dMCMC**
ρ	-0.745	-0.819
		(0.077)
$100\,\theta$	3.200	3.209
		(0.390)
σ	0.446	0.384
		(0.064)
κ	3.689	3.595
		(0.218)
ω	5.125	4.033
		(1.265)
$1000\,\nu$	4.000	4.617
		(1.369)
$1000\,\delta$	3.000	3.756
		(1.808)
$1000\,\alpha$	-7.000	-9.109
		(3.803)
ρ_z	-1.809	-1.642
		(0.523)
η_y	3.000	3.187
		(0.906)
$100\,\eta_x$	5.000	2.788
		(8.374)
$100\,\eta_{JY}$	3.610	4.035
		(0.842)
$1000\,\eta_{JX}$	2.000	1.651
		(1.296)
$100\,\sigma_c$	5.000	5.534
		(0.539)

hand, η_{JY}, and η_{JX} have relatively smaller RMSEs. The dMCMC tends to overestimate the option price errors.

Figures C.3 and C.4 in Appendix C compare the posterior means with 95% confidence intervals for 10,000 iterations of dMCMC when we include three options. We can see that the dMCMC has some difficulty and that the posterior volatility estimates do not yet capture the behaviour of the true volatility. When we add options, however, the posterior mean is close to the true volatility and the confidence intervals are tighter.

5.4 Empirical Application

We apply the joint estimation techniques described in Section 5.1 to the S&P 500 index returns series (excluding dividends) from 2000 to 2019 (this series is also used in Section 4.5). We include nine European call options per day. These nine options have all combinations with a moneyness of 0.95, 1, or 1.05, and maturities of 30 days, 90 days, or 150 days. Using the same priors, error specification, and adaptive Metropolis-Hastings parameter values as in Section 5.3, we obtain the SVCJ model parameter estimates given in the third column of Table 5.2 with $G = 50{,}000$ dMCMC iterations. After the 10,000 burn-in period, we have an acceptance probability of about 7%. Three parameters are rejected at the 5% level by the Geweke convergence test: θ, η_y, and η_{JX}. Trace plots for these parameters are given in Appendix E.

In the second column of Table 5.2, we give the parameter estimates from Jacobs & Liu (2018). with the other parameters. Although they also performed joint estimation of the S&P 500 index with returns and options data, there are a number of differences in the methodology and data used. They use 30 options per day and data from 1996 to 2015. They run a pMCMC algorithm for 2,500 iterations and use 10,000 particles for the SMC. We use the ES3, while they use the ES2 for the option price distribution. Finally, we estimate σ_c along with the other parameters.

Parameters η_y, ρ, ω, α, and ρ_z remain similar to our returns-only estimates in the second column of Table 4.3. When including options, θ σ, κ, and δ decrease. Jacobs & Liu (2018) get similar decreases for κ, θ, and σ, but not for δ. Interestingly, we get a very similar κ to the option-only estimates from their work (i.e., they obtained 0.6183). Like them, we find positive and statistically significant return jump risk premium η_{JY}. We also both find significant negative ρ_z parameters, although our value is closer to the one in Bégin & Boudreault (2020). Moreover, we find a positive, but not statistically significant, diffusive variance risk premium parameter η_x. Jacobs & Liu (2018) obtain statistically significant results using models without variance jumps, but mention that, when they include variance jumps, this risk premium is harder to identify and is sometimes negative. Our estimate of σ_c resembles those found by Hurn et al. (2015) when performing joint estimation using S&P 500 index returns and options prices from 1990 to 2011.

Table 5.2: Parameter estimates with returns and options.

Parameter	**JL2018** (Standard error)	**dMCMC Mean** (Standard error)
ρ	-0.924	-0.845
	(0.017)	(0.012)
$100\,\theta$	2.410	3.073
	(0.080)	(0.304)
σ	0.345	0.217
	(0.007)	(0.012)
κ	1.125	0.787
	(0.049)	(0.107)
ω	0.601	2.246
	(0.036)	(0.521)
$1000\,\nu$	60.800	12.285
	(1.300)	(3.265)
$1000\,\delta$	42.600	13.602
	(0.600)	(4.669)
$1000\,\alpha$	-10.400	-0.198
	(0.500)	(4.197)
ρ_z	-0.503	-2.265
	(0.008)	(0.486)
η_y	3.040	0.784
	(0.349)	(0.413)
$100\,\eta_x$	4.980	0.531
	(3.550)	(10.207)
$100\,\eta_{JY}$	3.610	-2.345
	(0.030)	(0.497)
$1000\,\eta_{JX}$	1.800	-3.548
	(1.500)	(2.453)
$100\,\sigma_c$		10.784
		(0.487)

Chapter 6

Conclusion and Future Work

The jump-diffusion framework captures key features of market dynamics through its stochastic volatility, instantaneous variance jumps, and return jumps. Although essential for generating certain stylized facts about the market, these components complicate parameter estimation. Specifically, Bayesian estimation which relies on Monte Carlo methods to obtain posterior parameter distributions becomes a difficult task in the presence of numerous latent factors.

One approach to Bayesian estimation of general state-space models presented in Andrieu et al. (2010) is to draw latent variable proposals from a particle filter within an MCMC method. In this project, we show that the discrete nonlinear filter of Kitagawa (1987) can be combined in a similar fashion to create a dMCMC algorithm that can estimate posterior parameter distributions for complex models. For this, we need two things from the discrete filter: likelihood evaluations and samples from the smoothed volatility distribution. For the former, Bégin & Boudreault (2020) already applied the DNF to jump-diffusion models and found that it was fast and accurate compared to the particle filter. For the latter, we apply a filter-forwarding-backward-sampling algorithm to the DNF as done by Frühwirth-Schnatter (1994) and Carter & Kohn (1994) for the Kalman fitler. As this sampling is fast, the benefits of using the DNF over the particle filter for likelihood evaluations carry over to Bayesian estimation with the dMCMC algorithm.

Moreover, we show that the DNF can efficiently evaluate the joint likelihood of a series using returns and options. It has been argued that there is information about market dynamics contained in returns and in options, thus including both is advantageous in the context of parameter estimation (Renault, 1997; Hurn et al., 2015; Jacobs & Liu, 2018). And, this allows for the simultaneous estimation of the physical and risk-neutral model dynamics. We show that the dMCMC can quickly perform joint estimation and allows for any option price error specification.

For future work, we believe that the dMCMC algorithm can aid in performing joint estimation with alternative volatility dynamics that do not necessarily yield closed-form option pricing formulas (for example, see Christoffersen et al., 2010). Finally, the dMCMC

and pMCMC algorithms could be extended to account for model uncertainty via Bayesian model averaging by including proposals along the model space with the reversible jump approach of Green (1995).

References

An, S., & Schorfheide, F. (2007). Bayesian analysis of DSGE models. *Econometric Reviews*, *26*(2-4), 113–172.

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.

Bakshi, G., Cao, C., & Chen, Z. (1997). Empirical performance of alternative option pricing models. *Journal of Finance*, *52*(5), 2003–2049.

Bartolucci, F., & De Luca, G. (2001). Maximum likelihood estimation of a latent variable time-series model. *Applied Stochastic Models in Business and Industry*, *17*(1), 5–17.

Bartolucci, F., & De Luca, G. (2003). Likelihood-based inference for asymmetric stochastic volatility models. *Computational Statistics & Data Analysis*, *42*(3), 445–449.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options. *Review of Financial Studies*, *9*(1), 69–107.

Bates, D. S. (2000). Post-'87 crash fears in the s&p 500 futures option market. *Journal of Econometrics*, *94*(1-2), 181–238.

Bégin, J.-F., & Boudreault, M. (2020). Likelihood evaluation of jump-diffusion models using deterministic nonlinear filters. *Journal of Computational and Graphical Statistics*, 1–15.

Bégin, J.-F., & Gauthier, G. (2020). Price bias and common practice in option pricing. *Canadian Journal of Statistics*, *48*(1), 8–35.

Black, F., & Scholes, M. (1973). The valuation of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Brine, K. R., & Poovey, M. (2017). *Finance in America: An unfinished story.* University of Chicago Press.

Carter, C. K., & Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika*, *81*(3), 541–553.

Chib, S., & Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *American Statistician*, *49*(4), 327–335.

Christoffersen, P., Jacobs, K., & Mimouni, K. (2010). Volatility dynamics for the S&P 500: Evidence from realized volatility, daily returns, and option prices. *Review of Financial Studies*, *23*(8), 3141–3189.

Cowles, M. K., & Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, *91*(434), 883–904.

Cowles, M. K., Roberts, G. O., & Rosenthal, J. S. (1999). Possible biases induced by MCMC convergence diagnostics. *Journal of Statistical Computation and Simulation*, *64*(1), 87–104.

Cox, J. C., Ingersoll Jr, J. E., & Ross, S. A. (1985). A theory of the term structure of interest rates. *Econometrica*, *53*(2), 385–408.

Creal, D. (2012). A survey of sequential Monte Carlo methods for economics and finance. *Econometric Reviews*, *31*(3), 245–296.

Danielsson, J. (1994). Stochastic volatility in asset prices estimation with simulated maximum likelihood. *Journal of Econometrics*, *64*(1-2), 375–400.

Danielsson, J., & Richard, J.-F. (1993). Accelerated Gaussian importance sampler with application to dynamic latent variable models. *Journal of Applied Econometrics*, *8*(S1), 153–173.

Dobrow, R. P. (2016). *Introduction to stochastic processes with R*. John Wiley & Sons.

Doucet, A., Briers, M., & Sénécal, S. (2006). Efficient block sampling strategies for sequential Monte Carlo methods. *Journal of Computational and Graphical Statistics*, *15*(3), 693–711.

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

Eraker, B. (2001). MCMC analysis of diffusion models with application to finance. *Journal of Business & Economic Statistics*, *19*(2), 177–191.

Eraker, B. (2004). Do stock prices and volatility jump? Reconciling evidence from spot and option prices. *Journal of Finance*, *59*(3), 1367–1403.

Eraker, B., Johannes, M., & Polson, N. (2003). The impact of jumps in volatility and returns. *Journal of Finance*, *58*(3), 1269–1300.

Fernández-Villaverde, J., & Rubio-Ramírez, J. F. (2007). Estimating macroeconomic models: A likelihood approach. *Review of Economic Studies*, *74*(4), 1059–1087.

Fridman, M., & Harris, L. (1998). A maximum likelihood approach for non-Gaussian stochastic volatility models. *Journal of Business & Economic Statistics*, *16*(3), 284–291.

Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, *15*(2), 183–202.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, *7*(4), 457–472.

Geweke, J. (1992). Evaluating the accurating of sampling-based approaches to the calculation of posterior moments. *Bayesian Statistics*, *4*, 169–193.

Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEEE proceedings F: Radar and Signal Processing)* (Vol. 140, pp. 107–113).

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, *82*(4), 711–732.

Harvey, A., Ruiz, E., & Shephard, N. (1994). Multivariate stochastic variance models. *Review of Economic Studies*, *61*(2), 247–264.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *21*, 97–109.

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.

Hurn, A. S., Lindsay, K. A., & McClelland, A. J. (2015). Estimating the parameters of stochastic volatility models using option price data. *Journal of Business & Economic Statistics*, *33*(4), 579–594.

Jacobs, K., & Liu, Y. (2018). Estimation and filtering with big option data. *Working Paper*.

Jacquier, E., Polson, N. G., & Rossi, P. E. (1994). Bayesian analysis of stochastic volatility models. *Journal of Business & Economic Statistics*, *12*(4).

Johannes, M., & Polson, N. (2010). MCMC methods for continuous-time financial econometrics. In *Handbook of financial econometrics: Applications* (pp. 1–72). Elsevier.

Johannes, M., Polson, N., & Stroud, J. (2009). Optimal filtering of jump diffusions: Extracting latent states from asset prices. *Review of Financial Studies*, *22*(7), 2759–2799.

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In I. Kadar (Ed.), *Signal Processing, Sensor Fusion, and Target Recognition VI* (Vol. 3068, pp. 182 – 193). SPIE.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45.

Kim, S., Shephard, N., & Chib, S. (1998). Stochastic volatility: Likelihood inference and comparison with arch models. *Review of Economic Studies*, *65*(3), 361–393.

Kitagawa, G. (1987). Non-Gaussian state-space modeling of nonstationary time series. *Journal of the American Statistical Association*, *82*(400), 1032–1041.

Langrock, R., MacDonald, I. L., & Zucchini, W. (2012). Some nonstandard stochastic volatility models and their estimation using structured hidden markov models. *Journal of Empirical Finance*, *19*(1), 147–161.

Lord, R., Koekkoek, R., & Dijk, D. V. (2010). A comparison of biased simulation schemes for stochastic volatility models. *Quantitative Finance*, *10*(2), 177–194.

Malik, S., & Pitt, M. K. (2011). Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, *165*(2), 190–209.

Melino, A., & Turnbull, S. M. (1990). Pricing foreign currency options with stochastic volatility. *Journal of Econometrics*, *45*(1-2), 239–265.

Mengersen, K. L., Tweedie, R. L., et al. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *Annals of Statistics*, *24*(1), 101–121.

Merton, R. C. (1973). Theory of rational option pricing. *Bell Journal of Economics and Management Science*, 141–183.

Merton, R. C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, *3*(1-2), 125–144.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*(6), 1087–1092.

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.

Pitt, M. K., Malik, S., & Doucet, A. (2014). Simulated likelihood inference for stochastic volatility models using continuous particle filtering. *Annals of the Institute of Statistical Mathematics*, *66*(3), 527–552.

Pitt, M. K., & Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, *94*(446), 590–599.

Pole, A., & West, M. (1990). Efficient Bayesian learning in non-linear dynamic models. *Journal of Forecasting*, *9*(2), 119–136.

Rémillard, B. (2013). *Statistical methods for financial engineering*. CRC Press.

Renault, E. (1997). Econometric models of option pricing errors. *Econometric Society Monographs*, *28*, 223–278.

Robert, C., & Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Roberts, G. O., & Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, *18*(2), 349–367.

Ruiz, E. (1994). Quasi-maximum likelihood estimation of stochastic volatility models. *Journal of Econometrics*, *63*(1), 289–306.

Scott, L. O. (1987). Option pricing when the variance changes randomly: Theory, estimation, and an application. *Journal of Financial and Quantitative analysis*, 419–438.

Shephard, N. (1993). Fitting nonlinear time-series models with applications to stochastic variance models. *Journal of Applied Econometrics*, *8*(S1), S135–S152.

Smith, A. F., & Roberts, G. O. (1993). Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *55*(1), 3–23.

Tanizaki, H., & Mariano, R. S. (1994). Prediction, filtering and smoothing in non-linear and non-normal cases using Monte Carlo integration. *Journal of Applied Econometrics*, *9*(2), 163–179.

Taylor, S. J. (1986). *Modelling financial time series*. World Scientific.

Watanabe, T. (1999). A non-linear filtering approach to stochastic volatility models with an application to daily stock returns. *Journal of Applied Econometrics*, *14*(2), 101–121.

Appendix A

Simulation Study Posterior Parameter Statistics

In this section, we give additional statistics from the simulation performed in Sections 4.4 and 5.3.

Table A.1: Parameter median comparison.

Parameter	**True Value**	**pMCMC Median**	**dMCMC Median**
η_y	3.000	2.588	2.503
ρ	-0.745	-0.747	-0.743
$100\,\theta$	3.200	2.778	2.951
σ	0.446	0.401	0.397
κ	3.689	3.679	3.782
ω	5.125	3.568	3.107
$1000\,\nu$	4.000	4.920	4.835
$1000\,\delta$	3.000	5.600	5.978
$1000\,\alpha$	-7.000	-4.761	-2.775

Table A.2: Parameter modes comparison.

Parameter	**True Value**	**pMCMC Mode**	**dMCMC Mode**
η_y	3.000	3.074	2.790
ρ	-0.745	-0.761	-0.753
$100\,\theta$	3.200	2.754	3.075
σ	0.446	0.396	0.400
κ	3.689	3.769	3.641
ω	5.125	3.874	2.837
$1000\,\nu$	4.000	5.880	4.760
$1000\,\delta$	3.000	5.007	5.253
$1000\,\alpha$	-7.000	-5.198	-4.387

Table A.3 gives the results for a simulation study similar to the one in Section 5.3, but with nine options. The nine options have all combinations between a moneyness of 0.95, 1 and 1.05, with maturities of 30 days, 90 days, and 150 days. With these nine options, we still

Table A.3: Parameter estimates with returns and nine options.

Parameter	**True Value**	**dMCMC Median** (RMSE)
ρ	-0.745	-0.809
		(0.069)
$100\,\theta$	3.200	2.915
		(0.403)
σ	0.446	0.382
		(0.068)
κ	3.689	3.427
		(0.399)
ω	5.125	5.364
		(0.693)
$1000\,\nu$	4.000	4.096
		(1.363)
$1000\,\delta$	3.000	3.992
		(2.233)
$1000\,\alpha$	-7.000	-6.936
		(4.338)
ρ_z	-1.809	-1.931
		(0.631)
η_y	3.000	2.835
		(0.934)
$100\,\eta_x$	5.000	-4.570
		(21.944)
$100\,\eta_{J^Y}$	3.610	3.558
		(0.466)
$1000\,\eta_{J^X}$	2.000	0.645
		(1.921)
$100\,\sigma_c$	5.000	7.886
		(2.879)

find better RMSEs for the jump parameters than with returns only and similar results for the other parameters. With more options, we obtain better estimates of ω, α, ρ_z, and η_{J^Y} than with the three options of Section 5.3. However, the estimates of η_x and σ_c have higher RMSEs. There is a large standard error of 0.169 associated with η_x.

Appendix B

Empirical Study Data

This section displays plots of the TSX Composite returns time series used in Section 4.5 and the S&P 500 returns series used in Sections 4.5 and 5.4.

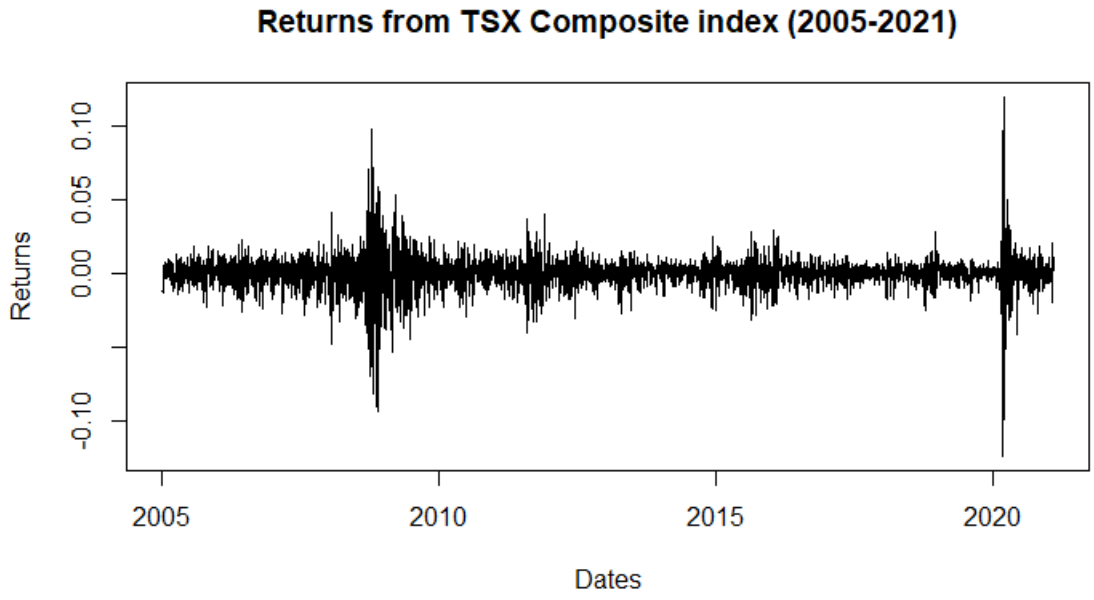

Figure B.1: Daily returns (excluding dividends) from the TSX Composite index from January 2005 to February 10th 2021.

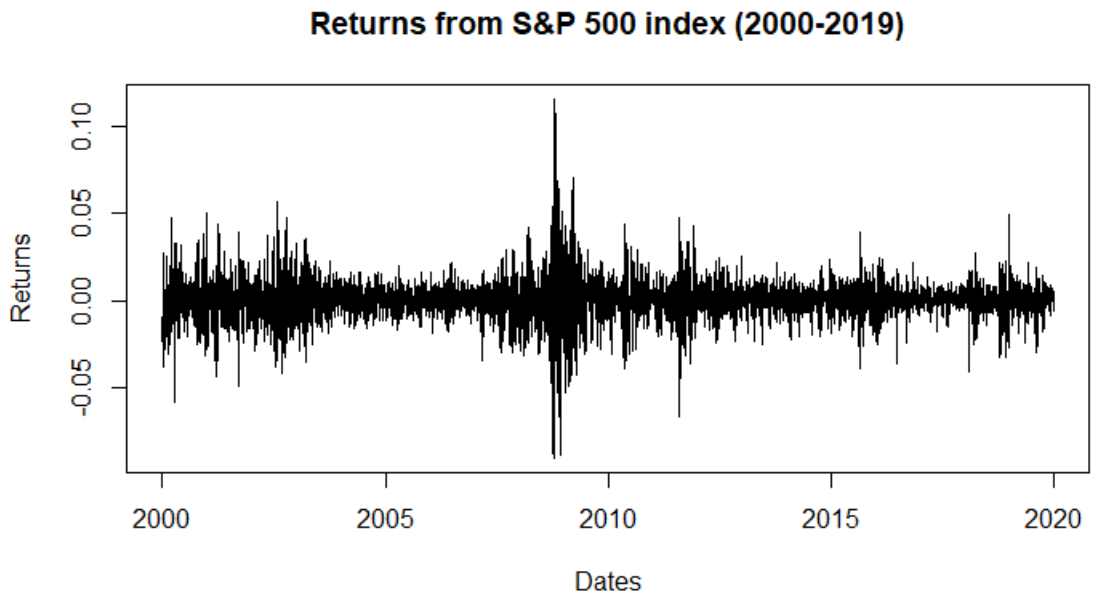

Figure B.2: Daily returns (excluding dividends) from the S&P 500 index from January 2000 to December 2019.

Appendix C

Posterior Volatility Comparisons

This section provides plots of the posterior mean instantaneous variance with 95% confidence intervals using different algorithms (pMCMC and dMCMC) and different simulated observables (returns and options).

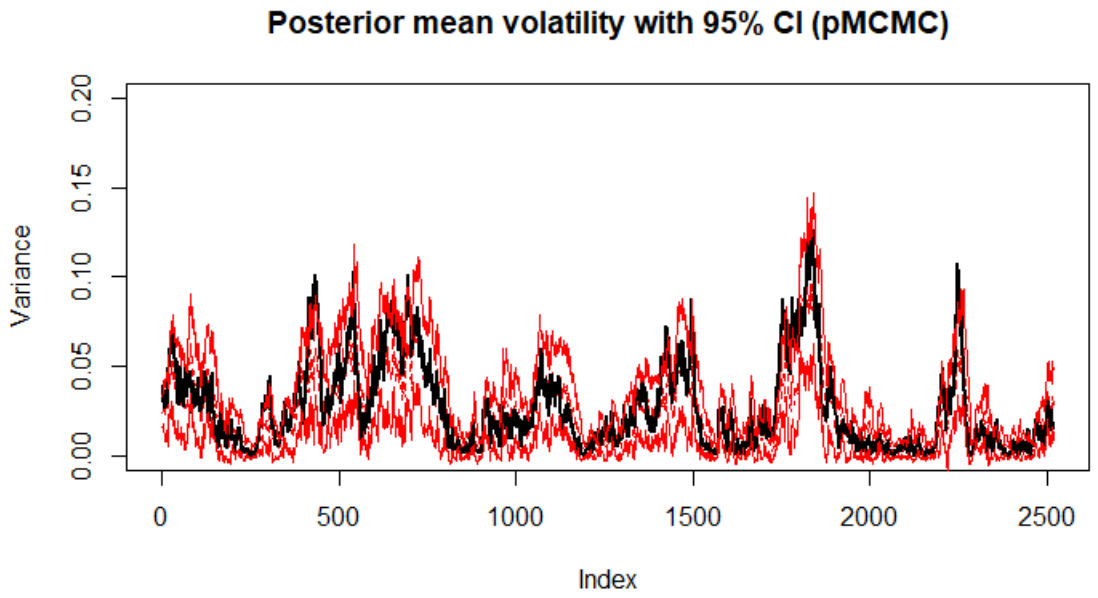

Figure C.1: Posterior mean volatility (dotted red) with 95% confidence interval (red) and true variance (black) for 10,000 pMCMC iterations.

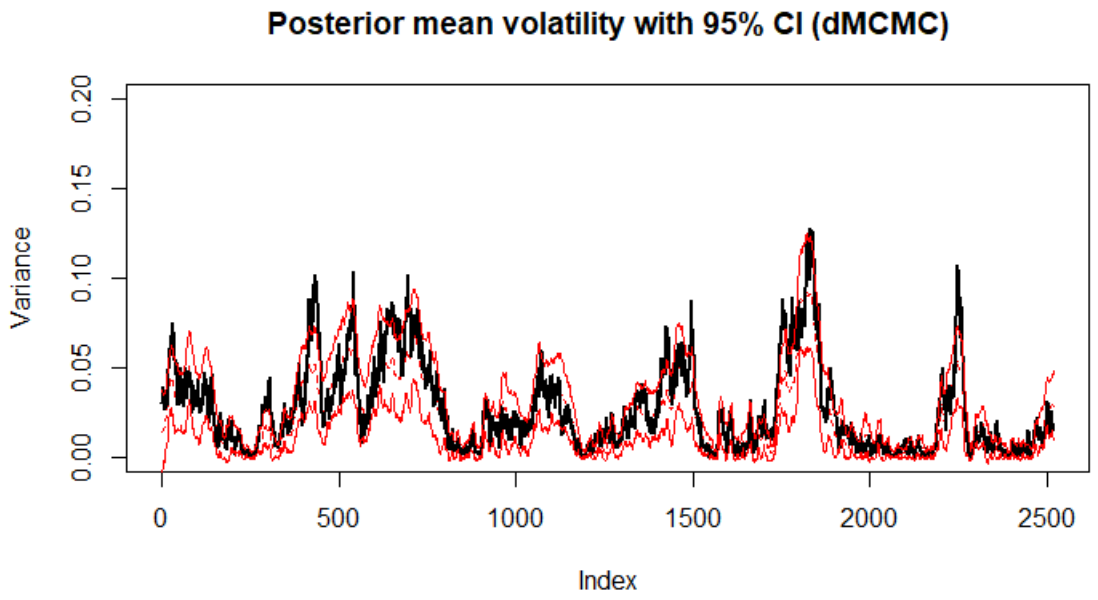

Figure C.2: Posterior mean volatility (dotted red) with 95% confidence interval (red) and true variance (black) for 10,000 dMCMC iterations.

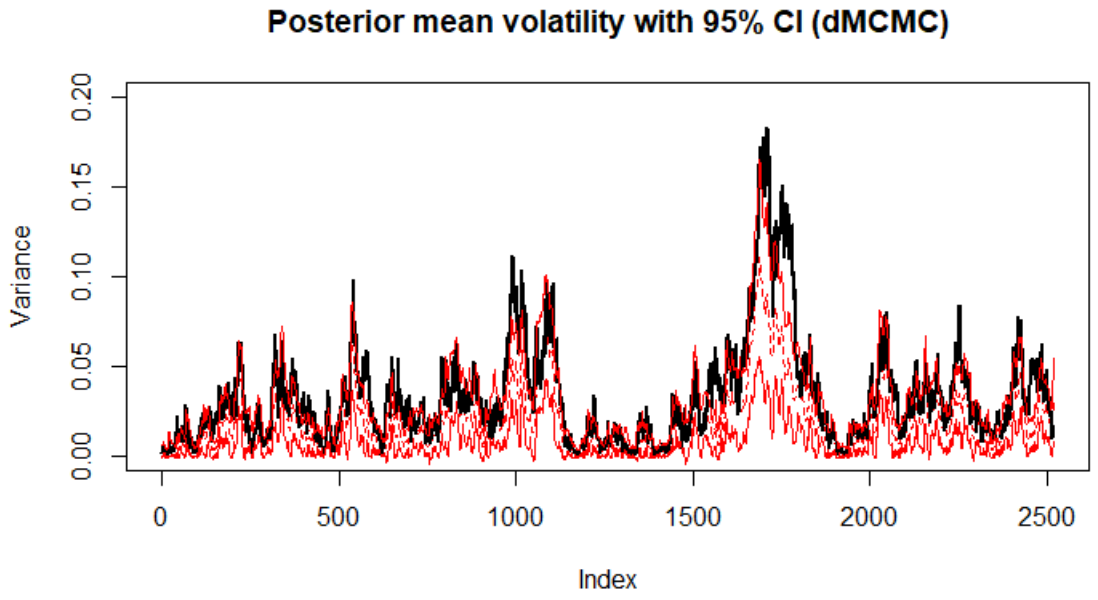

Figure C.3: Posterior mean volatility (dotted red) with 95% confidence interval (red) and true variance (black) for 10,000 dMCMC iterations with returns.

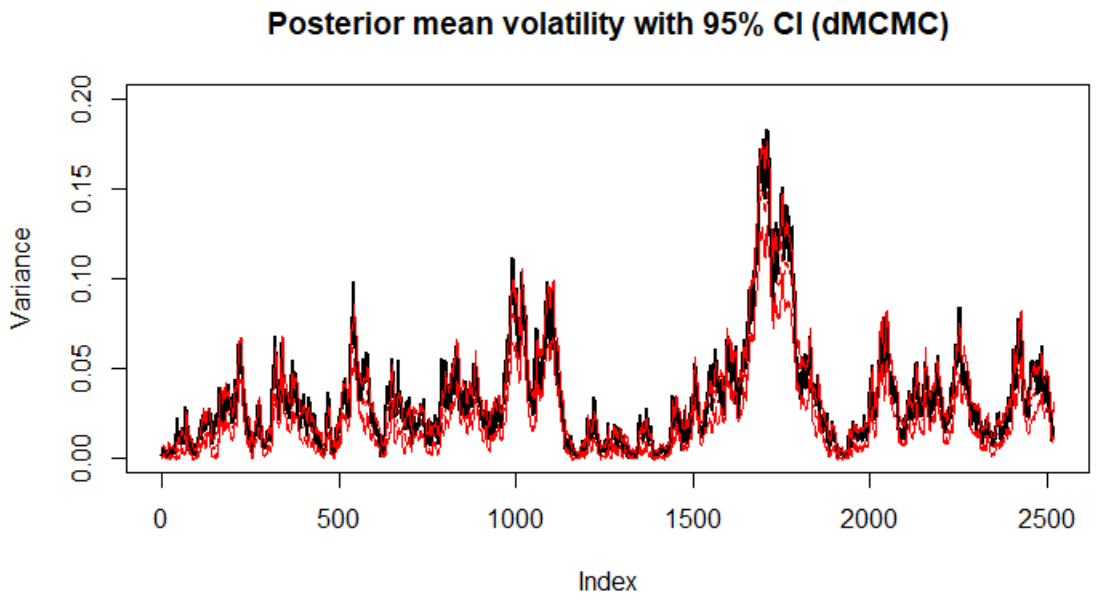

Figure C.4: Posterior mean volatility (dotted red) with 95% confidence interval (red) and true variance (black) for 10,000 dMCMC iterations with returns and three options.

Appendix D

Returns-Only Estimation Trace Plots

In this appendix, we provide the trace plots for parameters from the empirical application of the dMCMC in Section 4.5. Particularly, these are the parameter for which the Geweke test rejects convergence at the 5% level for either the TSX Composite or the S&P 500 series.

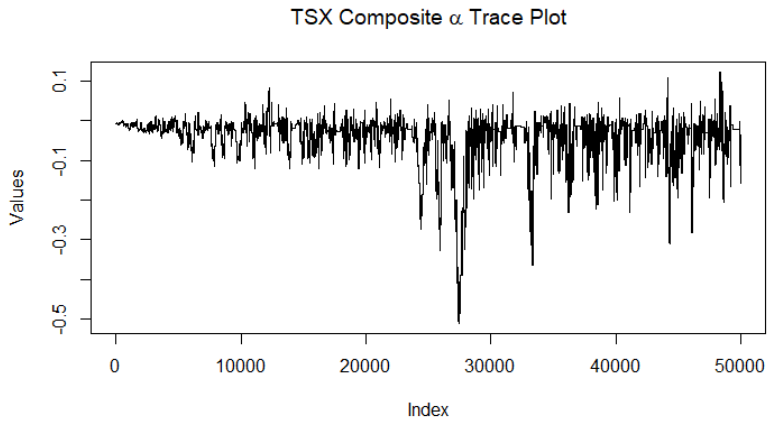

Figure D.1: Trace plot for α with 50,000 dMCMC iterations for the TSX Composite return series.

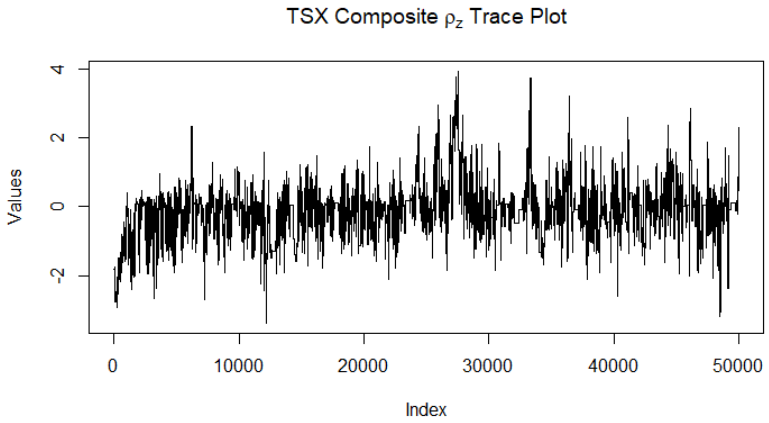

Figure D.2: Trace plot for ρ_z with 50,000 dMCMC iterations for the TSX Composite return series.

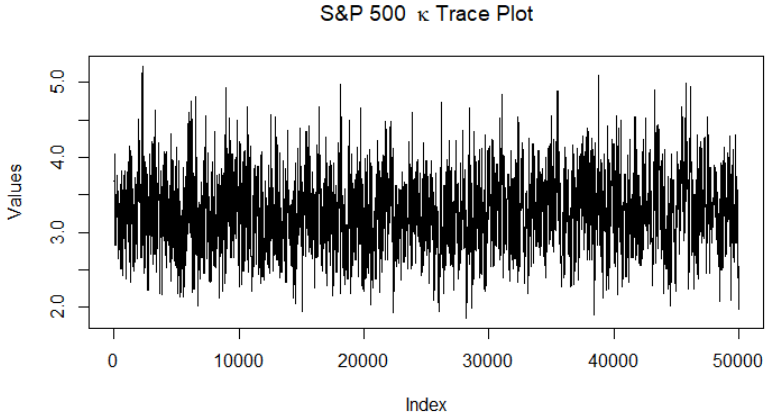

Figure D.3: Trace plot for κ with 50,000 dMCMC iterations for the S&P 500 return series.

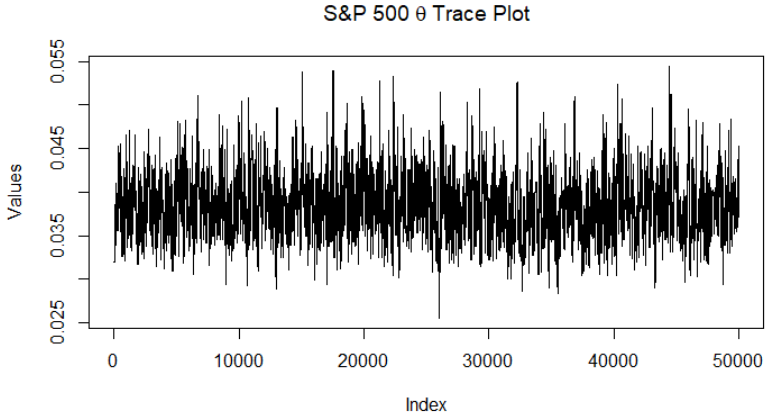

Figure D.4: Trace plot for θ with 50,000 dMCMC iterations for the S&P 500 return series.

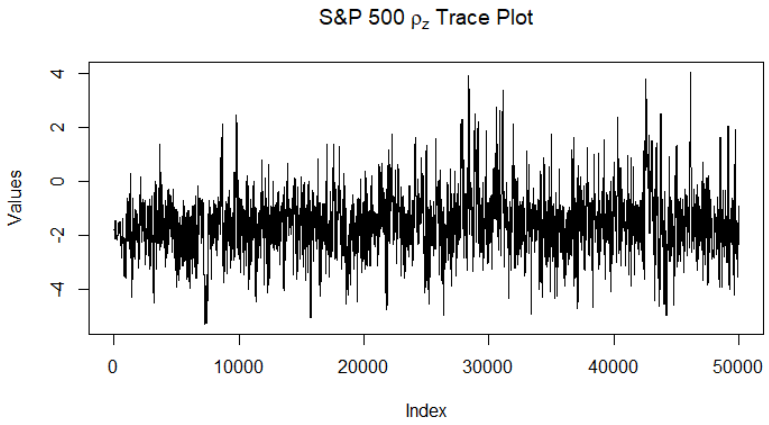

Figure D.5: Trace plot for ρ_z with 50,000 dMCMC iterations for the S&P 500 return series.

Appendix E

Joint Estimation Trace plots

In this appendix, we provide the trace plots for parameters from the empirical application of the dMCMC for joint estimation in Section 5.4. Particularly, these are the parameter for which the Geweke test rejects convergence at the 5% level.

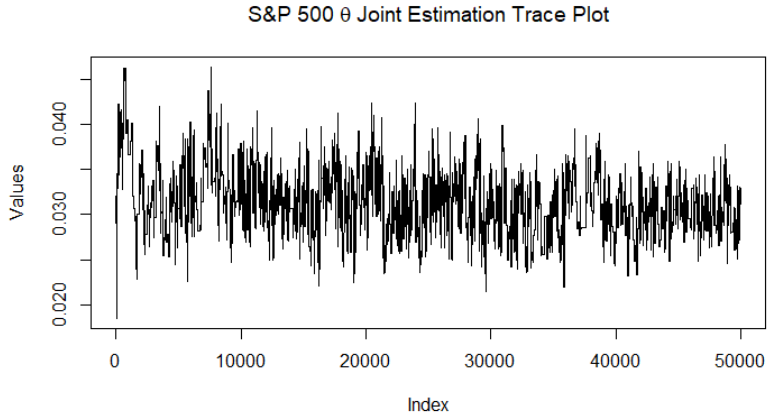

Figure E.1: Trace plot for θ with 50,000 dMCMC iterations for the S&P 500 return and options series.

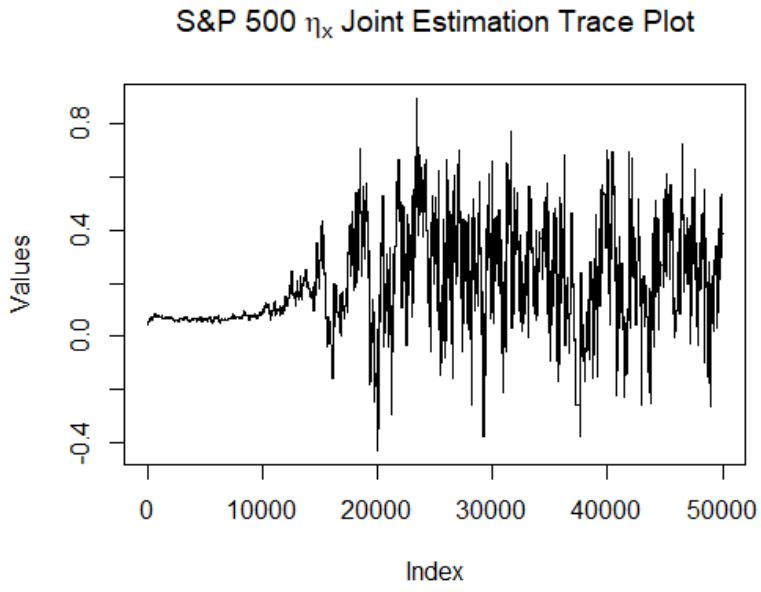

Figure E.2: Trace plot for η_x with 50,000 dMCMC iterations for the S&P 500 return and options series.

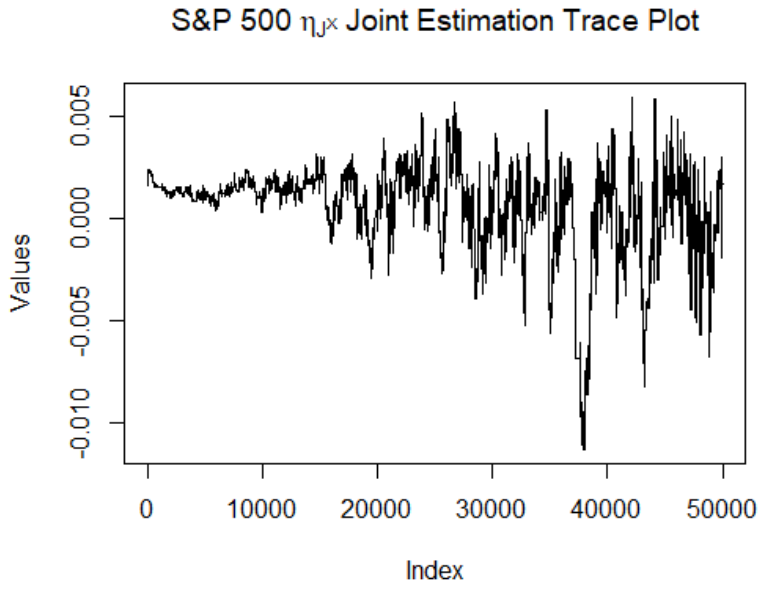

Figure E.3: Trace plot for η_{JX} with 50,000 dMCMC iterations for the S&P 500 return and options series.