Efficient learning via simulation: A marginalized resample-move approach

Andras Fulop\textsuperscript{a}, Junye Li\textsuperscript{b,∗}

\textsuperscript{a} Finance Department at ESSEC Business School, Paris–Singapore, 95021 Cergy-Pointoise, France
\textsuperscript{b} Finance department at ESSEC Business School, Paris–Singapore, 188064, Singapore

\textbf{A R T I C L E  I N F O}

Article history:
Received 7 September 2011
Received in revised form 2 January 2013
Accepted 10 May 2013
Available online 20 May 2013

Keywords:
State–space models
Particle filters
Parameter learning
State filtering
Resample–move
Markov chain Monte Carlo
Lévy jumps
Stochastic volatility
Credit risk

\textbf{A B S T R A C T}

In state–space models, parameter learning is practically difficult and is still an open issue. This paper proposes an efficient simulation-based parameter learning method. First, the approach breaks up the interdependence of the hidden states and the static parameters by marginalizing out the states using a particle filter. Second, it applies a Bayesian resample-move approach to this marginalized system. The methodology is generic and needs little design effort. Different from batch estimation methods, it provides posterior quantities necessary for full sequential inference and recursive model monitoring. The algorithm is implemented both on simulated data in a linear Gaussian model for illustration and comparison and on real data in a Lévy jump stochastic volatility model and a structural credit risk model.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

State–space models are common in statistics, engineering, economics and finance. There are two closely related statistical inference problems in these models, i.e., state filtering and parameter estimation. In a few special cases such as linear Gaussian models and Markov-switching models, optimal solutions are available for the filtering problem, opening the way to likelihood-based inference over the parameters (Durbin and Koopman, 2001; Kim and Nelson, 1998). Unfortunately, in general nonlinear and/or non-Gaussian models, the optimal filter is not known and the analytical likelihood is not obtainable. This makes parameter estimation non-trivial. State filtering in such systems is a well-studied issue and standard solutions are now available through sequential Monte Carlo methods, also called particle filters (Doucet et al., 2001; Del Moral et al., 2006). In contrast, estimation of the fixed parameters is more involved. In a batch context, numerous Markov chain Monte Carlo (MCMC) methods have been developed for specific classes of models (Jacquier et al., 1994; Eraker et al., 2003; Li et al., 2008; Fruhwirth-Schnatter and Sognor, 2008). The recent particle MCMC (PMCMC) approach of Andrieu et al. (2010), which combines MCMC chain and sequential Monte Carlo methods, provides a generic solution to parameter estimation in batch setups. They show that MCMC samplers converge to the real posterior distribution of parameters even when the likelihood approximated by sequential Monte Carlo methods is used. Afterwards, several improvements have appeared. Pitt et al. (2012) prove that the likelihood from the fully adapted auxiliary particle filter is unbiased and provide practical guidelines on how to choose the optimal number of particles. Duan and Fulop (2012) propose a parallelized density-tempered algorithm based on the sequential Monte Carlo sampler approach of Del Moral et al. (2006). Having observed that PMCMC is sensitive to both the number of state particles and the number of observations, Lindsten and Schön (2012) propose to introduce backward simulation in PMCMC algorithms.

However, often in applications, especially in finance and economics, one is more interested in sequential estimation, where estimates of both states and parameters are updated as new
observations arrive. For example, in finance, investors need to update their belief on volatility and pricing models recursively whenever new market data become known. As another example, in economics, analysts need to revise their forecasts as news on the state of economy evolve. Different from batch estimation, a sequential approach can fully take into account impacts of parameter and model uncertainties on decision-making over time. Under these recursive set-ups, MCMC techniques are time-consuming to use as they have to repeatedly regenerate Markov chains over time. Sequential joint inference of parameters and states is practically difficult and is still an open issue. The objective of this paper is to propose a generic solution to this sequential learning problem that is efficient and needs a light design effort from users.

There has already been progress towards tackling parameter learning in general state–space models. It has been quickly recognized that the simple solution of including fixed parameters in the state–space does not work. The underlying reason is that the constancy of model parameters violates the forgetting property that is needed for the successful running of particle filtering algorithms (Doucet et al., 2004). Liu and West (2001) propose a sequential kernel approximation to the posterior distribution by introducing artificial stochastic evolution to parameters, an approach followed and refined by Flury and Shephard (2009). This approach is general and easy to implement, but introduces extra uncertainty into the posterior distribution of parameters, a feature whose implications are not always clear. Gilks and Berzuini (2001) propose to rejuvenate particles using MCMC kernels that leave the target distribution unchanged. Their approach typically requires the design of Gibbs-type MCMC moves over parameters given the hidden states and vice versa, both of which are highly model-specific and can require lots of design input from users. Further, the approach does not use the natural structural difference between the fixed parameters and the hidden states available in state–space systems. Storvik (2002) and Fearnhead (2002) assume that the posterior distribution of parameters depends on a set of sufficient statistics, based on which the co-dependence of states and parameters can be broken up. Equipped with this property, Carvalho et al. (2010) propose a particle learning algorithm. This approach does not move the whole path of the hidden states, making it fast. However, this feature can also make such an approach unreliable as the sample size increases (see discussions by Chopin et al., 2011). Most importantly, the scope of the method is limited to models with small-dimensional sufficient statistics.

In this paper, we propose a parameter learning method that is general and is essentially black-box, needing a relatively smaller user input. The theoretical underpinning of our approach is that as in PPMC algorithms, the likelihood estimate provided by particle filters is unbiased. We first break up the interdependence of the hidden states and the static parameters by marginalizing out states using a particle filter. Here, the unbiased likelihood estimate is obtained by running the particle filtering algorithm for any given model parameter set. Then, we apply a Bayesian resample–move algorithm to the marginalized system. The basic idea of this algorithm is similar to Gilks and Berzuini (2001) and in particular, the version proposed by Chopin (2002). The original algorithm of Chopin (2002) requires the knowledge of the likelihood function and MCMC kernels of invariant distribution of parameters. In our algorithm, we propose to approximate the likelihood through a particle filter and use a particle MCMC algorithm of Andrieu et al. (2010) to generate MCMC kernels. This marginalized resample–move approach (MRM) is exact in the sense that for any fixed number of M particles used to marginalize out the hidden states, it delivers sequential samples from the posterior distributions as the number of particles over the fixed parameters, N, goes to infinity. The main implementation effort from users is the design of an efficient particle filter for the model at hand. Once the particle filtering algorithm is in place, the extra steps necessary for parameter learning are almost automatic. Further, the sequential marginal likelihood necessary for recursive Bayesian model comparison is naturally obtained from the algorithm. Of course, a cost we need to pay is computational time. However, different from MCMC methods, which are difficult to be parallelized in general, our algorithm is embarrassingly parallel, so it can benefit from the massive computational power of modern graphical processor-based parallel architectures (GPU) (see Lee et al. (2010) on the computational complexity and parallel simulation-based methods).

Recently, we have been referred that independently of and concurrently to us, Chopin et al. (forthcoming) have developed a set of algorithms similar to ours. Their treatment is more general and in particular, allows for particle Gibbs-type schemes. Further, they propose various strategies to automatically calibrate the number of state particles over time, making the acceptance rates stable as the sample size increases. Last, they also provide a thorough theoretical analysis of the validity and computational complexity of the algorithm. Our work, however, puts greater emphasis on interesting financial applications and computational implementation. In particular, we apply this learning approach to two state-of-the-art financial models and show how to use modern graphical processing units to make our demanding examples workable. Another recent entry related to our work is Duan and Fulop (2012) who propose a marginalized sequential Monte Carlo routine over the fixed parameters and a tempered sequence of distributions between the prior and the posterior. The most important difference is that their approach applies in a batch setup and does not provide a solution to the sequential problems we are concerned with. Of course, our sequential method also delivers full-sample posterior estimates at the final date. Here, the main advantage of the data-expanding learning approach is its computational efficiency as in the initial phases of the algorithm when resample-move steps are frequent, it only needs to screen through a small part of the data sample. In contrast, the density-tempered batch approach needs to process the full dataset from the beginning. This latter procedure is not without its merits either. It could perform more stably and better in datasets with outliers due to its more gradual incorporation of information. For a more detailed comparison and experimental evidence on the two schemes, please see Duan and Fulop (2012).

To illustrate the marginalized resample–move algorithm, we conduct simulation studies in a linear Gaussian model, where the exact likelihood is obtainable from the Kalman filter, benchmarking our approach to Chopin's method (Chopin, 2002) and the current workhorse of inference, batch MCMC. The three methods deliver virtually identical outputs for a given sample size, but ours and Chopin's delivers full sequential Bayesian inference and need very little design efforts. We also apply our method to two representative real financial applications where the exact likelihood is unavailable, a Lévy jump stochastic volatility model on S&P 500 index data and a structural credit risk model with stochastic asset volatility applied to Lehman Brothers. To our knowledge, a sequential Bayesian treatment of these models is novel in the literature. We demonstrate that our algorithm delivers reliable estimates for both models and is robust to the presence of outliers in both datasets. In both cases it allows us to track the sequential changes in the parameter estimates and the Bayes factors. We find that both can change significantly, especially in turbulent times.

Section 2 presents in detail the marginalized resample–move algorithm. Section 3 illustrates this learning algorithm with a linear Gaussian model. Section 4 provides two empirical applications with a Lévy jump stochastic volatility using S&P 500 returns and a structural credit risk model taking Lehman Brothers as an example. Finally, Section 5 concludes the paper.
2. Bayesian learning in state–space models

A state–space model can be written in the following general form

\[ y_t = H(x_t, u_t, \theta), \]

\[ x_t = F(x_{t-1}, v_t, \theta), \]

where the observation \( y_t \) is assumed to be conditionally independent given the state \( x_t \) with the distribution \( p(y_t|x_t, \theta) \), the state \( x_t \) is modeled as a Markov process that lives on a space \( X \) with the initial distribution \( p(x_0|\theta) \) and the transition law \( p(x_t|x_{t-1}, \theta) \), \( u_t \) and \( v_t \) are the mutually independent observation noise and state noise and \( \theta \in \Theta \) is a set of the static parameters.

The task of state filtering and parameter learning is to find the joint posterior distribution of states and parameters based on past and current observations,

\[ p(x_t, \theta|y_{1:t}) = p(x_t|y_{1:t}, \theta)p(\theta|y_{1:t}), \]

where \( p(x_t|y_{1:t}, \theta) \) solves the state filtering problem, and \( p(\theta|y_{1:t}) \) addresses the parameter inference issue. If functions \( H(\cdot) \) and \( F(\cdot) \) are linear in \( x_t \) and \( u_t \) and if Gaussian distributions are assumed for \( x_0, u_t, v_t \) and \( \theta \), the well-known Kalman filter can be applied, and the analytical solution of state filtering \( p(x_t|y_{1:t}, \theta) \) and the exact joint posterior distribution of states and parameters \( p(x_t, \theta|y_{1:t}) \) is available. When the model is nonlinear and/or non-Gaussian, the exact and analytical solutions no longer exist, and we have to rely on approximations. Sequential Monte Carlo methods/particle methods are powerful tools to tackle such nonlinear non-Gaussian models. Their basic idea is to represent distributions of all random variables with a number of particles drawn directly from the state space and to approximate the joint posterior density \( p(x_t, \theta|y_{1:t}) \) with the empirical point-mass estimate \( \hat{p}(x_t, \theta|y_{1:t}) \)

\[ \hat{p}(x_t, \theta|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_i^{(t)} \delta\left(x_t, \theta\right) - \left(x_t, \theta\right)^{(i)} \]

where \( \tilde{w}_i^{(t)} \) is the normalized importance weight for each particle, \( \left(x_t, \theta\right)^{(i)} \) is the state and parameter particle, and \( \delta(\cdot) \) denotes the Dirac delta function. Given this particle approximation, the key problem is how to jointly sample and propagate states and parameters whenever new data arrive.

2.1. State filtering and likelihood approximation

State filtering can be efficiently implemented using particle filters given the static parameters. For notational convenience, dependence on \( \theta \) is suppressed in most of this subsection. Particle filters are a class of recursive algorithms that can be interpreted as simulation-based extensions of the Kalman Filter. Given \( M \) samples \( \{x_{t-1,i}; i = 1, 2, \ldots, M\} \) representing the filtering density \( p(x_{t-1}|y_{1:t-1}) \) at time \( t-1 \), the recursion

\[ p(x_t|y_{1:t}) \propto \int p(y_t|x_t)p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \]

suggests the following importance sampling strategy (Doucet et al., 2001). First, draw from a known and easily sampled proposal density function, \( g(x_t|x_{t-1}, y_t) \). Then, attach importance weights \( w_t \) to account for the difference between the target and the proposal,

\[ w_t^{(i)} = \frac{p(y_t|x_t^{(i)}; x_{t-1}^{(i)})}{g(x_t^{(i)}|x_{t-1}^{(i)}, y_t)}. \]

The normalized weights are given by \( \tilde{w}_t^{(i)} = w_t^{(i)}/\sum_{j=1}^{M} w_t^{(j)} \).

Finally, focus on the most likely particles by resampling the population proportional to \( \tilde{w}_t \), if the effective sample size, \( ESS = 1/\sum_{i=1}^{M} (\tilde{w}_t^{(i)})^2 \), is below some threshold. Let \( a_t \) denote the sampled indices. Thus, we obtain the equally weighted sample

\[ \hat{p}(x_t|y_{1:t}) = \frac{1}{M} \sum_{i=1}^{M} \delta\left(x_t - x_t^{(a_t)}\right). \]

Particle filters provide an estimate of the likelihood of the observations

\[ \hat{p}(y_t|y_{1:t-1}, \theta) \hat{p}(y_t|\theta), \]

where

\[ \hat{p}(y_t|y_{1:t-1}, \theta) = \frac{1}{M} \sum_{i=1}^{M} w_t^{(i)} \]

Theorem (Del Moral, 2004). The likelihood (8) approximated by particle filters is unbiased,

\[ E\left[ \hat{p}(y_t|\theta) \right] = p(y_t|\theta), \]

where the expectation is taken with respect to all the random quantities used in particle filters. \( \square \)

Proof. For a proof of this theorem, please refer to Proposition 7.4.1 of Del Moral (2004). \( \square \)

2.2. Marginalized resample-move: a generic parameter learning algorithm

While particle filters make state filtering relatively straightforward, parameter learning, i.e., drawing from \( p(\theta|y_{1:t}) \) sequentially, remains a difficult task. Simply including the static parameters in the state–space and applying a particle filter over \( p(\theta, x_t|y_{1:t}) \) does not result in a successful solution due to the time-invariance of the static parameters that quickly leads to particle depletion. In what follows, we provide a generic solution to the parameter learning problem. The method first breaks up the interdependence of the hidden states and the static parameters by approximately marginalizing out the states using a particle filter and then applies a Bayesian resample-move algorithm in the sense of Gilks and Berzuini (2001) and Chopin (2002) to this marginalized system. Our treatment here is informal and written in an accessible style. For a more rigorous argument of the validity of the method, see Chopin et al. (forthcoming).

Given a prior \( p(\theta) \) over the static parameters, it is tempting to run a recursive algorithm using the sequence of estimated densities from particle filters, \( \hat{p}(\theta|y_{1:t}) \propto \prod_{t=1}^{l} \hat{p}(y_t|y_{1:t-1}, \theta) \hat{p}(y_t|\theta)p(\theta), t = 1, \ldots, T \). However, it is not clear whether the Monte Carlo estimation errors in the likelihood invalidate the algorithm. Fortunately, to show that this is not the case, we can apply in our sequential setup an ingenious auxiliary-variable argument developed recently in the batch MCMC context by Andrieu and Roberts (2009) and Andrieu et al. (2010). The key to the construction is Eq. (10) showing that particle filters provide an unbiased estimate of the true likelihood.

Define an auxiliary state space by including all the random quantities produced by the particle filtering algorithm. In particular, denote the random quantities produced by the particle filter in step \( l \) by \( u_l = [x_l^{(i)}, a_l^{(i)}; i = 1, \ldots, M] \). Then at time \( t \), the filter will only depend on the population of the state particles in step \( t - 1 \), so we can write

\[ \psi(u_t|y_{1:t}, \theta) = \prod_{l=1}^{t} \psi(u_l|u_{l-1}, y_l, \theta)\psi(u_1|y_1, \theta), \]

where \( \psi(u_{t+1}|y_{t+1}, \theta) \) is the density of all the random variables produced by the particle filter up to \( t \). Furthermore, the predictive
likelihood of the new observations can be written as
\[ \hat{p}(y_t|y_{1:t-1}, \theta) \equiv \hat{p}(y_t|u_t, u_{t-1}, \theta). \] (12)
We then construct an auxiliary density, which has the form
\[ \hat{p}(\theta, u_{t+1}|y_{1:t}) \propto p(\theta) \prod_{t=1}^{t} \hat{p}(y_t|u_t, u_{t-1}, \theta) \]
\[ \times \psi(u_t|u_{t-1}, y_t, \theta) \hat{p}(y_t|u_t, \theta) \psi(u_{t-1}|y_t). \] (13)

The unbiasedness property in (10) means that the original target, \(p(\theta|y_{1:t})\), is the marginal distribution of the auxiliary density and they also have the same normalizing constant. If we can sequentially draw from the auxiliary density, we automatically obtain samples from the original target.

Assume that we have a set of weighted samples that represent the target distribution, \(\hat{p}(\theta, u_{t+1}|y_{1:t})\), at time \(t = 1, \ldots, n\). Denote \(u_{t=1}^{(n)}, u_{t-1}^{(n)}\) as \(M\) particles after resampling the hidden states \(\{\theta^{(i)}; i = 1, \ldots, M\}\). Therefore, in total we have to maintain \(M \times N\) particles of the hidden states. The following recursive relationship between the target distributions at \(t = 1, \ldots, t\)
\[ \hat{p}(\theta, u_{t+1}|y_{1:t}) \propto \hat{p}(y_t|u_t, u_{t-1}, \theta) \]
\[ \times \psi(u_t|u_{t-1}, y_t, \theta) \hat{p}(\theta, u_{t-1}|y_{1:t-1}). \] (14)
indicates that we can arrive at a set of samples representing the target distribution, \(\hat{p}(\theta, u_{t+1}|y_{1:t})\), at time \(t\) through the following steps:
1. **Augmentation step.** For each \(\theta^{(i)}\), run the particle filtering algorithm on the new observation, \(y_t\). This is equivalent to sampling from \(\psi(u_t|u_{t-1}^{(n)}, y_t, \theta^{(i)})\).
2. **Reweighting step.** The incremental weights are equal to \(\psi(u_t|u_{t-1}^{(n)}, \theta^{(i)})\), leading to new weights
\[ s_t^{(i)} = \frac{\psi(u_t|u_{t-1}^{(n)}, y_t, \theta^{(i)})}{\hat{p}(y_t|u_t, u_{t-1}^{(n)}, \theta^{(i)})}, \] (15)
and the estimated likelihood of the fixed parameters is updated as
\[ \hat{p}(y_t|\theta^{(i)}) = \frac{\psi(u_t|u_{t-1}^{(n)}, y_t, \theta^{(i)}) \hat{p}(y_t|u_t, u_{t-1}^{(n)}, \theta^{(i)})}{\sum_{i=1}^{n} s_t^{(i)}} \] (16)
Then, the weighted sample \(\{\theta^{(i)}, u_t^{(i)}, \hat{p}(y_t|\theta^{(i)})s_t^{(i)}; i = 1, \ldots, N\}\) is distributed according to our target \(\hat{p}(\theta, u_{t+1}|y_{1:t})\). The normalized weight is given by
\[ \pi_t^{(i)} = \frac{1}{\sum_{i=1}^{n} s_t^{(i)}} \] (17)
and the effective sample size is \(ESS_t = \frac{1}{\sum_{i=1}^{n} \pi_t^{(i)}}. \) The marginal likelihood of the new observation, essential for model comparison, can be computed as
\[ p(y_t|y_{1:t-1}) = \int p(y_t|y_{1:t-1}, \theta) \pi(\theta) d\theta \]
\[ = \int \hat{p}(y_t|u_t, u_{t-1}, \theta) \times \psi(u_t|u_{t-1}, y_t, \theta) \hat{p}(\theta, u_{t-1}|y_{1:t-1}) d(\theta, u_t, u_{t-1}) \]
\[ \approx \sum_{i=1}^{N} \pi_t^{(i)} \hat{p}(y_t|u_t^{(i)}, u_{t-1}^{(i)}, \theta^{(i)}). \] (17)

Notice that the steps so far do not enrich the set of the fixed parameters represented by the particles. As the target distribution is changing, this will lead to a gradual deterioration of the performance of the algorithm. To deal with this issue, whenever the effective sample size falls below some fixed value \(B_1\), we implement a resample-move step in the sense of Gilks and Berzuini (2001) and Chopin (2002). The resample-move approach is a hybrid of particle methods and MCMC. Here, the population is first resampled proportional to the weights to multiply particles with high probability. Then, the set of particles is enriched by passing the particles through a Metropolis–Hastings kernel that does not change the target distribution, but improves its support and diversity.

3. **Resample-move step.** If \(ESS_t < B_1\), we further consider the following steps: (1) Resample the particles proportional to \(\pi_t^{(i)}\) and provide an equally-weighted sample \(\{\theta^{(i)}, u_t^{(i)}, \hat{p}(y_t|\theta^{(i)})s_t^{(i)}; i = 1, \ldots, N\}\); (2) Move each particle through a Markov kernel with a stationary distribution \(\hat{p}(\theta, u_{t+1}|y_{1:t})\) while the number of unique particles is below some threshold \(B_2\). Here we use marginal particle MCMC kernels from Andrieu et al. (2010) with a proposal distribution of the form
\[ h(\theta, u_{t+1}|\theta') = h_t(\theta|\theta') \psi(u_{t+1}|\theta'), \] (18)
where \(h_t(\theta|\theta')\) is a proposal that can be adapted to the past of the algorithm. For example, it can be an independent multivariate normal proposal with its mean and covariance fitted to the sample posterior covariance of \(\theta\). Proposing from \(\psi(u_{t+1}|\theta)\) simply entails the running of a particle filter through the entire dataset at \(\theta\). Importantly, the random numbers used here are independent from the past of the algorithm. The acceptance probability of a new proposed particle \(\theta^*\), \(u_{t+1}^*, \hat{p}(y_t|\theta^*)\) is
\[ \min \left\{ 1; \frac{p(\theta^*) \hat{p}(y_t|\theta^*)}{p(\theta) \hat{p}(y_t|\theta)} \right\} h_t(\theta|\theta') \psi(u_{t+1}|\theta') \psi(u_{t-1}|y_t). \] (19)
Notice that we can actually obtain a joint sample from \(p(\theta, x_t|y_t)\) from our learning algorithm by drawing one particle of the hidden states for each \(\theta^{(i)}\) at any time \(t\) (see Theorem 4 of Andrieu et al. (2010)). Alternatively, by a direct application of Theorem 6 of Andrieu et al. (2010), we can use the full particle population and approximate any expectation \(E[f(\theta, x_t|y_t)]\) as
\[ E \left[ f(\theta, x_t|y_t) \right] \approx \sum_{n=1}^{N} \sum_{i=1}^{M} \pi_t^{(i)} f(\theta^{(i)}, x_t^{(i)}). \] (20)
A sequential Bayes factor can be constructed for sequential model comparison, a feature that batch estimation cannot have. For any models \(M_1\) and \(M_2\), the Bayes factor at time \(t\) has the following recursive formula
\[ BF_{t+1} \equiv \frac{p(y_t| M_1)}{p(y_t| M_2)} \equiv \frac{p(y_t|y_{1:t-1}, M_1)}{p(y_t|y_{1:t-1}, M_2)} BF_{t-1}. \] (21)
Our particle learning algorithm naturally has the marginal likelihood estimate (17), which can be used in (21) for model assessment and monitoring over time.

2.3. **Discussions**

As a rule of thumb, we can simply choose \(B_1 = B_2 = N/2\) in practice. Of course, to keep our algorithm stable, we need to ensure that the acceptance rates in the move steps do not drop too much. As shown in Andrieu et al. (2010), this will typically require the number of state particles, \(M\), that is linearly related to the largest sample size \(T\) that one wants to tackle. Chopin et al. (forthcoming) provide a proof for this statement under the sequential context and design various strategies to automatically calibrate the number of state particles over time.

It is natural to ask how our marginalized algorithm compares with the original resample-move setup of Gilks and Berzuini (2001) that acts on the joint space \((x_1, \ldots, x_T, \theta)\). The first crucial difference is in the augmentation, reweighting and resampling steps. The approach of Gilks and Berzuini (2001) would carry \(H\) particles
\[ \{x_1(1), \ldots, x_t(1), \theta(i)\}, i = 1, \ldots, H \} \] and implement the reweighting steps on this ensemble jointly. Thus the eventual sample impoverishment due to resampling occurs at the same rate on the hidden states and the fixed parameters. In contrast, our algorithm can be interpreted as maintaining \( N \) separate banks of \( M \) particles, where we allow stratified resampling steps within each bank. This allows us to minimize sample impoverishment over the fixed parameters where rejuvenation is expensive, while allowing resampling for the hidden states where the dynamic nature of the system assures automatic replenishment of the particle cloud. Second, the move step we use needs less input than \cite{Gilks2001}. Typically their procedure involves Gibbs sampling from \( p(x_1, \ldots, x_\ell|\theta) \) and \( p(\theta|x_1, \ldots, x_\ell) \), both of which necessitate the design of highly model-specific proposals. In our approach, the required proposals are obtained almost automatically. The fixed parameters are sampled by a proposal \( h_t(\theta|\theta') \) fitted on the particle population. Further, the particle filter delivers an approximate draw from the hidden states \( \bar{p}(x_1, \ldots, x_\ell|\theta) \). Another advantage of our approach is that we do not need to store the whole particle path.

We can also interpret our algorithm as an extension of \cite{Chopin2002} to dynamic systems where the likelihood \( p(y_1, t|\theta) \) is not available in closed form, but can be estimated by sequential Monte Carlo methods.

Our algorithm should be related to the method of \cite{Carvalho2010}. Their approach can be interpreted as a variant of \cite{Gilks2001} in models, where there exists a fixed dimensional statistic \( s_t = h(x_1, \ldots, x_\ell) \) that is sufficient for the fixed parameters: \( p(\theta|x_1, \ldots, x_\ell) = p(\theta|s_t) \). They propose a procedure with two important features: (1) the move over \( \theta \) does not necessitate a complete scan over the whole dataset due to the existence of low-dimensional sufficient statistics; (2) They avoid moving \( x_1, \ldots, x_\ell \), arguing that the natural rejuvenation of \( s_t \) introduced by the system dynamics suffices. The resulting algorithm is quick but may be unreliable \cite{Chopin2011}. In comparison, the marginalized resampling-move approach is more general and more robust when rejuvenation of the particle paths is important.

The move step in Step 3 is by far the most computationally expensive part of the algorithm as it requires one to run \( N \) particle filters with the number of particles \( M \) through the whole dataset with computing time increasing linearly with the sample size \( t \). Fortunately, this is counter-balanced by the fact that in practice one needs to resort to the resample-move step less and less often as \( t \) grows as the target distribution settles down. This feature of resample-move when applied to the fixed parameters in static models, has already been observed by \cite{Chopin2002} and is corroborated in our applications. A decreasing need for resample-move is also demonstrated theoretically under strong but standard assumptions in Proposition 2 in Chopin et al. (forthcoming), provided that \( M \) increases gradually with the sample size. Furthermore, our algorithm is embarrassingly parallel, so it can benefit from the massive computational power of modern graphical processor-based parallel architectures (GPU). The GPU can accelerate applications running on the CPU by offloading some of the computation-intensive and time-consuming portions of the code. The rest of the application still runs on the CPU.

\section{3. Illustration and Monte Carlo}

In this section, we implement a simulation study using a linear Gaussian model, where the exact likelihood is obtainable from the Kalman filter. This provides us with a simple benchmark.

Consider the following linear Gaussian model

\begin{align}
    y_t &= \mu + x_t + \sigma_1 e_t, \\
    x_t &= \phi x_{t-1} + \sigma_2 q_t, \\
    \mu &\text{ captures the unconditional mean of } y_t, \phi \text{ is the persistent parameter of the state } x_t, (\phi, \eta) \sim N(0, I), \text{ and } x_0 \sim N(0, \sigma_2^2/1 - \phi^2). \end{align}

The efficiency of our approach depends on how accurate the likelihood can be approximated from particle filters. In implementation, we apply both the bootstrap filter of \cite{Gordon1993} and a fully-adapted particle filter for state filtering and likelihood estimation. The bootstrap filter directly takes the transition law of state as the proposal distribution. However, the adapted particle filter works differently by taking into account the latest information. At each time \( t \), we first resample with respect to \( w^{(t)}_i \propto p(y_t|x^{(t)}_{t-1}, \theta) = N(m_t, s_t) \), where \( s_t = 1 - \sigma_2^2 - \sigma_2^2 \), and \( m_t = s_t(\mu - \eta_t - \mu) + \sigma_2^2 \).

We generate a sequence of observations with the simple size \( T = 1000 \) using the true parameters \( \theta^* \) = \((0.5, 1.0, 0.8, 0.75)\). Fig. 1 presents the efficient sample sizes and acceptance rates at move steps when using the bootstrap filter (left panels) and the adapted filter (right panels). In both cases we work with an independent multivariate normal \( M \)-H proposal fitted on the particle set. The number of parameter particles is \( N = 500 \) while the number of state particles is \( M = 100 \). The priors are given by \( \theta_0 \sim N(\theta^*_0, \Sigma^*_0) \), where \( \theta^*_0 = (0.30, 1.5, 0.50, 0.50) \) and \( \Sigma_0 = diag(0.50, 1.0, 0.70, 0.65) \). Any non-positive values of \( \sigma_1 \) and \( \sigma_2 \), and any \( \phi \neq (-1, 1) \) are automatically discarded. The truncated priors are normalized to 1 to make sure to have the correct marginal likelihood estimate. We can clearly see that the algorithm takes much more move steps in the case of using the bootstrap filter, causing a very large computational cost. We also notice that the acceptance rate is decreasing to a very small value when the sample size becomes large in the bootstrap filter. However, when using the adapted filter, the move-step becomes less and less often, and the acceptance rate remain high (larger than 30%) over time. These results indicate that the likelihood approximated from the adapted filter is much more accurate than that approximated by the bootstrap filter.

The exact likelihood of this model can be easily obtained from the Kalman filter. Thus, the exact likelihood-based method \cite{Chopin2002} is implementable. This provides us a simple benchmark to evaluate the extra Monte Carlo error due to the particle filter. Fig. 2 illustrates Chopin’s learning method (right panels) and our parameter learning algorithm equipped with the adapted filter (left panels) using the same sequence of simulated data. Again, an independent multivariate normal \( M \)-H proposal is fitted on the particle set. The number of parameter particles is 500 and the number of state particles is 100. Looking at the efficient sample sizes, both methods have a decreasing rate of resampling, signaling that both are good at discarding uninteresting parts of the parameter domain at the beginning of the sample. The lower panel reports that both methods have stable acceptance rates, but they demonstrate different trends. Chopin’s method has acceptance rates approaching one as the sample grows, a result of the joint normal proposal and the asymptotic normality of the target. In contrast, the acceptance rates of marginalized resample-move begin to decrease somewhat as the sample size becomes large. This is the natural result of the increasing estimation error of the likelihood. To fight this phenomenon, based on the results of \cite{Andrieu2010}, we typically need to increase \( M \) linearly with \( T \).

While Chopin’s method directly uses the exact likelihood, our approach uses a particle filter to approximate the likelihood. Therefore, the selection of numbers \( M \) and \( N \) may affect the performance of our algorithm. Table 1 presents parameter learning results at final date \( T \) in our algorithm using different values of \( M \) and \( N \) and in Chopin’s method for the same simulated sequence of data as in this case all variations in the results can be attributed.
Fig. 1. Diagnostics: Bootstrap filter vs. Adapted filter.  
Note: The figure plots the sequential effective sample sizes (ESS) and acceptance rates in the marginalized resample-move algorithm (MRM Learning). The left panels are for the case of using the bootstrap filter, and the right panels are for the case of using the adapted filter. The learning algorithms are implemented on a sequence of simulated data, which is generated from the linear Gaussian model with the sample size $T = 1000$ using the true parameter set $\theta^* = (0.5, 1.0, 0.8, 0.75)$. The priors are the same as in Table 1.

Fig. 2. Diagnostics: MRM learning vs. Chopin learning. Note: The figure plots the sequential effective sample sizes (ESS) and acceptance rates in the marginalized resample-move algorithm (MRM Learning, left panels) and in Chopin’s method (right panels). The learning algorithms are implemented on a sequence of simulated data, which is generated from the local level model with the sample size $T = 1000$ using the true parameter set $\theta^* = (0.5, 1.0, 0.8, 0.75)$. The priors are the same as in Table 1.
to the Monte Carlo error in the particle filter. We choose the number of state particles (M) equal to 50, 100, or 200 and the number of parameter particles (N) equal to 500, 1000, or 2000. The table provides means and standard errors across 100 independent runs both for the posterior means and the estimates of the marginal likelihood. Using the noisy likelihood does increase the Monte Carlo variability of the estimate compared to Chopin (2002) but it does not introduce any systematic bias confirming the validity of the method.

In Table 2, we double-check that our algorithm provides reliable results even when we average over different data samples. Here we simulate 100 different samples of size T = 1000 by setting the number of state particles (M) equal to 100 and the number of parameter particles (N) equal to 500. In addition to Chopin’s method, we also report results from a batch MCMC method using the exact likelihood obtained by using the Kalman filter instead of the usual two-block MCMC estimation. The same priors as before are used in these three estimations. The results show very similar mean estimates and root mean square errors (RMSE). Thus, our learning method can provide credible estimates.

4. Empirical applications

4.1. A Levy jump stochastic volatility model

In this section, we provide an empirical application for a Lévy jump stochastic volatility model using S&P 500 index returns. Section 4.1.1 introduces a time-changed Lévy model, and Section 4.1.2 presents the empirical results.

4.1.1. A time-changed Lévy model

Under a given probability space (Ω, ℱ, ℙ) and the complete filtration (ℱₜ)₀≤t<∞, the asset price Sₜ has the following dynamics under the objective measure

\[
\ln Sₜ/S₀ = \int_0^t uₜ ds + \left( Wₜ - \kappa_0 (1) T \right) + ( Jₜ - \kappa_1 (1) t ),
\]

where \( uₜ \) captures its instantaneous mean, and \( \kappa_0 (1) \) and \( \kappa_1 (1) \) are convexity adjustments for the Brownian motion and the jump process, respectively, and can be computed from their cumulant exponents: \( \kappa(u) = \frac{1}{2} \ln(E[e^{u J}]) \), where \( L \) is either \( W \) or \( J \).

The asset price jumps due to the jump component, \( Jₜ \), in (24). The inclusion of jumps is important for capturing the large discontinuous movements in asset returns. Many studies have shown that it can help generate the return non-normality and can explain the implied volatility smile/skew at short horizons. Recent nonparametric studies by Ait-Sahalia and Jacod (2009), Cont and Mancini (2011), and Lee and Hannig (2010) provide strong and convincing evidence of infinite activity jumps in asset prices. Therefore, in this paper, we model the jump component using the Variance Gamma process (Madan et al., 1998), which is an infinite activity stochastic process and can be constructed through subordinating a Brownian motion with drift using an independent subordinator

\[
Jₜ = \alpha \xi + \eta \tilde{W}(\xi),
\]

where \( \tilde{W} \) is a standard Brownian motion and \( \xi \) is a subordinator that is the Gamma process \( \xiₜ = \Gamma(t;1,\nu) \) with unit mean rate and variance rate of \( \nu \).

\( T \) defines a stochastic business time (Clark, 1973; Carr et al., 2003), which captures the randomness of the diffusion variance over a time interval \([0,1]\):

\[
Tᵢ = \int_0^t Vᵢ ds
\]

which is finite almost surely. \( Vᵢ \), which should be nonnegative, is the instantaneous variance rate, reflecting the intensity of

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Simulation study in the linear Gaussian model.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( N = 500 )</td>
</tr>
<tr>
<td>( \mu )</td>
<td>( \sigma_1 )</td>
</tr>
<tr>
<td>A. MRM learning: ( M = 50 )</td>
<td>0.485</td>
</tr>
<tr>
<td>StDev</td>
<td>0.013</td>
</tr>
<tr>
<td>B. MRM learning: ( M = 100 )</td>
<td>0.484</td>
</tr>
<tr>
<td>StDev</td>
<td>0.011</td>
</tr>
<tr>
<td>C. MRM learning: ( M = 200 )</td>
<td>0.484</td>
</tr>
<tr>
<td>StDev</td>
<td>0.009</td>
</tr>
</tbody>
</table>

Note: The table presents the parameter learning results and normalizing constant (NC) estimates at the final date \( T \) in the marginalized resample-move method (MRM Learning), and in Chopin’s method. In the MRM algorithm, we choose the number of state particles \( M \) equal to 50, 100, or 200 and the number of parameter particles \( N \) equal to 500, 1000, or 2000. The learning algorithms are implemented on a sequence of simulated data, which is generated with the sample size \( T = 1000 \) using the true parameter set \( \sigma_0 = (0.5, 1.0, 0.8, 0.75) \). The priors are given by \( \theta_0 \sim \mathcal{N}\left(0, \Sigma_0\right) \), where \( \Sigma_0 = (0.30, 1.5, 0.50, 0.50) \) and \( \Sigma_0 = \text{diag}\left(0.50, 1.0, 0.70, 0.65\right) \). Any non-positive values of \( \sigma_1 \) and \( \sigma_2 \), and any \( \phi \neq (−1, 1) \) are automatically discarded.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Simulation study over different datasets in a linear Gaussian model.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \mu )</td>
</tr>
<tr>
<td>True value</td>
<td>0.50</td>
</tr>
<tr>
<td>A. MRM learning: ( M = 100 )</td>
<td>Mean</td>
</tr>
<tr>
<td>Median</td>
<td>0.511</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.138</td>
</tr>
<tr>
<td>B. Exact likelihood-based learning (Chopin, 2002)</td>
<td>Mean</td>
</tr>
<tr>
<td>Median</td>
<td>0.505</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.124</td>
</tr>
<tr>
<td>C. Exact likelihood-based MCMC</td>
<td>Mean</td>
</tr>
<tr>
<td>Median</td>
<td>0.496</td>
</tr>
<tr>
<td>RMSE</td>
<td>0.117</td>
</tr>
</tbody>
</table>

Note: The table presents the Monte Carlo simulation results for the linear Gaussian model. The number of simulations is 100, and the sample size of each simulated dataset is \( T = 1000 \). The model is estimated by the proposed learning algorithm, Chopin’s exact likelihood-based learning method, and the exact likelihood-based MCMC method. The priors are the same as in Table 1. For the learning algorithm, we set the number of state particles 100 and the number of parameter particles 500, and for MCMC, we set the number of iterations 10000, among which the last 6000 are kept for inference.
The instantaneous variance rate is modeled using the square-root process of Cox et al. (1985):

$$dV_t = \kappa (\theta - V_t)dt + \sigma \sqrt{V_t}dZ_t,$$

where $\kappa$ is the mean-reverting parameter, $\theta$ is the long-run mean parameter, $\sigma$ is the volatility parameter, $Z_t$ is a standard Brownian motion that is correlated to $W_t$ with a correlation parameter $\rho$ in order to accommodate the diffusion leverage effect and is independent of $J_t$. If we suppress the jump component, $J_t$, in (24), the model becomes the Heston stochastic volatility model (Heston, 1993).

The model can be cast into a state–space model framework. After discretizing the asset price process for a time interval $\tau$ using the Euler method, we have the following observation equation

$$\ln S_{t+\tau} = \ln S_{t-\tau} + \left(\mu - 0.5V_{t-\tau} - k(1)\tau\right)\tau + \sqrt{\tau V_{t-\tau}}w_{t},$$

where $w_t$ is a standard normal noise, and $J_t$ is the jump noise. We take diffusion variance $V_t$ and jump $J_t$ as the hidden states. Diffusion variance follows (27) and jumps have the dynamics in (25). The Euler approximation to the state process then yields

$$V_t = \kappa \theta \tau + (1 - \kappa \tau)V_{t-\tau} + \sigma \sqrt{\tau V_{t-\tau}}Z_t,$$

$$J_t = \omega \delta_t + \eta \sqrt{\tau V_{t-\tau}}\tilde{w}_t,$$

$$\delta_t = N(\tau; 1, \nu),$$

where $Z_t$ is a standard normal noise, which is correlated to $w_t$ in (28) with the correlation parameter $\rho$, and $\tilde{w}_t$ is an independent normal noise. The empirical studies in finance so far have found that the daily frequency is fine enough to make biases due to the discretization error in the simple Euler approximation inconsequential (Eraker et al., 2003; Li et al., 2008; Johannes et al., 2009). To ascertain that this holds in our application, a Monte Carlo study is conducted in Appendix A to compare the daily discretization with a finer one.

### 4.1.2. Parameter learning using S&P 500 index

We model the S&P 500 index using the model discussed in Section 4.1.1. The data used range from January 2, 1981 to December 31, 2010 and are in daily frequency, in total 7,570 observations. This dataset includes typical market behaviors such as the recent financial crisis in the late 2008, the dot-com bubble burst in 2002, the Asian financial crisis in 1998 and the market crash in 1987. Fig. 3 plots the time-series evolution of the S&P 500 index and index returns.

When implementing our parameter learning algorithm, we need to design an efficient particle filter in order to accurately approximate the likelihood function. The most commonly used particle filter is the bootstrap filter, which simply takes the state transition density as the proposal density. However, the bootstrap filter is known to perform poorly when the observation is informative on the hidden states. Our model has this feature because when we observe a large move in asset price, the jump can be largely pinned down by the observed return. On the other hand, when the return is small, it is mostly due to the diffusion component and contains little information on the jump. Hence, to provide an efficient sampler, we use an equally weighted two-component mixture as the proposal on the jump: the first component is a normal draw, equivalent to sampling from the transition density of the diffusion component, and the second component involves drawing from the transition law of the jump. That is, the proposal density of the jump is $\pi(j_t | \ln S_t, \ln S_{t-\tau}, V_{t-\tau}) = 0.5\pi_1(j_t | \ln S_t, \ln S_{t-\tau}, V_{t-\tau}) + 0.5\pi_2(j_t | \delta_t)$, where

$$\pi_1 \rightarrow N(\ln \delta_t - \ln \delta_{t-\tau} - (\mu - 0.5V_{t-\tau} - k(1))\tau, \tau V_{t-\tau}),$$

$$\pi_2 \rightarrow N(\omega \delta_t, \eta^2 \delta_t),$$

and $\delta_t$ and $V_t$ are drawn from their transition densities. In implementation, we use $N = 2000$, $M = 8000$, and $B_1 = B_2 = N/2$. Note that we need to choose a large number for the state particles because of our long sample, making the computing needs of our model massive. Here the parallel nature of our algorithm becomes crucial as we can tremendously speed up the algorithm by using a graphical processing unit (see Appendix B for details on the GPU implementation).
We estimate the above Variance Gamma stochastic volatility model and one of its nested models, the Heston stochastic volatility model. For estimation of the Heston model, we simply shut down the jump component from the Variance Gamma model in our algorithm. Estimation needs about one day for the general model while it is much faster for the Heston model. Batch Bayesian inference of the Heston model has been done by several authors, among them Fruhwirth-Schnatter and Sogner (2008), while Li et al. (2008) estimate the Variance Gamma model using an MCMC method. Fig. 4 presents the efficient sample size and the acceptance rate of the algorithm for the two models. In particular, during the financial crisis in 1987, the ESS in the Heston model drops to a very low level. The acceptance rate on average is higher in the Variance Gamma model than in the Heston model. Fewer move steps and a higher acceptance rate in the Variance Gamma model indicate that it can adapt better to the outliers.

Table 3 (Panel A and B) presents the posterior mean and 90% credible interval of parameter estimates at the final date for two models. In the jump model, we find that the posterior mean of $\omega$ is negative and its 90% credible interval is narrow and resides in the negative half-line, indicating that index returns jump downwards more frequently than jump upwards. The jump structure parameter $\nu$ has a posterior mean of about 1.5 and a 90% credible interval of [0.52, 2.6], implying that in general, small/tiny jumps happen with a very high frequency and large/huge jumps occur only occasionally. The mean-reverting parameter $\kappa$ of the diffusion volatility process is a little bit larger in the Heston model (3.71 vs. 3.15 when fat tails are allowed in the conditional distribution). In contrast, estimates of $\theta$ and $\sigma$ are very similar across the two specifications. $\rho$, which is about $-0.52$ in the Heston model and about $-0.62$ in the jump model, reveals the existence of the leverage effect. Not surprisingly, the log marginal likelihood (LMLH) strongly favors the presence of jumps ($2.476 \times 10^4$ vs. $2.468 \times 10^4$ in Heston).

Our learning approach provides us more than parameter estimates themselves. It gives us the whole picture of how parameters evolve over time with respect to accumulation of information. Fig. 5 presents the sequential learning of parameters in the jump model. Clearly, all parameter estimates vary a lot at the beginning when information is scarce, to stabilize as the sample gets larger. Importantly, some of the parameters can exhibit sudden moves at big outliers and the rate of information accumulation as measured by the credible intervals is not always uniform.

Fig. 6 presents the filtered volatility and jumps. The filtered volatility varies substantially over time, and the 90% credible interval is very tight, indicating existence of time-varying/stochastic volatility in the equity market. It reaches its highest value during the recent financial crisis. The filtered jumps can efficiently capture all turmoils observed in the market, in particular, during the 1987 market crash.

Fig. 7 presents the log sequential Bayes factors for comparison between the jump model and the Heston model over time. We notice that in the beginning before the 1987 market crash, both models perform nearly the same as the log Bayes factors vary around zero. However, starting from the 1987 market crash, we clearly see the superiority of the jump model as the Bayes factor abruptly move up to a high level and stay there onwards.

As suggested by Durham and Geweke (2011), we implement 15 independent runs of the algorithm for our preferred model (the jump model) to have a notion of Monte Carlo variability of the resulting estimates. The panel C in Table 3 presents means and standard deviations of the posterior mean and the 5th and 95th quantiles across these runs. Clearly, we see that all standard deviations are small, and in particular, the mean of the marginal likelihood estimates is about $2.476 \times 10^4$, but its standard deviation is only 1.45.

4.2. A structural credit risk model

4.2.1. A structural credit risk model with stochastic asset volatility

Merton (1974) laid the foundation to the literature on the structural approach to credit risk modeling. However, his model assumes constant asset volatility that may be overly restrictive,
especially during turbulent periods. For this reason, we investigate a more general specification where the asset volatility follows a square root process as in Heston (1993). The value of the firm at time $t$, $V_t$, is assumed to follow a diffusion with stochastic volatility with respect to the physical probability measure ($\Omega$, $\mathcal{F}$, $P$) and the complete filtration $\mathcal{F}_t$,

$$dV_t/V_t = \mu dt + \sigma_t dW_{t,1},$$

(34)

where $W_t$ is a standard Brownian motion. The variance process follows a square root process,

$$d\sigma^2_t = \kappa (\theta - \sigma^2_t) dt + \sigma_t \sigma dW_{t,1}.$$

(35)

For simplicity, we assume a zero correlation between $W_{t,1}$ and $W_{t,2}$, an assumption that could easily be relaxed. Note however, that the model already produces a negative correlation between equity innovations and equity volatility changes due to changes in leverage (leverage effect). The risk-free rate of interest, $r$, is assumed to be a constant. Furthermore, the firm has two classes of claims outstanding, an equity and a zero-coupon debt maturing at time $T$ with face value $F$. Due to limited liability, equity is a call

Table 3
Parameter estimates for equity price models.

<table>
<thead>
<tr>
<th>Model</th>
<th>$\mu$</th>
<th>$\omega$</th>
<th>$\eta$</th>
<th>$\kappa$</th>
<th>$\theta$</th>
<th>$\sigma$</th>
<th>$\rho$</th>
<th>LMLH</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Heston model</td>
<td>0.040</td>
<td>–</td>
<td>–</td>
<td>3.709</td>
<td>0.029</td>
<td>0.316</td>
<td>–0.522</td>
<td>$2.468 \times 10^4$</td>
</tr>
<tr>
<td>0.05 Qf</td>
<td>0.019</td>
<td>–</td>
<td>–</td>
<td>3.182</td>
<td>0.026</td>
<td>0.303</td>
<td>–0.569</td>
<td>–</td>
</tr>
<tr>
<td>0.95 Qf</td>
<td>0.070</td>
<td>–</td>
<td>–</td>
<td>4.587</td>
<td>0.031</td>
<td>0.333</td>
<td>–0.449</td>
<td>–</td>
</tr>
</tbody>
</table>
| B. Variance Gamma model | Mean   | 0.065    | 0.037  | 0.038     | 1.524    | 3.145    | 0.210  | 0.310     | $2.467 \times 10^4$
|                       | 0.05 Qf | 0.014    | 0.071  | 0.013     | 0.517    | 2.363    | 0.283  | –0.669    | –          |
|                       | 0.95 Qf | 0.108    | 0.017  | 0.062     | 2.619    | 3.953    | 0.332  | –0.560    | –          |
| C. Variance Gamma model (multiple runs) | Mean | 0.054 | 0.044 | 0.034 | 1.282 | 3.278 | 0.282 | 0.318 | $-0.610$ | $2.467 \times 10^4$
|                       | 0.05 Qf | 0.016 | 0.068 | 0.010 | 0.497 | 2.485 | 0.024 | 0.287 | $-0.662$ | –          |
|                       | 0.95 Qf | 0.092 | 0.020 | 0.058 | 2.346 | 4.146 | 0.012 | 0.351 | $-0.553$ | –          |

Note: The table reports the parameter estimates (posterior mean and central 90% credible interval) for the Lévy jump model using the marginalized resample-move method. In estimation, we set the number of state particles 8000 and the number of parameter particles 2000. The priors are given by $\theta_0 \sim N(\theta^*_0, \Sigma^*_0)$, where $\theta^*_0 = (0.10, -0.05, 0.10, 1.00, 3.00, 0.03, 0.30, -0.50)$ and $\Sigma^*_0 = \text{diag}(0.20, 0.08, 0.15, 3.00, 5.00, 0.05, 0.30, 0.50)$. Any non-positive values of $\eta$, $\kappa$, $\theta$, and $\sigma$ and any $\rho \notin (-1, 1)$ are automatically discarded.
option on the value of the firm with payout

\[ S_T = \max(V_T - F, 0). \]  

Then, as shown in Heston (1993), the equity claim in Eq. (36) can be priced at time \( t \) by

\[ S_t = S(V_t; \sigma_t^2, F, r, T - t) = V_t P_t - F e^{-r(T-t)} P_0, \]  

where

\[ P_t = 0.5 + \frac{1}{\pi} \int_{u=0}^{\infty} \Re \left\{ \frac{\exp(-iuF)}{iu} \right\} \exp(C_i + D_i \sigma_i^2 + iu \ln V_t) \, du, \quad i = 1, 2, \]

where \( u \in R \) is the characteristic index and \( D_i \) and \( C_i \) are known functions of the model parameters.

Unfortunately, the asset value of the firm, \( V_t \), is rarely observable. In contrast, for an exchange listed firm, one can obtain a time series of equity prices. We thus can infer the asset value using the equity prices and balance sheet information on debt. Further, the observed equity prices may be contaminated by microstructure noise that can be important, especially for smaller firms or firms in financial difficulties. Following Duan and Fulop (2009), the trading noise obeys a multiplicative structure leading to the following measurement equation for the log equity price

\[ \log S_t = \log S(V_t; \sigma_t^2, F, r, T - t) + \delta V_t, \]  

where \( V_t \) are i.i.d. standard normal random variables and the nonlinear pricing function \( S(V_t; \sigma, F, r, T - t) \) has been given earlier. The first order Euler approximation to the system yields the following transition equation for the state variables:

\[ \log V_t = \log V_{t-1} + \left( \mu - \frac{1}{2} \sigma_t^2 \right) \tau + \sigma_{t-1} \sqrt{\tau} \varepsilon_{1,t}, \]

\[ \sigma_t^2 = \sigma_{t-1}^2 + \kappa (\theta - \sigma_t^2) \tau + \sigma_{t-1} \sqrt{\tau} \varepsilon_{2,t}, \]

where \( \varepsilon_{1,t} \) and \( \varepsilon_{2,t} \) are independent standard normals.

To obtain efficient estimates of the likelihood, it is crucial in this model to use an adapted sampler that takes the information in the new observation into account. In our model, we employ a slight extension of the localized filter of Duan and Fulop (2009). Assume that we have particles \( \sigma_{i,t}^2 \) and \( \log V_{i,t} \) for \( i = 1, \ldots, M \) representing the filtering distribution at \( t = \tau \). Then the proposal distribution takes the following steps:

- We use the bootstrap sampler to draw for the next value of volatility, \( \sigma_{t}^{(i)} \sim p(\sigma_{t}^{2} | \sigma_{t-1}^{2}) \). This choice is simple and reasonably efficient as a single observation is not too informative on the volatility state.
- Given the sampled value of volatility, the localized filter of Duan and Fulop (2009) would involve simulating from the error \( v_{i}^{(t)} \) and solving the equation \( S_t e^{-\delta v_{i}^{(t)}} = S(V_t; \sigma_{t}^{2}, F, r, T - t) \) for \( V_t \). Instead of solving this nonlinear equation, we have found that an approximate inversion does a good job in locating the proposal around the new observation while necessitating fewer calls to the expensive pricing function.\(^1\)

\(^1\) This basically involves iterating a few times the equity pricing equation (37) by using its special form. In particular, we iterate over temporary asset values, by
4.2.2. Learning using Lehman Brothers pre-bankruptcy returns

Merton’s model has been estimated in batch setups by Duan and Fulop (2009) using MLE and by Huang and Yu (2010) using MCMC. The same stochastic volatility model is used in Duan and Fulop (2012) for a comparison of our sequential algorithm with their density-tempered approach in a batch setup. We implement our learning methodology for both models using Lehman’s data before its bankruptcy on September 15, 2008. In particular, our data sample from CRSP covers the period between January 1, 2007 and August 29, 2008, resulting in a sample size of 416.

Fig. 8 depicts the daily equity returns and suggests the presence of outliers, an unsurprising result given the turbulence in this period. We choose the maturity of debt to be equal to 3 years and use the 3-month constant maturity treasury yield from the St. Louis FED website as our risk-free rate.

We still need a proxy of the face value of debt, F. There is no consensus in the literature on how to compute this number from the balance sheet of the firm. The importance of estimating F for financials is pointed out in Duan (2010). We treat F as an unknown and use the current practice in forming our prior. In particular, we have a uniform prior over F with a lower bound equal to the KMV estimate of debt (current liabilities +0.5 long term debt) and an upper bound equal to total liabilities. On the other parameters we choose non-informative priors. For the asset drift we choose a normal prior with parameters \( \mu \sim N(0, 0.05) \). The remaining parameters have a uniform prior with the following upper and lower bounds: \( (\theta, \kappa, \sigma_V, \delta) \rightarrow U((0.001^2, 0.1 \times 10^2), (1 \times 10^6), (0.2^2, 20, 2, 0.05)) \). For comparison we have also estimated Merton’s model by setting \( \kappa = 0 \), \( \sigma_V = 0 \). Estimation needs about one and half hours for the stochastic volatility model and about one hour for the Merton’s model. See Appendix B for details on the GPU implementation.

In our generic algorithm, move steps with a single move are triggered whenever the ESS drops below 500. In this dataset we have found that to stabilize the results, more moves need to be taken whenever the ESS drops below 400, a sign of the arrival of outliers. The reason is that in such cases, after the reweighting with the outlier, the resulting particle cloud does not give a good enough description of the target. Fortunately a few move steps quickly result in a much more reliable particle ensemble (in particular, in such cases we have executed 6 move steps).

Fig. 9 reports the efficient sample size and the acceptance rate of the algorithm for the two model specifications. Looking at the ESS, one can observe a big drop in March 2008 for both models. This corresponds to a cluster of outliers with big negative returns followed by a large positive return. The algorithm recovers from this by taking several move steps in both specifications. One can observe however that the model allowing stochastic volatility makes fewer resample-move steps afterwards, a sign that it can adapt better to the new more turbulent data. Next, looking at the acceptance rates, we observe some important differences. For Merton’s model acceptance rates are high throughout, a sign of the efficiency of the localized filter we use. In contrast, in the stochastic volatility model the acceptance rates decrease a lot after March 2008. This is a result of the increased estimation error of the likelihood due to the outliers. Intuitively, filtering is a much harder task in this model, as the latent asset volatility is much less tied down by the data than the asset values.

Table 4 (panel A and B) reports the full-sample parameter posterior means together with the 5th and 95th percentiles of the posterior distribution for both Merton’s model and the stochastic volatility model. The results are in accord with common sense. In particular, we find from Panel B that all parameters controlling the stochastic volatility have a tight 90% credible interval. The log marginal likelihood in the stochastic volatility model is much larger than in Merton’s model (828.76 vs. 747.38). Fig. 10 reports the sequential posterior mean and the central 90% credible interval of the parameters in the preferred model and Fig. 11 shows the sequential estimates of the filtered asset volatility. As usual in finance, the data is not very informative on the expected return \( \mu \). Further, we observe an interesting pattern for \( \kappa \), the degree of mean reversion of volatility. Here, the arrival of information is uneven with a slow accumulation in the first half of the sample followed by a sudden shrinkage of the posterior at the outliers of March 2008. \( \theta \), \( \sigma_V \), and \( \delta \) are all well identified and quite stable. The estimated face value of debt, F, is much closer to its lower bound, the KMV estimate. Further, the value of the asset volatility varies strongly through time, with much larger values at the end of the sample. This underlines the importance of allowing for dynamics in asset volatility. As before, recursive model comparison can be done using the sequential Bayes factor. Fig. 12 shows that the stochastic volatility model is overwhelmingly preferable to Merton’s model after the fourth quarter of 2007.

Again, we implement 15 independent runs of our algorithm for the preferred model (the SV model) to see the Monte Carlo variability. Similar results as before can be found in Panel C of Table 4. Nearly all standard deviations of the posterior means and the 5th and 95th quantiles are pretty small. The mean of the log marginal likelihood is 828.85 with a standard deviation 0.38.

5. Concluding remarks

In this paper we propose the marginalized resample-move algorithm, a generic approach to sequential Bayesian inference over the dynamic states and the fixed parameters in general state–space models. The main idea of the methodology is to approximately marginalize out the hidden states and then to apply the resample-move algorithm to the marginalized system. The main effort required from users is to design a particle filter for the system, and the additional steps required for parameter learning are almost automatic. Different from batch estimation, the proposed learning method provides a tool to monitor and evaluate models in real time and can fully take into account impacts of parameter and model uncertainties on decision-making over time.
We have demonstrated that one can obtain reliable results for models and sample sizes representative in finance. Furthermore, the algorithm is embarrassingly parallel, so it can benefit from the massive computational power of modern graphical processor-based (GPU) parallel architectures. Given the relatively light design effort, our methodology seems to have a wide applicability to sequential inference problems in economics and finance. Data sizes up to a couple of thousands can already be routinely tackled by today’s technology, and the frontier is likely to shift rapidly given the speed of change in GPU-based scientific computing.

Appendix A. Monte Carlo study on discretizations

In both empirical applications, we first adopt the Euler method to discretize the models and then implement our learning algorithm. Here we conduct a Monte Carlo simulation to show that the Euler method with daily frequency does not introduce significant bias in estimation.

We take the Variance Gamma stochastic volatility model as an example. For each simulation, we generate a sequence of daily observations with sample size 3000 using a finer discretization scheme \( \Delta t = \frac{1}{252} \) to make sure that the data are generated from the true continuous-time model. The total number of simulations is 50. The true values of parameters are \( \Theta^* = (0.10, -0.05, 0.08, 1.00, 4.00, 0.03, 0.25, -0.50) \), which are close to the empirical estimates in Section 4.1.2. In estimation, two discretization schemes are taken into consideration: daily frequency and one-fifth of the daily frequency.

Table 5 presents the Monte Carlo simulation results. The following findings are in order. First, for each parameter, the means,
Fig. 10. Learning in the credit risk model with stochastic asset volatility.
Note: This Figure presents the posterior mean and central 90% credible interval of the recursive estimates of the model parameters in the structural credit risk model with stochastic volatility, implemented on Lehman Brothers. In estimation, we set the number of state particles 400 and the number of parameter particles 1000. The priors are $\mu \sim N(0, 0.05)$ and $(\theta, \kappa, \sigma_V, \delta) \sim U([0.001^2, 0, \times 10^2, 1 \times 10^2], (0.2^2, 20, 2, 0.05))$.

Fig. 11. Filtered asset volatility in the credit risk model with stochastic asset volatility.
Note: This Figure presents the posterior mean and central 90% credible interval of the recursive estimates of the stochastic asset volatility, $V_t$, in the structural credit risk model, implemented on Lehman Brothers.

medians and RMSE’s from both discretization schemes are very similar. While the RMSE’s for $\omega, v$ and $\rho$ are smaller in the one-fifth daily frequency scheme, the RMSE’s for $\eta, \kappa$ and $\sigma$ are smaller in the daily frequency scheme. Second, as for the log marginal likelihood estimate (LMLH), both schemes deliver very similar values (about $9.28 \times 10^3$) with nearly the same standard deviations (about $1.35 \times 10^3$). Third, the algorithm can deliver quite accurate estimates for most of the parameters in both scheme even though we use the data with small sample size. Fourth, the estimate of $v$

is not significant in each of these two schemes. This is due to the small sample issue as the relatively large value of $v$ generates a very small number of sizable jumps in each simulation.

The above findings ascertain that the discretization error from the Euler method with daily frequency is inconsequential. These results are consistent with those found by Eraker et al. (2003), Li et al. (2008), and Johannes et al. (2009).

Appendix B. Details of GPU implementations

In both of our empirical applications, we use GPU’s to speed up the computations. We program in MATLAB the main algorithm and
In the case of the Variance Gamma stochastic volatility model, this amounts to porting the whole particle filtering mechanism onto the GPU, coded in CUDA. We find that the use of double-precision floating numbers is necessary to achieve stable results. Obtaining the likelihood estimates in our algorithm entails running N particle filters concurrently, each with M state particles. In what follows, we describe how to parallelize important steps of the algorithm:

- We use the NAG GPU toolbox to generate normal and uniform random numbers simultaneously for the N × M particles. See Bradley et al. (2010) for details on the implementation.
- Proposal generation and importance weight calculation do not necessitate interaction between particles and are done parallelly over the N × M particles.
- The resampling step necessitates the interaction of state particles for a given fixed parameter set Θ. Here, one thread runs all the necessary calculations for a given Θ, allowing us to run N filters at the same time. The steps are:
  - Form a recursive sum of the importance weights. This step also provides the likelihood estimate of the new observation as a by-product.
  - Sample indices of the resampled particles using stratified resampling. Note that this step differs from the bisect search approach proposed in Lee et al. (2010) which allows them to find the sampled index of each particle independently. The reason is that in contrast to them we can parallelize the algorithm in the parameter dimension and we find this level of parallelism sufficient.
  - Repopulate the particle vector from the proposal using the sampled indices.

In contrast, in the structural credit risk model, the huge majority of the computational load is concentrated on the evaluation of the equity formula. This is a situation that could be likely shared in a wide range of applications, for example, in any empirical study using derivative data lacking a closed-form solution. Here to economize on the coding effort, we find it to be efficient to keep the particle filtering algorithm on the CPU and only call the GPU for the evaluation of the pricing function. Further, in contrast to the previous case, we find very good speedups from using the Jacket Toolbox for MATLAB, allowing for fast code development. One of the main differences between the two applications is that computing the option prices can be done by pure vectorized operations without the use of branching commands and indexing into the matrices, whereas these two features are unavoidable for the proposal generating stage and resampling stage of the particle filtering algorithm in the Variance Gamma model. As a result, the ability to directly access the memory allocation and data movement in the lower level CUDA implementation carries a larger speed premium in the latter case.

References


