8-2014

A flexible and automated likelihood based framework for inference in stochastic volatility models

Hans J. SKAUG
University of Bergen

Jun YU
Singapore Management University, yujun@smu.edu.sg

Citation

SKAUG, Hans J. and YU, Jun. A flexible and automated likelihood based framework for inference in stochastic volatility models. (2014). *Computational Statistics and Data Analysis*. 76, 642-654.
Available at: https://ink.library.smu.edu.sg/soe_research/1615

Published in Computational Statistics & Data Analysis, August 2014, Volume 76, Pages 642–654.
http://dx.doi.org/10.1016/j.csda.2013.10.005

A flexible and automated likelihood based framework for inference in stochastic volatility models

Hans J. Skaug [a,*], Jun Yu [b]

[a] *Department of Mathematics, University of Bergen, P.O. Box 7800, 5020 Bergen, Norway*

[b] *Sim Kee Boon Institute for Financial Economics, School of Economics, Lee Kong Chian School of Business, Singapore Management University, 90 Stamford Road, Singapore 178903, Singapore*

A B S T R A C T

The Laplace approximation is used to perform maximum likelihood estimation of univariate and multivariate stochastic volatility (SV) models. It is shown that the implementation of the Laplace approximation is greatly simplified by the use of a numerical technique known as automatic differentiation (AD). Several algorithms are proposed and compared with some existing maximum likelihood methods using both simulated data and actual data. It is found that the new methods match the statistical efficiency of the existing methods while significantly reducing the coding effort. Also proposed are simple methods for obtaining the filtered, smoothed and predictive values for the latent variable. The new methods are implemented using the open source software AD Model Builder, which with its latent variable module (ADMB-RE) facilitates the formulation and fitting of SV models. To illustrate the flexibility of the new algorithms, several univariate and multivariate SV models are fitted using exchange rate and equity data.

Keywords:
Empirical Bayes
Laplace approximation
Automatic differentiation
AD Model Builder
Simulated maximum likelihood
Importance sampling

1. Introduction

Maximum likelihood (ML) estimation of nonlinear and non-Gaussian state space models has recently received a great deal of attention in the statistics literature as well as in the econometrics literature. A leading example of nonlinear and non-Gaussian state space models in financial econometrics is the class of stochastic volatility (SV) models for which the likelihood function is expressed by a high dimensional integral that cannot be evaluated analytically due to the presence of a latent volatility process. As a result, ML estimation is computationally demanding. While in the earlier literature some statistically inefficient but numerically simple methods have been proposed (e.g. Andersen and Sorensen, 1996; Harvey et al., 1994), ML remains appealing partly because of its statistical efficiency and partly because computing power is rapidly improving.

Various ML methods have been proposed in the literature. Contributions include Shephard and Pitt (1997), Durbin and Koopman (1997), Sandmann and Koopman (1998), Liesenfeld and Richard (2003, 2006), Durham (2006) and Kleppe and Skaug (2012). The idea in these papers is to evaluate the likelihood function numerically by integrating out the latent volatility process via the Laplace approximation or importance sampling techniques, followed by numerical maximization of the approximate likelihood function.

Many of the algorithms for these ML methods have been implemented in low level programming languages such as C++ or FORTRAN in order to increase the computational speed. For example, the importance sampler of Durham (2006) was implemented using FORTRAN. While these specially tailored packages are numerically efficient, the effort required to write, debug and modify the code is usually large. In addition, to use the method of Durham (2006), one has to find the analytical

* Corresponding author. Tel.: +47 55 58 48 61; fax: +47 55589672.
E-mail addresses: skaug@math.uib.no (H.J. Skaug), yujun@smu.edu.sg (J. Yu).

expressions for the first and second order derivatives of the joint log density of returns (observations) and volatilities (latent random variables), which are required for the Laplace approximation.

One purpose of this paper is to illustrate several algorithms for performing ML estimation of SV models using automatic differentiation (AD), combined with the Laplace approximation and importance sampling. We also present a simple empirical Bayes approach to estimation of volatilities, conditionally on the returns (observations), both as a filter and as a smoother, as well as a predictor. We then demonstrate the ease by which univariate and multivariate SV models can be explored using the latent variable module of the open source software package *AD Model Builder* (Fournier et al., 2011), referred to as *ADMB* in short. The ADMB software is available from http://admb-project.org. Historically, the latent variable module of ADMB was a separate program, but is today fully integrated with the rest of ADMB. It nevertheless has its own user manual, and we shall refer to it here as ADMB-RE. AD is a technique for calculating the exact numerical derivatives of functions defined as computer algorithms (Gallant, 1991), and should not be confused with symbolic differentiation as performed by for instance MATHEMATICA and MAPLE. As both the Laplace approximation and numerical likelihood optimization require derivatives, ADMB-RE facilitates parameter estimation and is an ideal software to do ML estimation for nonlinear and non-Gaussian state-space models.

Our paper is related to Meyer et al. (2003) where the likelihood function of the basic SV model was approximated by the Laplace approximation via AD. The focus of their paper was, however, to develop an efficient MCMC algorithm. Moreover, our method for approximating the likelihood function is different from theirs. Meyer et al. (2003) used a Kalman filter approach, where a sequence of one-dimensional Laplace approximations was used to perform the one-step updates of the Kalman filter. We apply a single multivariate Laplace approximation jointly with respect to all volatilities. While it is trivial to generalize our joint Laplace approximation to multivariate SV models, the same does not seem to be the case for the sequential univariate normal approximation of Meyer et al. Our work is also related to Liesenfeld and Richard (2006) where efficient importance sampling was used to perform both ML analysis and Bayesian analysis of SV models. The main difference between our algorithms and theirs is the different importance sampling techniques used. Finally, papers that are similar to ours in the use of a joint Laplace approximation are Durham (2006) and Martino et al. (2011), but neither of these papers use AD, and hence lack the ease and flexibility of implementation of our approach.

The rest of the paper is organized as follows. Section 2 introduces the Laplace approximation and discusses how to use it to perform classical maximum likelihood (ML) estimation. Section 3 explains how the Laplace approximation facilitates smoothing, filtering and predicting the latent variable. Section 4 describes AD and the software ADMB-RE. In Section 5 we examine the relative performance of ADMB-RE using both simulated data and actual data and demonstrate how more flexible univariate and multivariate SV models can be fitted under ADMB-RE. Section 6 concludes. Information about how to obtain the ADMB-RE code for the model developed in this paper is provided in Appendix A.

2. Maximum likelihood estimation

For illustrative purposes, we focus on the so-called basic SV model which is defined by

$$\begin{cases} X_t = \sigma_X e^{h_t/2}\epsilon_t, \quad t = 1, \ldots, T, \\ h_{t+1} = \phi h_t + \sigma\eta_t, \quad t = 1, \ldots, T-1, \end{cases} \tag{1}$$

where X_t is the return of an asset, $\epsilon_t \overset{\text{i.i.d.}}{\sim} N(0,1)$, $\eta_t \overset{\text{i.i.d.}}{\sim} N(0,1)$, $\text{corr}(\epsilon_t, \eta_t) = 0$, and $h_1 \sim N(0, \sigma^2/(1-\phi^2))$. The parameters of interest are $\theta = (\sigma_X, \phi, \sigma)'$.

Let $\mathbf{X} = (X_1, \ldots, X_T)'$ and $\mathbf{h} = (h_1, \ldots, h_T)'$. The likelihood function of the basic SV model is given by

$$p(\mathbf{X}; \theta) = \int p(\mathbf{X}, \mathbf{h}; \theta) d\mathbf{h} = \int p(\mathbf{X}|\mathbf{h}; \theta) p(\mathbf{h}; \theta) d\mathbf{h}. \tag{2}$$

This is a high-dimensional integral which does not have a closed form expression due to the non-linear dependence of X_t on h_t, and hence must be approximated.

Following Skaug (2002), our first algorithm (termed LA-ML) employs the Laplace approximation, that is, we match $p(\mathbf{X}, \mathbf{h}; \theta)$ and a multivariate normal distribution for $\mathbf{h}$ as closely as possible (up to a constant proportion). More precisely, the Laplace approximation to the integral (2) is

$$p(\mathbf{X}; \theta) \approx |\det(\Omega)|^{-1/2} p(\mathbf{X}, \mathbf{h}^*; \theta), \tag{3}$$

where

$$\mathbf{h}^* = \arg\max_{\mathbf{h}} \ln p(\mathbf{X}, \mathbf{h}; \theta) \quad \text{and} \quad \Omega = \frac{\partial^2 \ln p(\mathbf{X}, \mathbf{h}^*; \theta)}{\partial \mathbf{h} \partial \mathbf{h}'}. \tag{4}$$

The Laplace approximation is exact when $p(\mathbf{X}, \mathbf{h}; \theta)$ is Gaussian in $\mathbf{h}$. It typically works well when $p(\mathbf{h}; \theta)$ is Gaussian or nearly Gaussian and $p(\mathbf{h}; \theta)$ is more informative about $\mathbf{h}$ than $p(\mathbf{X}|\mathbf{h})$ is, in the sense of observed Fisher information. This is indeed the case for almost all the SV models used in practice. From an empirical viewpoint, the normality of $\mathbf{h}$ is documented as one of the stylized facts about volatility in Andersen et al. (2001). Theoretical results about the accuracy of

the Laplace approximation, easily applicable in the present context, are lacking in the literature. It is worth noting that the accuracy of the approximation may vary with the value of the parameter θ. It is interesting to investigate the accuracy of the approximation for the problem at hand and this will be done in Section 5.

For SV models $\mathbf{h}^*$ does not have a closed form expression. To find $\mathbf{h}^*$, a numerical optimization method, such as Newton's method, is needed. While Ω may have a closed form expression, it is typically tedious and error prone to derive this expression by hand. Durham (2006) suggests using symbolic programs to calculate Ω. We will evaluate Ω using AD.

In the case where the Laplace approximation does not work satisfactorily, one can improve the approximation by using importance sampling, which is a Monte Carlo approach to high-dimensional numerical integration. The likelihood can be written as

$$p(\mathbf{X};\theta) = \int p(\mathbf{X},\mathbf{h};\theta)d\mathbf{h} = \int \frac{p(\mathbf{X},\mathbf{h};\theta)}{q(\mathbf{h})}q(\mathbf{h})d\mathbf{h}, \tag{5}$$

where $q(\mathbf{h})$ is the importance density. The idea of importance sampling is to draw samples $\mathbf{h}^{(1)},\ldots,\mathbf{h}^{(S)}$ from $q(\mathbf{h})$ so that we can approximate $p(\mathbf{X};\theta)$ by

$$\frac{1}{S}\sum_{i=1}^{S}\frac{p(\mathbf{X},\mathbf{h}^{(i)};\theta)}{q(\mathbf{h}^{(i)})}. \tag{6}$$

In general, $q(\cdot)$ depends on both $\mathbf{X}$ and θ. The law of large numbers ensures the convergence of (6) to $p(\mathbf{X};\theta)$ as $S\to\infty$.

For the method to be computationally efficient, we should match $p(\mathbf{X},\mathbf{h};\theta)$ and $q(\mathbf{h})$ as closely as possible while ensuring that it is still easy to sample from $q(\mathbf{h})$. To do that, following Shephard and Pitt (1997) and Durbin and Koopman (1997), we choose q to be the Laplace approximation to $p(\mathbf{X},\mathbf{h};\theta)$. That is $\mathbf{h}^{(s)}\sim N(\mathbf{h}^*,-\Omega^{-1})$ where $\mathbf{h}^*$ and Ω are from (4). It should be noted that $q(\mathbf{h})$ depends on θ through $\mathbf{h}^*$ and Ω. When maximizing the likelihood approximation (6), smoothness of the θ-surface is ensured by using "common random numbers", i.e. $\mathbf{h}^{(i)}=\mathbf{h}^*+\left(-\Omega^{-1}\right)^{1/2}\epsilon^{(i)}$, where $\left(-\Omega^{-1}\right)^{1/2}$ is the matrix square root (Cholesky decomposition) and $\epsilon^{(i)}$ is a vector of independent standard normal variables (common to all values of θ). Skaug and Fournier (2006) provide a formula for the gradient of the likelihood approximation (6) in the case that the importance density is the Laplace approximation.

While the importance sampling procedure used in the present paper is based on the Laplace approximation, other important sampling techniques exist in the literature. For example, the importance sampler developed by Liesenfeld and Richard (2003, 2006) and others is based on a particular factorization of the importance density. The importance sampling densities used in Liesenfeld and Richard (2003, 2006) differ from those of Shephard and Pitt (1997) and Durbin and Koopman (1997) in the sense that they are based on "global" (as opposed to "local") approximations to the integrand. Lee and Koopman (2004) compared the performance of these two approaches in the context of univariate SV models and found the two methods to perform equally well. In this paper, the method that maximizes the likelihood function obtained from simulations is called simulated ML (SML).

3. Smoothing, filtering and predicting latent variables

In this section, we develop several algorithms for obtaining the filtered, smoothed and forecasted values of the latent variable, all based on the Laplace approximation. A major advantage of the proposed algorithms is the ease of implementation, requiring little programming effort, and low computational cost. This is in sharp contrast to other filtration methods available in the literature, including the unscented Kalman filter of Li (2013).

We first propose the algorithm for obtaining the smoothed estimate of the latent variable, with and without taking into account the two sources of error discussed above. Denote by $\hat{\theta}$ the ML estimate of θ. A natural smoother of $\mathbf{h}$ is the empirical Bayes estimator

$$\hat{\mathbf{h}}^s = \arg\max_{\mathbf{h}} \ln p(\mathbf{X},\mathbf{h};\hat{\theta}). \tag{7}$$

That is, we use the mode of the conditional density $p(\mathbf{h}|\mathbf{X})$ to estimate $\mathbf{h}$, conditionally on $\mathbf{X}$. This is because $\ln p(\mathbf{X},\mathbf{h}) = \ln p(\mathbf{h}|\mathbf{X}) + \ln p(\mathbf{X})$, and the latter quantity does not depend on $\mathbf{h}$. It should be pointed out that the smoother (7) comes as a by-product of the Laplace approximation.

By ignoring errors in parameter estimation, we get the approximate covariance matrix of the smoothed estimates of the latent variable,

$$-\left[\frac{\partial^2 \ln p(\mathbf{X},\hat{\mathbf{h}}^s;\hat{\theta})}{\partial\mathbf{h}\partial\mathbf{h}'}\right]^{-1}. \tag{8}$$

A variance formula that takes the uncertainty in $\hat{\theta}$ into account is

$$-\left[\frac{\partial^2 \ln p(\mathbf{X},\hat{\mathbf{h}}^s;\hat{\theta})}{\partial\mathbf{h}\partial\mathbf{h}'}\right]^{-1} + \frac{\partial\hat{\mathbf{h}}^s}{\partial\theta}\Sigma_{\hat{\theta}}\left(\frac{\partial\hat{\mathbf{h}}^s}{\partial\theta}\right)', \tag{9}$$

where the expression for the Jacobian matrix $\partial\hat{\mathbf{h}}^s/\partial\theta$ is given in Skaug and Fournier (2006). This formula may be derived in different ways. In a Bayesian context a similar formula was given in Kass and Steffey (1989), but the particular version (9) was independently discovered by D. Fournier, and this is also the version that is implemented in ADMB-RE.

The smoothed estimate of $\mathbf{h}$ can be used to generate model diagnostics. For example, an estimate of η_t is given as $(\hat{h}^s_{t+1} - \hat{\phi}\hat{h}^s_t)/\hat{\sigma}$. Some care is needed in the interpretation of the estimated η_t, because although the η_t are independent, the smoothed values will not be (Harvey and Koopman, 1992). Similarly, one can estimate ϵ_t by $y_t \exp(-0.5\hat{h}^s_t)/(\hat{\sigma}_X)$.

Prediction of future values of the latent variable can easily be done within this framework by extending the $\mathbf{h}$ vector with as many time periods as needed. If we want a K-step prediction, let $\mathbf{h} = (h_1, \ldots, h_T, h_{T+1}, \ldots, h_{T+K})'$. For the augmented $\mathbf{h}$ vector, the optimization problem (7) yields both the smoothed values $\hat{h}^s_1, \ldots, \hat{h}^s_T$ and the predicted values $\hat{h}^s_{T+1}, \ldots, \hat{h}^s_{T+K}$. Under the AR(1) structure for the latent variable, it is clear that $\hat{h}^s_{T+K} = \phi^K \hat{h}^s_T$, $K > 0$, but for more general latent processes an explicit expression may not exist.

It seems natural that θ is estimated using an un-augmented $\mathbf{h}$, and that prediction is conducted after the model has been fitted. However, we point out that from a practical perspective it is convenient to use the augmented $\mathbf{h}$ also during the estimation phase. The reason is that no extra programming is required in order to obtain the prediction of $(h_{T+1}, \ldots, h_{T+K})$ and its covariance matrix, which is given by (9). As long as K is small compared to T the computational overhead is negligible. But we need to argue that augmenting $\mathbf{h}$ does not change the Laplace approximation (3) of the likelihood, and hence not the estimate of θ. This follows from general properties of the Laplace approximation. To see why, consider the following re-parametrization of the latent variables: $\mathbf{u} = (h_1, \ldots, h_T, \eta_T, \ldots, \eta_{T+K-1})'$, where $\eta_t = h_{t+1} - \phi h_t$ for $t = T, \ldots, T + K - 1$. This is a linear transformation, and it can be proved that the Laplace approximation (3) is invariant under a one-to-one linear transformation of the latent variables. Further, $\mathbf{u}_1 = (h_1, \ldots, h_T)'$ and $\mathbf{u}_2 = (\eta_T, \ldots, \eta_{T+K-1})'$ are independent, also when we condition on $\mathbf{X}$. For a conditional distribution with the property that $p(\mathbf{u}|\mathbf{X}) = p_1(\mathbf{u}_1|\mathbf{X})p_2(\mathbf{u}_2|\mathbf{X})$ we have a corresponding factorization of the Laplace approximation (3). But, since $p_2(\mathbf{u}_2|\mathbf{X}) = p(\eta_T, \ldots, \eta_{T+K-1})$ is Gaussian the Laplace approximation of $\int p_2(\mathbf{u}_2)d\mathbf{u}_2 = 1$ exactly, and it follows that inclusion of $\eta_T, \ldots, \eta_{T+K-1}$ does not affect the likelihood of θ.

Another important quantity is the filtered estimate of the latent variable, that is the estimate of h_t conditional on $X_1, \ldots, X_t$, given the parameter value θ. Numerous nonlinear filtering algorithms have been proposed. We propose an alternative filtered estimate of h_t based on the Laplace approximation. In particular, we define

$$(\hat{h}^f_1, \ldots, \hat{h}^f_t)' = \arg\max_{h_1,\ldots,h_t} \ln p(X_1, \ldots, X_t, h_1, \ldots, h_t; \hat{\theta}), \tag{10}$$

where $\hat{\theta}$ is the estimate of θ obtained from the entire sample $\{X_t\}_{t=1}^T$. The filtered estimate of h_t is $\hat{h}^f_t$. Obviously, it involves the repeated evaluation of the Laplace approximation but is straightforward to program after the estimation based on the full sample is completed.

4. Automatic differentiation and ADMB-RE

The need for calculating derivatives arises in two places in the proposed algorithms. First, the numerical solution of the optimization problem (4) is greatly simplified if the exact gradient of the objective function is available. Second, as noted above, the Hessian matrix (4) occurring in the Laplace approximation is typically too complicated to be evaluated by hand.

Automatic differentiation (AD) is a technique to numerically evaluate the derivatives of a function specified by a computer program (Gallant, 1991). It exploits the fact that every computer program, no matter how complicated, executes a sequence of elementary arithmetic operations such as additions or elementary functions. By applying the chain rule repeatedly to these operations, derivatives of arbitrary order can in principle be computed automatically, and accurate to computer precision.

Two classical ways of evaluating derivatives are symbolic differentiation and the method of finite differences. While both AD and symbolic differentiation utilize the chain rule, it is important to emphasize differences between the two methods. The result of symbolic differentiation is a mathematical formula, while AD is aimed at producing numerical derivatives. Clearly, the mathematical derivative formula may be evaluated numerically, so there does not seem to be much to be gained. However, the real distinction is that AD may be applied to entire computer programs. Application of symbolic differentiation here is possible at a sub-expression level, but would involve quite a bit of manual labor. Two important drawbacks with finite differences are round-off errors in the discretization process, and cancellation. These problems multiply when calculating higher order derivatives. Automatic differentiation calculates exact (to machine precision) numerical derivatives and hence is not subject to the same problems.

Two different approaches to AD exist, the forward mode and the reverse mode. They differ in how the chain rule is used to propagate derivatives through the computation. The forward mode propagates derivatives of intermediate variables with respect to the independent variables. In contrast, the reverse mode of AD propagates derivatives of the final result with respect to all intermediate variables in the program. The program is first evaluated without derivative calculations, and subsequently derivatives are calculated in reverse order of the original program flow. During the latter phase the program must keep track of the values of all intermediate variables that impact the final result, and thus the reverse algorithm may be very memory consuming (Gallant, 1991).

```
DATA_SECTION
  init_int n                      // Length of time series
  init_vector X(1,n)              // Time series
PARAMETER_SECTION
  init_number phi
  init_number sigma
  init_number sigma_x
  random_effects_vector h(1,n)  // Vector of latent variables
  objective_function_value g
PROCEDURE_SECTION
  int i;                          // Local variables
  double log_2pi = 0.9189385;
  dvariable phi2 = square(phi);

  // Contribution from log[p(h_1)]
  g -= -0.5*log_2pi -log(sigma) + 0.5*log(1-phi2)
          - 0.5*square(h(1)/sigma*(1-phi2));

  // Contribution from log[p(h_t|h_{t-1})]
  for (i=2;i<=n;i++)
    g -=  -0.5*log_2pi  -log(sigma)
            -.5*square((h(i)-phi*h(i-1))/sigma);

  // Contribution from log[p(X_t|h_t)]
  for (i=1;i<=n;i++)
  {
    dvariable sigma_X = sigma_x*exp(0.5*h(i));
    g -= -0.5*log_2pi -log(sigma_X) - 0.5*square(X(i)/sigma_X);
  }
```

Fig. 1. ADMB code for the basic SV model (1). Appendix B gives details about how to compile and run the program. At the end of the PARAMETER_SECTION the objective function holds the value $g = -\log[p(\mathbf{X}|\mathbf{h},\theta)p(\mathbf{h}|\theta)]$. The ADMB code that can be found in a link provided in Appendix A includes some additional directives, omitted for presentation purposes here, but which are needed to achieve the CPU times reported in the present paper.

AD Model Builder (ADMB) is an open source statistical modeling framework based on C++ that combines AD with a quasi-Newton function minimizer (Fournier et al., 2011). An example of an ADMB program is given in Fig. 1, and Appendix B provides details about how to compile and run this program. When compiled and executed, the program will minimize the objective function (the negative log-likelihood) with respect to the independent parameters, i.e. those variables defined in the first part of the PARAMETER_SECTION. The use of AD to calculate the gradient of the objective function makes ADMB fast and numerically stable.

The latent variable module ADMB-RE, usually referred to as the random effects module, allows the Laplace approximation (3) to be calculated automatically, i.e. with the Hessian matrix in (4) calculated by automatic differentiation. The keyword `random_effects_vector` (see Fig. 1) instructs the compiler that the vector $\mathbf{h}$ should be the target for the Laplace approximation. The objective function in the program must be taken to be $g(\mathbf{h},\theta) = -\log[p(\mathbf{X}|\mathbf{h},\theta)p(\mathbf{h}|\theta)]$. In order to facilitate the numerical optimization of the log-likelihood based on (3), ADMB-RE calculates the third order mixed derivatives of $g(\mathbf{h},\theta)$ by automatic differentiation (Skaug and Fournier, 2006). Optionally, ADMB-RE employs the importance sampling approximation (6) with the density q chosen as explained at the end of Section 2. Appendix A provides some ADMB commands, the file structure and the sample ADMB code for the basic SV model.

5. Performance of the proposed algorithms

5.1. Fitting the basic SV model to real data

For the purposes of illustration and comparison, we first fit the basic SV model to a widely used exchange rate data series and then to an equity data series. The exchange rate series consists of 945 observations on daily return of pound/dollar exchange rate from 01/10/1981 to 28/06/1985 (Fig. 2) while the equity data series consists of 1598 observations on daily return of S&P500 index from 02/01/2007 to 10/05/2013. The same exchange rate data were also used in Harvey et al. (1994), Shephard and Pitt (1997), Meyer and Yu (2000) and Meyer et al. (2003).

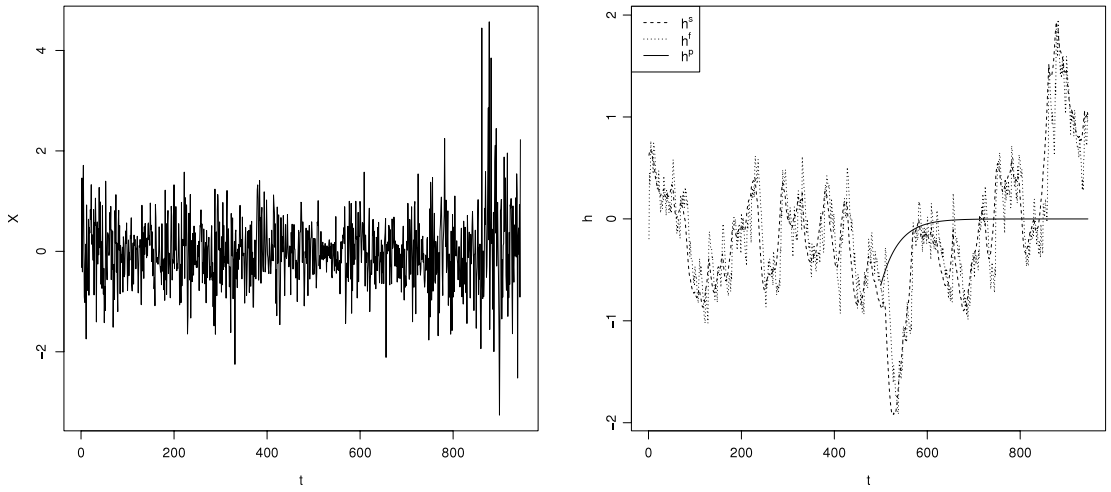

Fig. 2. Basic SV model fit to pound/dollar exchange rate data (t in days since 01/10/1981). Left: raw data X_t. Right: smoothed (h^s), filtered (h^f) and predicted (h^p) volatility process obtained using the Laplace approximation for the exchange rate data (see Table 1). The prediction is made at $t = 500$ using data up to $t = 500$ both in the estimation of parameters and in the prediction of h.

Table 1
The basic SV model fitted to the daily returns of pound/dollar exchange rate data with different likelihood approximations (method).

Method	Parameter	Estimate	SE	CPU(s)	Log-Likelihood
LA-ML	ϕ	0.9743	0.0122	9.78	−918.791
	σ	0.1697	0.0363		
	σ_X	0.6330	0.0688		
SML	ϕ	0.9748	0.0122	14.53	−918.669
($S = 64$)	σ	0.1687	0.0355		
	σ_X	0.6337	0.0697		
SML	ϕ	0.9737	0.0120	15.98	−918.363
($S = 128$)	σ	0.1738	0.0360		
	σ_X	0.6311	0.0694		
SML	ϕ	0.9741	0.0120	19.71	−918.594
($S = 256$)	σ	0.1719	0.0370		
	σ_X	0.6315	0.0680		

For the exchange rate data, the estimates and asymptotic standard errors based on the observed Fisher information matrix for ϕ, σ, σ_X, the log-likelihood values for the LA-ML method and the SML method with different values of S, are reported in Table 1. Also reported in Table 1 is the CPU time in seconds, on a Pentium IV 3.2 GHz PC running on WINDOWS XP, which is the time-to-convergence of the optimizer.

It can be seen that the ML methods can obtain parameter estimates within seconds and the two sets of estimates, standard errors and log-likelihood values are very close to each others, indicating that the error associated with the Laplace approximation is very small in this case. Since LA-ML is not simulation-based, not surprisingly it is the fastest method, taking less than 10 s of CPU time.

To assess the performance of the SML methods, we have tried two approaches. Firstly, we increase the number of the importance samples from 64 to 128 and then to 256. Table 1 reports the results when $S = 128$ and $S = 256$. While large values of S increase the computational cost, the CPU time is still within seconds. Importantly, the two sets of estimates, standard errors and log-likelihood values are very close to those when $S = 64$, suggesting that the SML method is reliable.

Secondly, for the SML method with the different values of S, we calculate the Monte Carlo standard errors (MCSE) of the SML estimates by repeating the estimation for 100 different sets of common random numbers in the importance sampling. Table 2 reports the empirical mean and standard deviation across the resulting 100 sets of parameter estimates and log-likelihood values. Note that MCSE includes Monte Carlo variation only, and not the sampling variation in the parameter estimates. Hence, if the approximation error of a SML method is small, we expect the MCSEs to be much smaller than the asymptotic standard errors. This is indeed the case as the MCSE ranges between 0.0008 and 0.0052 for the parameter estimates and between 0.1701 and 0.3442 for the log-likelihood value. In accordance with the basic properties of averages it appears that MCSE for the log-likelihood goes down at a rate $S^{-1/2}$.

Smoothed (h^s), filtered (h^f) and predicted (h^p) values of the volatility process are shown in Fig. 2. Although apparently similar, the filtered and smoothed solutions differ on average by 21% at an exponential scale, i.e. $\{\exp(h_s) - \exp(h_f)\}/\exp(h_s)$.

Table 2
Evaluation of Monte Carlo standard error (MCSE) of the various SML estimators of Table 1. Average point estimates (Estimate) and MCSE are calculated across 100 different sets of common random numbers.

Method		Estimate	MCSE
SML	ϕ	0.9742	0.0014
($S = 64$)	σ	0.1713	0.0052
	σ_X	0.6316	0.0011
	Log-Lik	−918.612	0.3442
SML	ϕ	0.9742	0.0008
($S = 128$)	σ	0.1709	0.0038
	σ_X	0.6317	0.0010
	Log-Lik	−918.647	0.2660
SML	ϕ	0.9740	0.0008
($S = 256$)	σ	0.1721	0.0035
	σ_X	0.6313	0.0008
	Log-Lik	−918.623	0.1701

Table 3
The basic SV model fitted to the daily returns of S&P500.

Method	Parameter	Estimate	SE	MCSE	CPU (s)	Log-Likelihood
LA-ML	ϕ	0.9836	0.0058		38.82	−2504.13
	σ	0.1932	0.0235			
	σ_X	1.0458	0.1519			
SML	ϕ	0.9835	0.0058	0.00030	43.28	−2503.74
($S = 128$)	σ	0.1949	0.0235	0.00227		
	σ_X	1.0458	0.1524	0.00034		

Further, the prediction, which is that of an autoregressive process of order 1 is unable to predict the dip in h values that occur around $t = 550$.

For S&P500 index, the estimates, asymptotic standard errors and MCSEs for σ_X, ϕ, σ, the log-likelihood values, and the CPU time, for the LA-ML method and the SML method with $S = 128$, are reported in Table 3. The results for SML are calculated from 100 different sets of common random numbers. Both LA-ML and SML take less than one minute of CPU time. The two sets of estimates, asymptotic standard errors and log-likelihood values are very close to each other. The MCSEs for SML estimates are very small.

5.2. Flexible modeling

A major difficulty in most existing algorithms and computing softwares is that any change in the model specification entails a substantial effort in careful algebraic derivations (such as finding the full-conditional distributions and finding the expressions of the first two derivatives) followed by a serious effort in coding and debugging. One exception is BUGS, as demonstrated in Meyer and Yu (2000) and Yu and Meyer (2006) in the context of SV models. As in BUGS, modification of the model is straightforward in ADMB-RE and usually only involves changing a few lines of code. To illustrate the strength and flexibility of ADMB-RE, we consider some variations over the basic SV model and one multivariate SV model. Appendix C provides details about the modifications of the ADMB-RE code.

First, we consider two univariate SV models, namely, t-SV model, and SV model with the leverage effect. We fit both models to the S&P500 series using LA-ML and SML. The so-called t-SV model is obtained by assuming that ϵ_t in the basic SV model (1) has a Student-t distribution with $\nu > 2$ degrees of freedom (also to be estimated). To introduce a leverage effect into the basic SV model (1), we assume that ϵ_t and η_t are $N(0, 1)$ with $\text{corr}(\epsilon_t, \eta_t) = \rho$. A reparameterization of the model (similarly to Yu, 2005) yields

$$\begin{cases} X_t = \sigma_X e^{h_t/2} \left(\dfrac{\rho}{\sigma}(h_{t+1} - \phi h_t) + \sqrt{1-\rho^2}\, w_t \right), \\ h_{t+1} = \phi h_t + \sigma \eta_t, \end{cases} \tag{11}$$

where w_t is $N(0, 1)$ and $\text{corr}(\eta_t, w_t) = 0$. Note that $\epsilon_t = \rho\eta_t + \sqrt{1-\rho^2} w_t$.

Table 4 reports the estimates, asymptotic standard errors, MCSEs for all parameters, the CPU time and the log-likelihood values for the two univariate models, obtained from the two ML methods. For SML, we choose $S = 128$. The results for the basic SV model are also reported for the purpose of comparison. First, it can be seen that both models can be quickly fitted by LA-ML, with the resulting estimates and log-likelihood being almost identical to those resulting from SML. The CPU time increases when the model specification becomes more complicated, as expected. However, the computational cost remains low. Second, both estimation procedures suggest that the model extensions improve the fit over the basic model, judged

Table 4
Parameter estimates for LA-ML, SML and MCMC (WinBUGS) for three univariate SV models fitted to the S&P500 series. The numbers given in parentheses are standard errors for ML and posterior standard errors for MCMC. The numbers in square brackets are MCSEs.

	Basic	Leverage-SV	t-SV
LA-ML			
ϕ	0.9836(0.0058)	0.9695(0.0058)	0.9872(0.0050)
σ	0.1932(0.0235)	0.2450(0.0252)	0.1706(0.0225)
σ_X	1.0458(0.1519)	1.1245(0.074)	0.9369(0.1527)
ρ		−0.7846(0.048)	
ν			9.1969(2.736)
CPU (s)	38.82	89.02	55.81
Log-lik	−2504.13	−2465.320	−2497.37
SML ($S = 128$)			
ϕ	0.9835(0.0058)[0.00030]	0.9695(0.0058)[0.00037]	0.9869(0.0051)[0.00012]
σ	0.1949(0.0235)[0.00227]	0.2449(0.0252)[0.00229]	0.1738(0.0231)[0.00095]
σ_X	1.0458(0.1524)[0.00034]	1.1247(0.075)[0.00047]	0.9382(0.1518)[0.00014]
ρ		−0.7825(0.049)[0.00359]	
ν			9.2617(2.753)[0.0952]
CPU (s)	43.28	743.23	125.03
Log-lik	−2503.737[0.3937]	−2465.236[0.4102]	−2496.95[0.1319]
MCMC			
ϕ	0.9845(0.0057)[0.00017]	0.9702(0.0058)[0.00022]	0.9883(0.0050)[0.00016]
σ	0.1921(0.0232)[0.00113]	0.242(0.0255)[0.00126]	0.1661(0.0232)[0.00118]
σ_X	1.055(0.6162)[0.00235]	1.127(0.0804)[0.00112]	0.9636(1.064)[0.00427]
ρ		−0.7605(0.055)[0.00261]	
ν			10.39(2.898)[0.1249]
CPU (s)	3013	3814	4681

by the increase in the log-likelihood value, which is larger than 3.32, the 1% critical value for the χ^2 test. Also note that the difference in the log-likelihood values is large compared to the MC uncertainty associated with the SML log-likelihood values (square brackets). This is especially true for the SV model with the leverage effect, with the estimated ρ being nearly −0.8. In the t-SV model, the estimated degrees of freedom is $\nu \approx 9$, providing some evidence of fat-tails in ϵ_t.

Table 4 reports Bayesian MCMC results obtained from WinBUGS (Spiegelhalter et al., 2003) which is one of the most popular implementations of BUGS. Unlike ML methods, Bayesian procedures require prior distributions to be specified for all parameters. For the basic SV model these priors were taken from Meyer and Yu (2000). Prior mean and standard deviation (in parenthesis) are ϕ: 0.86(0.11), σ: 0.12(0.050) and σ_X: $1 \cdot 10^5$ $(3 \cdot 10^6)$. For the leverage model we use a prior with ρ: 0(0.58) and for the t-SV model we use a prior with ν: 63(36.37). These are, with the exception of σ_X, slightly informative priors, relative to the information content in the data. Following Meyer and Yu (2000) we let WinBUGS generate 100,000 MCMC iterations from each model, after having discarded the initial 10,000 iterations to ensure that the chain has reached its equilibrium distribution. The remaining 100,000 samples pass the Geweke convergence test. When comparing the posterior means in Table 4 produced by WinBUGS to our ML estimates, the main conclusion is that both sets of point estimates and standard deviations correspond well. It is not clear whether the differences in the point estimates are due to skewness in the posterior distribution or influence from the prior. The exception is σ_X for the t-SV, for which the SE exceeds that of the ML estimates seven times. The explanation is that the posterior for σ_X is highly skewed, but also that there is considerable MC error in the SE value reported in Table 4 for WinBUGS.

The CPU times for WinBUGS in Table 4 are in the range 40–80 times longer than those for ADMB-RE. Note, however, that no attempt has been made to find the minimum number of iterations required for WinBUGS to satisfy the Geweke convergence criterion. However, the fact that the MCSE of parameter estimates are comparable between ADMB-RE and WinBUGS indicates that our choice of 100,000 MCMC iterations may be reasonable. The rationale being that, although the two methods return different quantities (ML estimates for ADMB-RE and posterior means for WinBUGS) it is reasonable to require that the Monte Carlo error around the target value be the same in order for the comparison to be "fair". The MCSE for WinBUGS is calculated within a single MCMC chain, taking serial correlation into account.

To further illustrate the flexibility of our proposed algorithms and ADMB-RE, we fit a multivariate SV model to a dataset that has been used in the literature using LA-ML and SML with $S = 128$. It consists of 945 daily observations on three exchange rates from 01/10/1981 to 28/06/1985: Pound/Dollar, Deutschmark/Dollar and Yen/Dollar. The same data were used in Harvey et al. (1994) and Liesenfeld and Richard (2006). As in BUGS, multivariate SV models can be fit with a dozen lines on code in ADMB-RE. The multivariate model used here is taken from Harvey et al. (1994)

$$\begin{cases} \mathbf{X}_t = \exp(0.5\mathtt{diag}(\mathbf{h}_t))\epsilon_t, \ \epsilon_t \sim N(0, \Sigma), \\ \mathbf{h}_{t+1} = \mu + \Phi(\mathbf{h}_t - \mu) + \eta_t, \ \eta_t \sim N(0, \Sigma_\eta), \end{cases} \tag{12}$$

Table 5
Parameter estimates (Est.), asymptotic standard errors (SE) and MCSE for the multivariate SV model (12) fit with ADMB-RE to three exchange rate series: Pound/Dollar (P), Deutschmark/Dollar (D) and Yen/Dollar (Y).

		Σ			μ	Φ			Σ_η		
		P	D	Y		P	D	Y	P	D	Y
LA-ML											
Est.	P	1	−0.807	−0.675	−0.945	0.948	0	0	0.055	0	0
	D	−0.807	1	0.813	−0.876	0	0.938	0	0	0.038	0
	Y	−0.675	0.813	1	−1.314	0	0	0.938	0	0	0.072
SE	P	0	0.013	0.02	0.155	0.018	0	0	0.018	0	0
	D	0.013	0	0.012	0.112	0	0.019	0	0	0.011	0
	Y	0.02	0.012	0	0.147	0	0	0.02	0	0	0.021
CPU (s)	1421	Log-lik:	−1771.13								
SML ($S = 128$)											
Est.	P	1	−0.807	−0.675	−0.943	0.949	0	0	0.055	0	0
	D	−0.807	1	0.813	−0.874	0	0.94	0	0	0.038	0
	Y	−0.675	0.813	1	−1.312	0	0	0.942	0	0	0.067
SE	P	0	0.013	0.019	0.157	0.018	0	0	0.017	0	0
	D	0.013	0	0.012	0.115	0	0.019	0	0	0.01	0
	Y	0.019	0.012	0	0.152	0	0	0.019	0	0	0.019
MCSE	P	0	6e−04	8e−04	0.0039	0.0027	0	0	0.0037	0	0
	D	6e−04	0	7e−04	0.0016	0	0.002	0	0	0.0017	0
	Y	8e−04	7e−04	0	0.0018	0	0	0.0037	0	0	0.0055
CPU (s)	5510	Log-lik:	−1770.07								

where $\mathbf{X}_t = (X_{1,t}, X_{2,t}, X_{3,t})'$, $\mathbf{h}_t = (h_{1,t}, h_{2,t}, h_{3,t})'$, $\Sigma_\eta = \text{diag}\{\sigma^2_{\eta,1}, \sigma^2_{\eta,2}, \sigma^2_{\eta,3}\}$, $\mu = (\mu_1, \mu_2, \mu_3)'$, $\Phi = \text{diag}\{\phi_1, \phi_2, \phi_3\}$, and

$$\Sigma = \begin{pmatrix} 1 & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & 1 & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & 1 \end{pmatrix},$$

is a correlation matrix. Table 5 shows the results for our two ML methods. The two sets of parameter estimates are close to each other and appear reasonable. The estimated covariances in Σ are all statistically significantly different from zero, with the correlations between Pound and Deutschmark and between Pound and Yen being negative and the correlation between Deutschmark and Yen being positive. Indeed the importance of the correlations is also manifested in the log-likelihood value of the multivariate model which substantially improves upon the sum of the likelihood values obtained under the basic univariate SV model (not reported).

Note that the matrices Φ and Σ_η have been assumed to be diagonal, which means that the components of $\mathbf{h}_t$ are (a priori) independent. From a technical point of view, there would be no problem to allow the non-zero off-diagonal terms in Φ and Σ_η.

5.3. Fitting SV models to simulated data

To investigate the statistical efficiency of the proposed algorithms, we design two experiments. In the first experiment, we generate 500 datasets, each with $T = 500$ observations, from the basic SV model (1) where we set $(\sigma, \phi, \alpha) = (0.363, 0.9, -0.736)$ with $\alpha = 2(1-\phi)\log\sigma_X$. This parameter setting has been widely used in the literature to compare the performance of alternative estimation methods; see, for example Andersen and Sorensen (1996), Fridman and Harris (1998), Sandmann and Koopman (1998), Andersen et al. (1999) and Bates (2006). We adopt this parameter setting to avoid the need to compute these alternative estimates. Instead we only fit the basic SV model to each simulated sequence using LA-ML and SML (with $S = 128$). The use of 500 simulation replica allows the bias and the root mean square error (RMSE) of the estimators to be assessed. The simulation experiment is also conducted for the larger sample size $T = 2000$. Table 6 reports the results and appends to Andersen et al.'s Table 7 and Bates' Table 9.

Several conclusions may be drawn from Table 6. First, our LA-ML provides accurate estimates. In the case where the sample size is $T = 500$, it is clearly more efficient than QML. Also it appears more efficient than GMM and EMM in terms of σ. In the case where the sample size is $T = 2000$, it outperforms QML, GMM and EMM and performs nearly as well as AML. Second, our SML stands out as one of the most efficient estimation procedures in both cases. When the sample size is $T = 2000$, it provides the most efficient estimate for α and ϕ. We also experiment with larger values for S and the results remain qualitatively unchanged.

Next, we conduct a similar simulation experiment ($T = 2000$ and 500 simulation replica) where we include the leverage model and t-SV model in addition to the basic SV model. This experiment is limited to our methods (LA-ML and SML with $S = 128$). The parameter setting used is $\sigma_X = 1$, $\phi = 0.95$, $\sigma = 0.2$, $\rho = -0.6$, $\nu = 8$, which is close to the actual

Table 6
Comparison of different estimation methods with respect to bias and RMSE under the basic SV model (1) using simulated data. QML is the quasi ML method of Harvey et al. (1994). GMM is the generalized method of moments of Andersen and Sorensen (1996). EMM is the efficient method of moments of Gallant and Tauchen (1996) and implemented by Andersen et al. (1999). AML is the approximate ML method of Bates (2006). N-ML is the numerical ML of Fridman and Harris (1998). MCL is the Monte Carlo likelihood method of Sandmann and Koopman (1998). LA-ML is our Laplace approximation based ML method while SML is our simulated ML method.

	$T = 500$			$T = 2000$		
	α	ϕ	σ	α	ϕ	σ
True value	−0.736	0.90	0.363	−0.736	0.90	0.363
	Bias					
QML	−0.70	−0.09	0.09	−0.117	−0.02	0.20
GMM	0.12	0.02	−0.12	0.15	0.02	−0.08
EMM	−0.17	−0.02	0.02	−0.57	−0.07	−0.004
AML	−0.15	−0.02	0.02	−0.039	0.005	0.005
N-ML	−0.13	−0.02	0.01	NA	NA	NA
MCL	0.14	0.0	0.01	NA	NA	NA
LA-ML	−0.248	−0.033	0.025	−0.058	−0.008	0.0018
SML	−0.130	−0.020	0.012	0.024	0.003	−0.016
	RMSE					
QML	1.60	0.22	0.27	0.46	0.06	0.11
GMM	0.59	0.08	0.17	0.31	0.04	0.12
EMM	0.60	0.08	0.20	0.224	0.030	0.049
AML	0.42	0.06	0.08	0.173	0.023	0.043
N-ML	0.43	0.05	0.08	NA	NA	NA
MCL	0.27	0.04	0.08	NA	NA	NA
LA-ML	0.632	0.085	0.099	0.195	0.026	0.043
SML	0.439	0.057	0.081	0.144	0.019	0.038

Table 7
Bias and RMSE (parenthesis) of estimators under three different SV models evaluated using simulated data (500 simulation replica each of length $T = 2000$), for true parameter $\sigma_X = 1, \phi = 0.95, \sigma = 0.2$, $\rho = -0.6$, $\nu = 8$. CPU times are averages across the 500 simulation replica.

Parameter	Basic SV	Leverage SV	t-SV
	LA-ML		
σ_X	0.0038(0.0443)	0.0026(0.0378)	0.0055(0.0462)
σ	0.0102(0.0333)	0.0088(0.0257)	0.0116(0.0473)
ϕ	0.0070(0.0203)	0.0042(0.0133)	0.0070(0.0225)
ρ		0.0104(0.0717)	
ν			0.3889(2.0770)
CPU (s)	57.12	103.55	66.43
	SML ($S = 128$)		
σ_X	0.0024(0.0445)	0.0023(0.0378)	0.0009(0.0464)
σ	0.0165(0.0336)	0.0095(0.0255)	0.0337(0.0620)
ϕ	0.0073(0.0188)	0.0038(0.0136)	0.0136(0.0282)
ρ		0.0062(0.0713)	
ν			0.2056(2.0718)
CPU (s)	67.20	309.92	178.24

estimates obtained from daily financial returns. Table 7 reports the average bias and RMSE of each estimator, and the mean of the CPU time in seconds.

Several conclusions may be drawn from Table 7. First, both methods continue to provide accurate estimates in the new parameter setting. Second, both methods can estimate all the parameters in the SV model with the leverage effect and the t-SV model. Third, the computational cost in all cases is not high.

6. Conclusion

This paper explores the use of AD and the Laplace approximation as a basis for obtaining numerical and simulated ML estimation for univariate and multivariate SV models. Also based on the Laplace approximation, we develop simple algorithms for obtaining the filtered, smoothed and forecasted latent variable. We show how the software ADMB-RE facilitates the automatic implementation of these algorithms. The proposed techniques are flexible, robust, computationally efficient, and statistically efficient. Applications of the proposed techniques using both simulated and actual data highlight these advantages.

The present paper focuses on SV models, but the flexibility of the approach means that it could also be applied to the conditional intensity processes (Bauwens and Galli, 2009), multinomial-multi-period probit models (Liesenfeld and Richard, 2010) and dynamic factor models for multivariate count data (Jung et al., 2010). However, for the purpose of illustrating the various features of ADMB-RE the class of SV models is rich enough. Further, not all models contain latent variables, or in some cases it may be possible to do the integration analytically. AD is nevertheless useful for obtaining the gradient of the likelihood function (Bastani and Guerrieri, 2008).

While AD and the Laplace approximation facilitate the ML estimation of nonlinear and non-Gaussian state space models, they can be employed to perform a Bayesian analysis as well, in the spirit similar to Meyer et al. (2003). The idea is as follows. First, we can approximate the likelihood function $p(\mathbf{X}|\theta)$ by the Laplace approximation (3), making the augmentation of the parameter vector redundant. Second, we can use a tailored MH algorithm to obtain a MCMC sample from the posterior of θ. For a general survey of tailored MH approaches, see Chib (2001). To use a tailored MH algorithm we can first fit the model by maximizing the approximated marginal likelihood (3). Denote by $\hat{\theta}$ the resulting estimate of θ, and by $\hat{\Sigma}_{\hat{\theta}}$ its covariance matrix based on the observed Fisher information. The MH-proposal density is taken to be a multivariate normal, centered at the current parameter value, with covariance matrix $\hat{\Sigma}_{\hat{\theta}}$. Note further that for each value of θ proposed by the MH-algorithm the Laplace approximation is invoked via Eq. (3). This algorithm, which has been implemented in ADMB-RE (Fournier et al., 2011), differs from that of Meyer et al. (2003) who used a sequential Laplace approximation.

Many efficient MCMC algorithms have been proposed to estimate SV models in the econometrics literature (see e.g. Omori et al., 2007). Chib et al. (2009) provided an interesting survey of the literature. We do not necessarily expect Bayesian methods based on AD and the Laplace approximation to perform better than these efficient MCMC methods.

The programming effort required to build SV models in ADMB-RE and WinBUGS is comparable in the sense that both languages require only that the user provides a probabilistic description of the model. All algorithmic and computational details are hidden. We found (Table 4) that the time taken to obtain point estimates with associated standard errors is much lower with ADMB-RE than with WinBUGS. It is, however, important to point out that after having run WinBUGS the full posterior distribution is available. Bolker et al. (2013) provides a more detailed comparison of ADMB and BUGS in the field of ecology, although that paper did not look at state-space models in particular.

Judging convergence can be very difficult in MCMC samplers, while for methods that calculate the MLE, the gradient of the log likelihood function provides an explicit convergence measure. Since ADMB calculates the gradient by automatic differentiation, to machine precision, this convergence measure is an important part of our proposed method. Further, our method allows classical frequentist inference about the parameters (but not the volatilities) which is preferred over Bayesian analysis by some researchers.

Acknowledgments

We would like to thank an Associate Editor, two referees, David Fournier, Tore Kleppe, Vita Petersone and Magne Solheim. Skaug thanks his hosts at Singapore Management University for support and hospitality in a visit. Yu gratefully acknowledges the financial support from Singapore Ministry of Education Academic Research Fund Tier 2 under the grant number MOE2011-T2-2-096.

Appendix A. Supplementary material

The computer code used in the paper, both raw and compiled, can be found at http://www.mysmu.edu/faculty/yujun/research.html.

Appendix B. ADMB commands, file structure, sample code

ADMB exists for the WINDOWS, LINUX and MAC platforms as a set of C++ libraries and a preprocessor that converts the ADMB code (e.g. Fig. 1) into C++. The C++ compiler is not part of ADMB, and must be installed separately. A model implemented in ADMB consists of three separate files that share the base name (here taken to be basicSV), but with different extensions:

.tpl is known as the "template file", and specifies the likelihood function in terms of θ and $\mathbf{h}$ using a language that resembles C++. An example is given in Fig. 1.
.dat contains the data needed by the model, including dimensions of arrays.
.pin contains initial values of θ used in the numerical optimization of the integrated likelihood.

The template file basicSV.tpl is compiled by the command

```
admb -r basicSV
```

into an executable file basicSV.exe (under WINDOWS) using a C++ compiler, where -r is a command line option invoking the latent variable module. When executed to perform ML estimation, basicSV.exe produces several output files. We here only discuss basicSV.std which contains point estimates and standard errors for θ and $\mathbf{h}$ (smoothed values). In addition ADMB-RE has the ability to calculate the standard error of any user specified function $w(\theta, \mathbf{h})$, using the delta-method.

Fig. 1 shows the template file which implements the basic SV model (1). Taking the logarithm (and changing the sign) of the joint likelihood function $p(\mathbf{X}, \theta|\mathbf{h})p(\mathbf{h}|\theta)$ we obtain the objective function

$$-\left[\log\left\{p(h_1;\sigma,\phi)\right\}\right]-\left[\sum_{t=2}^{T}\log\left\{p(h_t|h_{t-1};\phi,\sigma)\right\}\right]-\left[\sum_{t=1}^{T}\log\left\{p(X_t|h_t;\sigma_X)\right\}\right]. \tag{B.1}$$

The three terms in square brackets are specified separately in the template file. It should be noted that the Hessian matrix Ω is banded under this model, i.e. $\Omega_{ij}=0$, whenever $|i-j|>1$. This special structure can be exploited by ADMB-RE both in the manipulation of the matrix Ω and in the derivative calculations involved in the evaluation of Ω itself. (In order for ADMB-RE to recognize this structure, some directives must be added to the code in Fig. 1.)

An ADMB program is compiled and run from a command line prompt, but an IDE (Integrated Developer Interface) exists (Fournier et al., 2011). To do the maximum likelihood estimation based on the Laplace approximation one gives the command:

```
basicSV -ilmn 5
```

where the option `-ilmn 5` invokes a limited-memory Newton method for the optimization problem (4). For high dimensional problems this optimization algorithm is more efficient than the quasi-Newton method used as the default in ADMB-RE. To perform the simulated maximum likelihood estimation, we use the command

```
basicSV -ilmn 5 -is 64
```

where `-is 64` specifies $S=64$ in the simulated maximum likelihood method.

Appendix C. ADMB code to estimate flexible SV models

To fit the t-SV model, the only required change is in the conditional distribution $X_t|h_t$. Hence, the only change we need to make in the code in Fig. 1 is to introduce a new parameter `nu` in the `PARAMETER_SECTION`, and to replace the last for-loop with

```
for (i=1;i<=n;i++)
{
  dvariable sigma_X = sigma_X*exp(0.5*h(i));
  dvariable t = X(i)/sigma_X;
  g -= gammln(0.5*(nu+1.0)) - gammln(0.5*nu) - 0.5*log(nu) - 0.5723649429;
  g -= -log(sigma_X) - (nu+1.0)/2*log(1+square(t)/nu);
}
```

Here, `gammln` is the log-gamma function.

To fit the leverage SV model, the first line in (11) is coded in ADMB-RE as:

```
g -=  -0.5*log_2pi - 0.5*log(1-square(rho))
      - 0.5*rho*(rho*square(X(i))*exp(h(i))/sigma_X)
      + rho*square((h(i+1)-phi*h(i))/sigma)
      - 2*X(i)*(h(i+1)-phi*h(i))*(1-square(rho))/(sigma*sigma_X*exp(0.5*h(i)))
      - 0.5*h(i) -log(sigma_X) - 0.5*square(X(i)/sigma_X)/exp(h(i));
```

References

Andersen, T.G., Bollerslev, T., Diebold, F.X., Ebens, H., 2001. The distribution of realized stock return volatility. Journal of Financial Economics 61, 43–76.

Andersen, T., Chung, H., Sorensen, B., 1999. Efficient method of moments estimation of a stochastic volatility model: a Monte Carlo study. Journal of Econometrics 91, 61–87.

Andersen, T., Sorensen, B., 1996. GMM estimation of a stochastic volatility model: a Monte Carlo study. Journal of Business and Economic Statistics 14, 329–352.

Bastani, H., Guerrieri, L., 2008. On the application of automatic differentiation to the likelihood function for dynamic general equilibrium models. In: Bischof, C.H., Bucker, H.M., Hovland, P., Naumann, U., Utke, J. (Eds.), Advances in Automatic Differentiation. In: Lecture Notes in Computational Science and Engineering, vol. 64. Springer, Berlin, Heidelberg, pp. 303–313.

Bates, D., 2006. Maximum likelihood estimation of latent affine processes. Review of Financial Studies 19, 909–965.

Bauwens, L., Galli, F., 2009. Efficient importance sampling for ML estimation of SCD models. Computational Statistics and Data Analysis 53, 1974–1992.

Bolker, B.M., Gardner, B., Maunder, M., Berg, C.W., Brooks, M., Comita, L., Crone, E., Cubaynes, S., Davies, T., Valpine, P., Ford, J., Gimenez, O., Kery, M., Kim, E.J., Lennert-Cody, C., Magnusson, A., Martell, S., Nash, J., Nielsen, A., Regetz, J., Skaug, H.J., Zipkin, E.F., 2013. Strategies for fitting nonlinear ecological models in R, AD Model Builder, and BUGS. Methods in Ecology and Evolution 4, 501–512.

Chib, S., 2001. Markov chain Monte Carlo methods: computation and inference. In: Heckman, J.J., Leamer, E. (Eds.), Handbook of Econometrics, vol. 5. North-Holland, Amsterdam, pp. 3569–3649.

Chib, S., Omori, Y., Asai, M., 2009. Multivariate stochastic volatility. In: Andersen, T.G., Davis, R.A., Kreiss, Jens-Peter, Mikosch., T. (Eds.), Handbook of Financial Time Series. Springer-Verlag, Berlin, pp. 365–400.

Durbin, J., Koopman, S.J., 1997. Monte Carlo maximum likelihood estimation for non-Gaussian state space models. Biometrika 84, 669–684.

Durham, G.S., 2006. Monte Carlo methods for estimating, smoothing, and filtering one and two-factor stochastic volatility models. Journal of Econometrics 133, 273–305.

Fournier, D.A., Skaug, H.J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M.N., Nielsen, A., Sibert, J., 2011. AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods & Software 27, 233–249.
Fridman, M., Harris, L., 1998. A maximum likelihood approach for non-Gaussian stochastic volatility models. Journal of Business and Economic Statistics 16, 284–291.
Gallant, A., Tauchen, G., 1996. Which moments to match? Econometric Theory 12, 657–681.
Griewank, A., Corliss, G. (Eds.), 1991. Differentiation of Algorithms. SIAM, Philadelphia.
Harvey, A.C., Koopman, S.J., 1992. Diagnostic checking of unobserved components time series models. Journal of Business and Economic Statistics 10, 377–389.
Harvey, A.C., Ruiz, E., Shephard, N., 1994. Multivariate stochastic variance models. Review of Economic Studies 61, 247–264.
Jung, R.C., Liesenfeld, R., Richard, J.F., 2010. Dynamic factor models for multivariate count data: an application to stock-market trading activity. Journal of Business and Economic Statistics 29 (1), 73–85.
Kass, R.E., Steffey, D., 1989. Approximate Bayesian inference in conditionally independent hierarchical models (parametric empirical Bayes models). Journal of the American Statistical Association 84, 717–726.
Kleppe, T., Skaug, H., 2012. Fitting general stochastic volatility models using laplace accelerated sequential importance sampling. Computational Statistics and Data Analysis 56 (11), 3105–3119.
Lee, K.M., Koopman, S.J., 2004. Estimating stochastic volatility models: a comparison of two importance samplers. Studies in Nonlinear Dynamics and Econometrics 8, 1–15.
Li, J., 2013. An unscented Kalman smoother for volatility extraction: evidence from stock prices and options. Computational Statistics and Data Analysis 58, 15–26.
Liesenfeld, R., Richard, J.F., 2003. Univariate and multivariate stochastic volatility models: estimation and diagnostics. Journal of Empirical Finance 10, 505–531.
Liesenfeld, R., Richard, J.F., 2006. Classical and Bayesian analysis of univariate and multivariate stochastic volatility models. Econometric Reviews 25, 335–360.
Liesenfeld, R., Richard, J.F., 2010. Efficient estimation of probit models with correlated errors. Journal of Econometrics 156 (2), 367–376.
Martino, S., Aas, K., Lindqvist, O., Neef, L.R., Rue, H., 2011. Estimating stochastic volatility models using integrated nested Laplace approximations. European Journal of Finance 17 (7), 487–503.
Meyer, R., Fournier, D.A., Berg, A., 2003. Stochastic volatility: Bayesian computation using automatic differentiation and the extended Kalman filter. Econometrics Journal 6, 408–420.
Meyer, R., Yu, J., 2000. BUGS for a Bayesian analysis of stochastic volatility models. Econometrics Journal 3, 198–215.
Omori, Y., Chib, S., Shephard, N., Nakajima, J., 2007. Stochastic volatility with leverage: fast likelihood inference. Journal of Econometrics 140, 425–449.
Sandmann, G., Koopman, S.J., 1998. Maximum likelihood estimation of stochastic volatility models. Journal of Econometrics 63, 289–306.
Shephard, N., Pitt, M.K., 1997. Likelihood analysis of non-Gaussian measurement time series. Biometrika 84, 653–667.
Spiegelhalter, D.J., Thomas, A., Best, N.G., Gilks, W.R., 2003. WinBUGS User Manual (Version 1.4). MRC Biostatistics Unit, Cambridge, UK.
Skaug, H.J., 2002. Automatic differentiation to facilitate maximum likelihood estimation in nonlinear random effects models. Journal of Computational and Graphical Statistics 11, 458–470.
Skaug, H.J., Fournier, D., 2006. Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models. Computational Statistics and Data Analysis 51, 699–709.
Yu, J., 2005. On leverage in a stochastic volatility model. Journal of Econometrics 127, 165–178.
Yu, J., Meyer, R., 2006. Multivariate stochastic volatility models: Bayesian estimation and model comparison. Econometric Reviews 25, 361–384.