EXACT FAST SMOOTHING IN SWITCHING MODELS WITH APPLICATION TO STOCHASTIC VOLATILITY

Ivan Gorynin, Stéphane Derrode, Emmanuel Monfrini, Wojciech Pieczynski

To cite this version:
Ivan Gorynin, Stéphane Derrode, Emmanuel Monfrini, Wojciech Pieczynski. EXACT FAST SMOOTHING IN SWITCHING MODELS WITH APPLICATION TO STOCHASTIC VOLATILITY. EUSIPCO, Aug 2015, Nice, France. hal-01165338

HAL Id: hal-01165338
https://hal.science/hal-01165338
Submitted on 19 Jun 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
EXACT FAST SMOOTHING IN SWITCHING MODELS WITH APPLICATION TO STOCHASTIC VOLATILITY

Ivan Gorynin*, Stéphane Derrode†, Emmanuel Monfrini* and Wojciech Pieczynski*

* Telecom SudParis, SAMOVAR, CNRS UMR 5157, Évy, France.
† École Centrale de Lyon, LIRIS, CNRS UMR 5205, Écully, France.

ABSTRACT

We consider the problem of statistical smoothing in nonlinear non-Gaussian systems. Our novel method relies on a Markov-switching model to operate recursively on series of noisy input data to produce an estimate of the underlying system state. We show through a set of experiments that our technique is efficient within the framework of the stochastic volatility model.

Index Terms— Nonlinear systems, Stochastic volatility, Bayesian smoother, Conditionally Gaussian linear state-space model, Smoothing in switching systems.

1. INTRODUCTION

Let us consider two random sequences $X_1^N = (X_1, \ldots, X_N)$ and $Y_1^N = (Y_1, \ldots, Y_N)$. $X_n$ and $Y_n$ respectively take their values in $\mathbb{R}^d$ and $\mathbb{R}^p$. $X_1^N$ is hidden, while $Y_1^N$ is observed. We explore the problem of “Bayesian smoothing” which aims to recover, for $n = 1, \ldots, N$, $X_n$ from $Y_1^N$.

With the usual notations for conditional expectation, one gets

\[ \mathbb{E}[X_n | Y_1^N] = \mathbb{E}[X_n | R_n^N, Y_n] \] (1)

and the following recursion equation:

\[ X_{n+1} = \Phi(X_n, U_{n+1}); \]

\[ Y_n = \Psi(X_n, V_n); \] (2)

where $U_1, V_1, \ldots, U_N, V_N$ are convenient independent variables, and functions $\Phi$ and $\Psi$ specify the state equation and the measurement equation. There is no known Bayesian smoothing algorithm for this general framework, and one uses various approximations [5, 6]. Among them, the Monte Carlo methods are very attractive, but they can meet difficulties in some situations. Our algorithm makes use of the smoothing in the “Conditionally Markov switching hidden linear model” (CMSHLM) [7], which can accurately fit the given HMM. In this way, we state a new general method for smoothing and show its interest within the framework of the classic stochastic volatility model [8–10].

The paper is organized as follows. In the next section, we remind the CMSHLM model and show its relevance for solving the problem. In Section 3, we describe the “Stationary conditionally Gaussian observed Markov switching model” (SCGOMSM) which is a particular Gaussian CMSHLM. Our smoothing method, which is based on the Expectation-Maximization (EM) algorithm and Gaussian CMSHLM, is given in Section 4. The fifth section presents experimental results, and the sixth one draws conclusions and perspectives.

2. EXACT SMOOTHING IN CONDITIONALLY MARKOV SWITCHING HIDDEN LINEAR MODELS

Let $R_1^N = (R_1, \ldots, R_N)$ be a random sequence taking its values in $\Omega = \{1, \ldots, K\}$.

Definition 1 We say that a discrete time Markov process $(X_1^N, R_1^N, Y_1^N)$ is a “Conditionally Markov switching hidden linear model” (CMSHLM) if it verifies

\[ p\left(r_{n+1}, y_{n+1} | x_n, r_n, y_n \right) = p\left(r_{n+1} | y_{n+1} | r_n, y_n \right), \] (3)

and the following recursion equation:

\[ X_{n+1} = F_{n+1}(R_{n+1}, Y_{n+1})X_n + G_{n+1}(R_{n+1}, Y_{n+1})W_{n+1} + H_{n+1}(R_{n+1}, Y_{n+1})Y_{n+1}, \] (4)

with suitable matrices $F_{n+1}(R_{n+1}, Y_{n+1}), G_{n+1}(R_{n+1}, Y_{n+1}), H_{n+1}(R_{n+1}, Y_{n+1})$ and Gaussian unit-variance white noise vector $W_{n+1}$.

We can state the following:

Proposition 1

Let $(X_1^N, R_1^N, Y_1^N)$ be a CMSHLM. Next, for each $n$ in $\{1, \ldots, N\}$, $p\left(r_n | y_N \right)$, $\mathbb{E}[X_n | y_N]$ and $\mathbb{E}[X_n X_n^T | y_N]$ can be computed with a complexity linear in $N$. 

Proof} Firstly, let us prove that
\[
E \left[ X_n \mid y_1^N \right] = \sum_{r_n} p(r_n \mid y_1^N) E \left[ X_n \mid r_n, y_1^N \right],
\]
(5)
\[
E \left[ X_n X_n^\top \mid y_1^N \right] = \sum_{r_n} p(r_n \mid y_1^N) E \left[ X_n X_n^\top \mid r_n, y_1^N \right].
\]
(6)

According to (3), \( p(x_n \mid r_n, y_1^{n+1}) = p(x_n \mid r_n, y_1^n) \) and more generally, \( p(x_n \mid r_n, y_1^N) = p(x_n \mid r_n, y_n) \). Then (5) comes from \( p(x_n \mid y_1^N) = \sum_{r_n} p(r_n \mid y_1^N) p(x_n \mid r_n, y_1^N) \), and we prove (6) in a similar way.

Secondly, the posterior marginals \( p(r_n \mid y_1^N) \) are calculable using the forward-backward algorithm. The “forward” and “backward” probabilities \( \alpha_n(r_n) = p(r_n, y_1^n) \), \( \beta_n(r_n) = p(y_1^N \mid r_n, y_n) \) are recursively computable with
\[
\alpha_1(r_1) = p(r_1, y_1);
\]
\[
\alpha_{n+1}(r_{n+1}) = \sum_{r_n \in \Omega} \alpha_n(r_n)p(r_{n+1} \mid r_n, y_n); \quad \text{(7)}
\]
\[
\beta_N(r_N) = 1;
\]
\[
\beta_n(r_n) = \sum_{r_{n+1} \in \Omega} \beta_{n+1}(r_{n+1})p(r_{n+1} \mid r_n, y_n). \quad \text{(8)}
\]

This algorithm enables us to calculate the jump smoothed values \( p(r_n \mid y_1^N) \) and the jump filtered values \( p(r_n \mid y_1^n) \) using
\[
p(r_n \mid y_1^n) = \frac{\alpha_n(r_n)\beta_n(r_n)}{\sum_{r_n \in \Omega} \alpha_n(r_n)\beta_n(r_n)}, \quad \text{(9)}
\]
\[
p(r_n \mid y_1^N) = \frac{\alpha_n(r_n)}{\sum_{r_n \in \Omega} \alpha_n(r_n)}. \quad \text{(10)}
\]

We can also get \( E \left[ X_n \mid r_n, y_1^n \right] \) and \( E \left[ X_n X_n^\top \mid r_n, y_1^n \right] \) inductively [7]. Indeed, the recursive formula to compute \( E \left[ X_{n+1} \mid r_{n+1}, y_1^{n+1} \right] \) from \( E \left[ X_n \mid r_n, y_1^n \right] \) is
\[
E \left[ X_{n+1} \mid r_{n+1}, y_1^{n+1} \right] = \sum_{r_n} p \left( r_n \mid r_{n+1}, y_1^{n+1} \right)
\]
\[
\left( F_{n+1}(r_{n+1}, y_1^{n+1}) E \left[ X_n \mid r_n, y_1^n \right] + H_{n+1}(r_{n+1}, y_1^{n+1}) \right). \quad \text{(11)}
\]

The recursion to compute \( E \left[ X_n X_n^\top \mid r_n, y_1^{n+1} \right] \) is similar to the previous one:
\[
E \left[ X_{n+1} X_{n+1}^\top \mid r_{n+1}, y_1^{n+1} \right] = \sum_{r_n} p \left( r_n \mid r_{n+1}, y_1^{n+1} \right)
\]
\[
\left( F_{n+1}(r_{n+1}, y_1^{n+1}) E \left[ X_n X_n^\top \mid r_n, y_1^n \right] F_{n+1}(r_{n+1}, y_1^{n+1})^\top + H_{n+1}(r_{n+1}, y_1^{n+1}) E \left[ X_n \mid r_n, y_1^n \right] H_{n+1}(r_{n+1}, y_1^{n+1})^\top + G_{n+1}(r_{n+1}, y_1^{n+1}) G_{n+1}(r_{n+1}, y_1^{n+1})^\top + H_{n+1}(r_{n+1}, y_1^{n+1}) H_{n+1}(r_{n+1}, y_1^{n+1})^\top \right). \quad \text{(12)}
\]

Both expressions require \( p \left( r_n \mid r_{n+1}, y_1^{n+1} \right) \), which is obtainable using (10) and
\[
p \left( r_n \mid r_{n+1}, y_1^{n+1} \right) = \sum_{r_n} p \left( r_{n+1}, y_1^{n+1} \mid r_n, y_n \right) p \left( r_n \mid y_1^n \right).
\]
\[
p \left( r_n \mid r_{n+1}, y_1^{n+1} \right) = \sum_{r_n} p \left( r_{n+1}, y_1^{n+1} \mid r_n, y_n \right) p \left( r_n \mid y_1^n \right).
\]
\[
\text{and the covariance matrix}
\]
\[
\mathbf{E}(r_1^{n+1}) = \left[ \begin{array}{c} \mathbf{M}(r_{n+1}) \\ \mathbf{M}(r_1^{n+1}) \end{array} \right] = \left[ \begin{array}{c} E \left[ Z_n \mid r_1 \right] \\ E \left[ Z_{n+1} \mid r_1 \right] \end{array} \right] \quad \text{(14)}
\]
\[
\text{and the multivariate normal distribution:}
\]
\[
p \left( x_1^{n+1} \mid y_1^{n+1} \right) = \mathcal{N} \left( \mathbf{x}_{1, n+1}^{n+1}, \mathbf{y}_{1, n+1}^{n+1} \right) \quad \text{(15)}
\]
\[
\text{Definition 2} We say that a discrete time stationary Markov process \( \left( X_1^{n+1}, R_1^{n+1}, Y_1^{n+1} \right) \) is a “Stationary conditionally Gaussian observed Markov switching model” (SGCOMSM) if it verifies (14)-(16) and has the following property:
\[
p \left( y_1^{n+1} \mid x_1^{n+1}, r_1^{n+1} \right) = p \left( y_1^{n+1} \mid r_1^{n+1} \right). \quad \text{(17)}
\]

Let us remember that SGCOMSMs can be very close to the classic “conditionally Gaussian linear state-space model” (CGLSSM) [11, 12], which does not offer the possibility of fast smoothing [5].

**Proposition 2** If a discrete time stationary Markov process \( \left( X_1^{n+1}, R_1^{n+1}, Y_1^{n+1} \right) \) is a SGCOMSM, it is also a CMSLHM with \( F_{n+1}, G_{n+1} \) and \( H_{n+1} \) in (4) given by (26)-(28).

**Proof** According to (14)-(16), we have \( p \left( r_{n+1} \mid x_n, r_n, y_n \right) = p \left( r_{n+1} \mid r_n \right) \). Then we use (17) to prove that a SGCOMSM has property (3) of the CMSLHM.

To find out the corresponding \( F_{n+1}, G_{n+1}, H_{n+1} \) in (4), we set
\[
\mathbf{A}(r_1^{n+1}) = \mathbf{S}(r_1^{n+1}) \mathbf{S}(r_1^{n+1})^- \left( \begin{array}{c} p(r_1^{n+1}) \\ \mathbf{B}(r_1^{n+1}) \end{array} \right) \quad \text{(18)}
\]
and consider \( \mathbf{B}(r_1^{n+1}) \) and \( \mathbf{Q}(r_1^{n+1}) \) such that
\[
\mathbf{B}(r_1^{n+1}) \mathbf{B}(r_1^{n+1})^\top \mathbf{S}(r_1^{n+1}) = \mathbf{S}(r_1^{n+1}) \mathbf{S}(r_1^{n+1})^- \mathbf{S}(r_1^{n+1}) \mathbf{S}(r_1^{n+1}) \quad \text{(19)}
\]
\[ Q(r_{n+1}^+) = \begin{bmatrix} Q_1(r_{n+1}^+) \\ Q_2(r_{n+1}^+) \\ Q_4(r_{n+1}^+) \end{bmatrix} = B(r_{n+1}^+)B^T(r_{n+1}^+) \]  

Equation (17) induces that the matrix \( A(r_{n+1}^+) \) has the following form:
\[ A(r_{n+1}^+) = \begin{bmatrix} A_1(r_{n+1}^+) & A_2(r_{n+1}^+) \\ 0 & A_4(r_{n+1}^+) \end{bmatrix}. \]  

Hence, we can state that the discrete time process \((Z_1^n)^T\) satisfies the following recursion equation:
\[ Z_{n+1} = A(R_{n+1}^+)(Z_n - M(R_n)) + B(R_{n+1}^+)W_{n+1} + M(R_{n+1}), \]  

where \( W_1, \ldots, W_N \) are Gaussian unit-variance white noise vectors. Let us set \( M(r_n) = \begin{bmatrix} M_1(r_n) \end{bmatrix}^T, M_2(r_n) \) \( \sim N(1, \ldots, 1)^T \)
\[ p(x_{n+1}, y_{n+1} \mid x_n, r_{n+1}^+ \), \( y_n \) is a multivariate normal probability density function. Its covariance matrix is \( Q(r_{n+1}^+) \)
and its mean is
\[ A(r_{n+1}^-) = \begin{bmatrix} N_1(r_{n+1}^+) \\ N_2(r_{n+1}^+) \end{bmatrix} = \begin{bmatrix} A_1(r_{n+1}^-)A_2(r_{n+1}^+)N_1(r_{n+1}^+) \\ A_4(r_{n+1}^-)N_2(r_{n+1}^+) \end{bmatrix}, \]  

where we set
\[ N_1(r_{n+1}^+) = M_1(r_{n+1}) - A_1(r_{n+1}^+)M_1(r_n) \\ N_2(r_{n+1}^+) = M_2(r_{n+1}) - A_4(r_{n+1}^+)M_2(r_n). \]  

\[ p(x_{n+1}, y_{n+1} \mid x_n, r_{n+1}^+, y_{n+1}^+) \] is also a multivariate normal probability density function with mean vector is
\[ Q_2(r_{n+1}^-)Q_4^{-1}(r_{n+1}^-)(y_{n+1} - A_4(r_{n+1}^-)y_n - N_2(r_{n+1}^-)) + A_1(r_{n+1}^-)x_n + A_2(r_{n+1}^-)y_n = N_1(r_{n+1}^-), \]  

and covariance matrix
\[ Q_1(r_{n+1}^-) - Q_2(r_{n+1}^-)Q_4^{-1}(r_{n+1}^-)Q_3(r_{n+1}^-). \]  

This allows to complete the proof and to specify \( F_{n+1}, G_{n+1} \) and \( H_{n+1} \):
\[ F_{n+1}(r_{n+1}^+, y_{n+1}^+) = A_1(r_{n+1}^+), \]  
\[ H_{n+1}(r_{n+1}^+, y_{n+1}^+) = A_2(r_{n+1}^+)y_n + N_1(r_{n+1}^+) + \]  
\[ Q_2(r_{n+1}^-)Q_4^{-1}(r_{n+1}^-)(y_{n+1} - A_4(r_{n+1}^-)y_n - N_2(r_{n+1}^-), \]  
\[ G_{n+1}(r_{n+1}^+, y_{n+1}^+) = Q_1(r_{n+1}^-). \]  

To sum up, the smoothing procedure in the SCGOMSM at first recursively computes \( p(y_{n} \mid y_n^+) \), \( E[X_n \mid r_n, y_n^+] \), \( E[X_nX_n^T \mid r_n, y_n^+] \) using Algorithm 1 below, then computes the state smoothed values using (8) and (9) and finally produces the smoothed output using (5) and (6).

Algorithm 1: Given \( E[X_n \mid r_n, y_n^+] \), \( E[X_nX_n^T \mid r_n, y_n^+] \), \( p(r_n \mid y_n^+) \), and \( y_{n+1} \):
- Calculate \( F_{n+1}(r_{n+1}^+, y_{n+1}^+), H_{n+1}(r_{n+1}^+, y_{n+1}^+) \) and \( G_{n+1}(r_{n+1}^+, y_{n+1}^+) \) with (26)-(28);
- Calculate \( p(r_{n+1}^+, y_{n+1}^+ \mid r_n, y_n) = p(r_{n+1}^+ \mid r_n)p(y_{n+1}^+ \mid y_n) \). Let us remind that the probability density function \( p(y_{n+1}^+ \mid y_n) \) is multivariate normal with mean vector \( A_2(r_{n+1}^+)y_n + N_2(r_{n+1}^+) \) and covariance matrix \( Q_4(r_{n+1}^+) \);
- Use (7) and (10) to calculate \( p(r_{n+1}^+ \mid y_{n+1}^+), \) then use (13) to calculate \( p(r_{n+1} \mid y_{n+1}^+, y_{n+1}^+) \);
- Use (11) and (12) to calculate \( E[X_{n+1} \mid r_{n+1}^+, y_{n+1}^+] \) and \( E[X_{n+1}X_{n+1}^T \mid r_{n+1}^+, y_{n+1}^+] \).

4. APPROXIMATING NON-LINEAR NON-GAUSSIAN MODELS

Let us consider a stationary HMM that we described in the introduction. The probability density functions \( p(x_{n+1}^+ \mid y_n) \) and \( p(y_{n+1} \mid x_n) \) do not depend on \( n \), so \( p(x_{1}^+, y_{1}^+) \) defines the entire distribution of \( x_{1}^+, y_{1}^+ \). The idea is to approach it by a Gaussian mixture of \( K^2 \) components in such a way that this mixture derives from a SCGOMSM [13].

More precisely, it is possible to find an approximation
\[ p(x_{1}^+, y_{1}^+ \mid) \approx \sum_{1 \leq i, j \leq K} c_{ij} p_{ij}(x_{1}^+, y_{1}^+), \]
with an arbitrary precision and under mild conditions. In order to find a SCGOMSM close to the given HMM, we assume that \( c_{ij} \) is the distribution of the random variable \( (R_1, R_2) \) taking its values in \( \Omega^2 = \{1, \ldots, K\}^2 \), which means that we set \( c_{ij} = P(R_1 = i, R_2 = j) \). With respect to (16), we can then parametrize the conditional distribution \( p_{ij}(x_{1}^+, y_{1}^+) = p(x_{1}^+, y_{1}^+ \mid r_1 = i, r_2 = j) \) by the mean vector \( y_0 \) and the covariance matrix \( \Sigma_{ij} \). The suitable values for \( c_{ij}, y_0 \) and \( \Sigma_{ij} \) can be found with a training sample \((x_{1}^+, y_{1}^+) \) simulated within the HMM framework, and the Expectation-Maximization algorithm (EM) that we describe below.

Algorithm 2 inputs the simulated sample to produce the sequence \((c_{ij}^{(q)}, y_0^{(q)}, \Sigma_{ij}^{(q)})\), \( q \in \mathbb{N}, 1 \leq i, j \leq K \), assumed to tend to \( (c_{ij}, y_0, \Sigma_{ij}) \) as \( q \rightarrow \infty \).

Algorithm 2
1. Let \( q \leftarrow 0 \). Use the k-means clustering algorithm to assign each couple \((x_{1}^+, y_{1}^+) \) to one of the \( K \) clusters and consider, for each \( 1 \leq i \leq K \) and \( 1 \leq m \leq M, \phi_{0}^{(q)}(i) \) that equals 1 if \((x_{1}^+, y_{1}^+) \) belongs to the i-th cluster and 0 otherwise. Furthermore, consider, for each \( 1 \leq i, j \leq K \) and \( 1 \leq m < M, \psi_{0}^{(q)}(i, j) = \phi_{0}^{(q)}(i)\phi_{0}^{(q)}(j); \)
2. Use the formulas below to calculate, for each \( 1 \leq i, j \leq K \) and \( 1 \leq m < M, \psi_{1}^{(q)}(i, j) = \phi_{1}^{(q)}(i)\phi_{1}^{(q)}(j); \)
3. Use the formulas below to calculate, for each \( 1 \leq i, j \leq K \) and \( 1 \leq m < M, \psi_{2}^{(q)}(i, j) = \phi_{2}^{(q)}(i)\phi_{2}^{(q)}(j); \)
4. Use the formulas below to calculate, for each \( 1 \leq i, j \leq K \) and \( 1 \leq m < M, \psi_{3}^{(q)}(i, j) = \phi_{3}^{(q)}(i)\phi_{3}^{(q)}(j); \)
Finally, our new smoothing method is:

\[ c_{ij}^{(q+1)} = \frac{1}{M-1} \sum_{m=1}^{M-1} \psi_m(q)(i,j), \]  
(30)

\[ \psi_m^{(q+1)}(i,j) = \frac{\sum_{m=1}^{M-1} z_m^{r+1} \psi_m^{(q)}(i,j)}{\sum_{m=1}^{M-1} \psi_m^{(q)}(i,j)}, \]  
(31)

\[ \Xi_{ij}^{(q+1)} = \frac{\sum_{m=1}^{M-1} (z_m^r - \psi_m^{(q+1)}) (z_m^r - \psi_m^{(q)})^T}{\sum_{m=1}^{M-1} \psi_m^{(q)}(i,j)}, \]  
(32)

where \( z_m^{r+1} = (x^r_m, y^r_m, x^{r+1}_m, y^{r+1}_m)^T \).

3. To compute \( \psi_m^{(q+1)}(i,j) \), use (33) to compute the “forward” probabilities and use (34) to compute the “backward” ones, then use (35) to calculate \( \psi_m^{(q+1)}(i,j) \) for each \( 1 \leq i, j \leq K \) and \( 1 \leq m < M \).

\[ \alpha_1(i) = \alpha(i, \psi_m^{(q+1)}(i,j) | r_1 = i); \]  
\[ \alpha_m+1(j) = \sum_{i \in \Omega} \alpha_m(i) \psi_m^{(q+1)}(i,j) p_{ji}; \]  
(33)

\[ \beta_M(j) = 1; \]  
\[ \beta_m(i) = \sum_{j \in \Omega} \beta_{m+1}(j) \psi_m^{(q+1)}(i,j) p_{ij}; \]  
(34)

\[ \psi_m^{(q+1)}(i,j) = \frac{\alpha_m(i) \beta_{m+1}(j)}{\sum_{i^*, j^* \in \Omega} \alpha_{i^*, j^*}^{(q+1)}(i,j) p_{ji} \beta_{m+1}(j)}, \]  
(35)

where \( a_i = \sum_{j \in \Omega} \psi_m^{(q+1)}(i,j) p_{ji} = \frac{\psi_m^{(q+1)}(i,j)}{a_i} \) and \( \psi_m^{(q+1)}(i,j) = p(x_m^r, y_m^r | r_m = i, r_{m+1} = j; x_m^r, y_m^r) \). The conditional probability density functions \( p(x_m^r, y_m^r | r_1 = i) \) and \( p(x_m^r, y_m^r | r_m = i, r_{m+1} = j; x_m^r, y_m^r) \) are multivariate normal distribution defined by \( \psi_m^{(q+1)}(i,j) \) and \( \Xi_{ij}^{(q+1)} \) with respect to (14)-(23).

4. Stop according to a stopping criterion or increment \( q \) and repeat from Step 2.

Finally, our new smoothing method is:

**Algorithm 3** Given input data \( y_1^N \),
1. Consider a training sample \( (x_1^M, y_1^M) \) simulated within the HMM framework;
2. Apply Algorithm 2 to the training sample to compute \( (\epsilon_{ij}, \Psi_{ij}, \Xi_{ij})_{1 \leq i, j \leq K} \);
3. Compute recursively \( p(r_n | y_1^n) \), \( E[X_n | r_n, y_1^n] \), \( E[X_n, X_{n+1}^T | r_n, y_1^n] \) using Algorithm 1;
4. Calculate state smoothed values using (8) and (9);
5. Calculate the smoothed output using (5) and (6);

### 5. Experiments

Let us consider the stochastic volatility model defined by the following recursion equations:

\[ X_1 = U_1; \]  
(36)

\[ X_{n+1} = \mu + \phi(X_n - \mu) + \sigma U_{n+1}; \]  
(37)

\[ \gamma_n = \beta \exp(X_n/2) V_n; \]  
(38)

with fixed \( \mu \) and \( \sigma \), and Gaussian unit-variance white noise vectors \( U_1, V_1, \ldots, U_N, V_N \). In the non-degenerate case, this model is stationary if and only if \( \phi < 1 \) and \( \frac{\sigma^2}{2 \phi^2} = 1 \) [14]. Therefore, we consider four sets of parameters, corresponding to \( \mu = 0.5 \), \( \beta = 0.5 \), different values of \( \sigma^2 \) and \( \phi \) with respect to the condition \( \sigma^2 = 1 - \phi^2 \).

For each experiment instance, we aim to compare the behaviour of our method for \( K = 2, 3, 5 \) or 7. In concrete terms, we simulate a training sample \( (x_1^M, y_1^M) \) and a test sample \( (x_1^N, y_1^N) \) within the framework of the stochastic volatility model. Next, we use Algorithm 3 to compute a smoothed output, and we use \( x_N \) to calculate the mean square error (MSE). For our study cases, the training sample size is \( M = 20000 \), the number of iterations of EM is \( Q_{EM} = 100 \) and the test sample size is \( N = 1000 \).

We also use a particle smoother to compute \( E[X_n | Y_1^{n+T}] \) with \( T = 5 \), and we measure the MSE. We found out that using greater values of \( T \) needs more particles to cope with the degeneracy phenomenon, but does not change the MSE value. We thus consider that \( E[X_n | Y_1^{n+T}] \) is a good approximation of \( E[X_n | Y_1^{n+T}] \).

The results of our experiments are presented in Table 1. In the case of \( K = 5 \), the results are similar to those obtained with a particle smoother (PS). When choosing 100 particles, PS is quicker than our method because of the EM algorithm. However, when the underlying SCGOMSM fits the HMM, our method is as fast as the Kalman filter, and faster than PS. For the higher values of \( K \), the results remain stable and analogous to the PS ones, which probably means that they are close to the theoretical ones.

### 6. Conclusion

We put forward a new method to find the hidden signal in the framework of a nonlinear and non-Gaussian model. This method is general and works under slight conditions: we only need to be able to sample data according to the given HMM. When our model fits the nonlinear and non-Gaussian system,
Table 1. MSE of 100 separate experiments in the cases described above, for our method and for the particle smoother (1500 particles).

<table>
<thead>
<tr>
<th>Nb of mixture components</th>
<th>φ</th>
<th>σ²</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.02</td>
<td>0.39</td>
<td>0.23</td>
<td>0.14</td>
<td>0.13</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>0.19</td>
<td>0.48</td>
<td>0.39</td>
<td>0.35</td>
<td>0.34</td>
<td>0.33</td>
</tr>
<tr>
<td>3</td>
<td>0.80</td>
<td>0.36</td>
<td>0.56</td>
<td>0.50</td>
<td>0.47</td>
<td>0.47</td>
<td>0.46</td>
</tr>
<tr>
<td>4</td>
<td>0.50</td>
<td>0.75</td>
<td>0.70</td>
<td>0.67</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
</tr>
</tbody>
</table>

the method is as fast as the classic Kalman filter in linear systems.

We tested our approach in the framework of the stochastic volatility model, and it turns out that the mean square error obtained is very close to the theoretical one, the latter estimated by a particle smoother.

As a conclusion, let us mention two perspectives. The first one is to consider different and more complex stochastic volatility models [3, 15–17]; the second one is to consider more advanced families of switching models allowing fast exact smoothing.

REFERENCES


