Dipartimento di Statistica
"Giuseppe Parenti"

W O R K I N G P A P E R 2012/11

DMM - A Fortran program for dynamic mixture models

G. Fiorentini, C. Planas, A. Rossi

Università degli Studi di Firenze

Dipartimento di Statistica "G. Parenti" – Viale Morgagni 59 – 50134 Firenze - www.ds.unifi.it

DMM – A Fortran program for dynamic mixture models *

G.Fiorentini[(a)], C.Planas[(b)] and A.Rossi[(b)]

November 2012

Abstract

DMM is a Fortran program for Bayesian analysis of dynamic mixture models which produces posterior samples of the unobserved state vector, of the discrete latent variables, and of model parameters together with the marginal likelihood of the data set. Besides computational efficiency, **DMM** has several attractive features: the endogenous series can be univariate or multivariate, stationary or non-stationary, with some possibly missing observations, and they may be linked to some exogenous variables. We describe the methodology implemented and we show how to use the program with some examples.

KEYWORDS: Bayesian analysis, conditionally Gaussian state space model, Markov processes, Markov Chain Monte Carlo, software, time series, unobserved components.

*The ideas expressed here are those of the authors and do not necessarily reflect the position of the European Commission. We are grateful to Gianluca Caporello for useful advices.
(*a*) University of Florence, Department of Statistics, Informatics and Applications, and Rimini Centre for Economic Analysis.
(*b*) Joint Research Centre of European Commission.
e-mails: *fiorentini@ds.unifi.it*, *christophe.planas@jrc.ec.europa.eu*, *alessandro.rossi@jrc.ec.europa.eu*.

1 Introduction

DMM is a Fortran program for Bayesian analysis of dynamic mixture models. This framework includes linear specifications such as the structural time series models (Harvey, 1989), the dynamic linear models (West and Harrison, 1997), as well as non-linear specifications with outliers, structural changes, and parameters changes of random magnitude and timing (see for instance Giordani, Kohn, and van Dijk, 2007). Such a generality makes dynamic mixture models well suited to the analysis of a wide range of problems, and a large number of applications can be found in a variety of disciplines (see Kim and Nelson, 1999, Scott, 2002, and Fruhwirth-Schnatter, 2006).

Dynamic mixture models typically include a continuous unobserved state vector, some discrete latent variables that control discontinuities or change-points, plus the model parameters. **DMM** delivers posterior samples of each of these quantities using a Gibbs strategy. Up-to-date techniques are used to limit the computational burden to $O(\mathtt{T})$ operations, $\mathtt{T}$ denoting the sample size. In short, the state vector is sampled using the simulation smoother developed by Durbin and Koopman (2002), the discrete latent variables are drawn using a block extension of the Gerlach, Carter, and Kohn (2000) sampler proposed by Fiorentini, Planas and Rossi (2012), and the parameters are sampled using the slice sampler devised by Neal (2003). The program also produces predictive densities as well as the data marginal likelihood which can be used for model discrimination and model averaging. Besides computational efficiency, **DMM** has several attractive features: the endogenous series can be univariate or multivariate, stationary or non-stationary, they can be linked to some exogenous variables, and some observations may be missing. The program can also generate observations from the model specified by the user. **DMM** can be freely downloaded at *eemc.jrc.ec.europa.eu/Software-DMM.htm*.

To run the program, two inputs files are needed: the first contains the general information about the model, the prior distributions, and the data in human-readable format; the second is a dynamic link library (DLL) which returns the matrices of the state space representation given model parameters. This architecture gives a great flexibility: any parameterization can be chosen and any specification within the class of conditionally Gaussian models can be analyzed. In addition, the Fortran low-level language guarantees computational speed, which is a very important feature when resorting to Markov Chain Monte Carlo (MCMC) techniques. We give instructions to easily create such a DLL from a Fortran source code using the free GNU Fortran compiler for Microsoft Windows.

The program description is organized as follows. Section 2 presents the general model. Section 3 reviews the prior assumptions and the MCMC methodology. Section 4 details the input files, explains how to build the DLL, and describes the program output. Section 5 illustrates the program capabilities using the Nile riverflow and the US business cycle examples. Some remarks

are collected in Section 6. Section 7 concludes.

2 Model and assumptions

2.1 State space framework

DMM handles models that admit the general state space representation:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{c}_t\mathbf{z}_t + \mathbf{H}_t\mathbf{x}_t + \mathbf{G}_t\mathbf{u}_t \\ \mathbf{x}_t &= \mathbf{a}_t + \mathbf{F}_t\mathbf{x}_{t-1} + \mathbf{R}_t\mathbf{u}_t \end{aligned} \tag{2.1}$$

where $t = 1, \cdots, \mathtt{T}$, $\mathbf{y}_t$ is the $\mathtt{ny} \times 1$ vector of endogenous variables, $\mathbf{z}_t$ is the $\mathtt{nz} \times 1$ vector of exogenous series, $\mathbf{x}_t$ is the $\mathtt{nx} \times 1$ state vector, and $\mathbf{u}_t$ is the $\mathtt{nu} \times 1$ vector of shocks. The possibly time-varying vectors and matrices $\mathbf{c}_t$, $\mathbf{H}_t$, $\mathbf{G}_t$, $\mathbf{a}_t$, $\mathbf{F}_t$, and $\mathbf{R}_t$ are determined by the model parameters $\boldsymbol{\theta}$ and by a vector of discrete latent variables $\mathbf{S}_t = (S_{1t}, \cdots, S_{\ell t}, \cdots)$ with associated transition probabilities $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_\ell, \cdots)$. The following assumptions must be satisfied:

A1. Shocks are Gaussian: $\mathbf{u}_t \overset{\text{iid}}{\sim} N(\mathbf{0}, \mathbf{I})$.

A2. Each variable $S_{\ell t}$ takes $\mathtt{ns}_\ell$ values following independent Markov processes with transition probabilities $\pi_{\ell ij} \equiv \Pr(S_{\ell t} = i | S_{\ell t-1} = j)$, $i, j = 1, \cdots, \mathtt{ns}_\ell$, that are collected into the vector $\boldsymbol{\pi}_\ell$.

A3. The system matrices $\mathbf{c}_t$, $\mathbf{H}_t$, $\mathbf{G}_t$, $\mathbf{a}_t$, $\mathbf{F}_t$, and $\mathbf{R}_t$ are known function of the model parameters $\boldsymbol{\theta}$ and of the contemporaneous variable $\mathbf{S}_t$ only.

A4. Given $\mathbf{S}_t$, the matrices $\mathbf{c}_t$, $\mathbf{H}_t$, $\mathbf{G}_t$, $\mathbf{a}_t$, $\mathbf{F}_t$, and $\mathbf{R}_t$, do not depend on the transition probabilities $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_\ell, \cdots)$.

A5. The eigenvalues of the transition matrix $\mathbf{F}_t$ are either less than or equal to one in modulus.

As underlined in A1, **DMM** focuses on conditionally Gaussian state space models. Assumption A2 restricts the dynamics of the discrete latent variables to Markov processes. This includes as a special case independent random variables with Bernoulli or categorical distribution. Assumptions A3 and A4 are standard hypothesis that help simplifying the MCMC simulations, as will be seen in Section 3. Finally, Assumption A5 excludes the case of regimes with explosive roots.

Without loss of generality, we impose that two variables S_{it} and S_{jt} do not impact the same matrix, so the maximum number of discrete latent variables $\mathtt{ns}$ is equal to six. This is not

restrictive since any two discrete variables with finite number of states can always be merged into a new variable defined by the cartesian product $S_{it} \times S_{jt}$. Hence the number of values that each matrix can take is equal to the number of states of the related discrete variable: for instance if $\mathbf{F}_t$ depends on S_{jt} then $\mathbf{F}_t \in \{\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_{\mathtt{ns}_j}\}$. We shall denote Z_t the Cartesian product of all discrete variables, i.e. $Z_t = (S_{1t} \times \cdots S_{\ell t} \times \cdots)$; for instance if the model includes two discrete variables S_{1t} and S_{2t} taking values in $\{1, 2\}$, then $Z_t \in \{1, 2, 3, 4\}$ corresponding to $(S_{1t}, S_{2t}) \in \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

We illustrate how to cast dynamic mixture models into the state space format (2.1) with two examples.

2.2 Example 1: Nile riverflow

The Nile riverflow series is a popular case study in the statistical literature (Cobb, 1978). The data are measurements of the annual flow volume at Aswan from 1871 to 1970. Mendelssohn (2011) fits a local level model with intervention variables that capture a level shift in 1899 and an additive outlier in 1913. He attributes these irregularities to the building of the Aswan dam and to a year of exceptionally low rainfalls. Rather than specifying the intervention dates, **DMM** performs outlier detection through some discrete latent variables that control the occurrence of irregularities as in:

$$\begin{aligned} y_t &= \mu_t + e_t \\ e_t &= [(S_{1t} - 1)\delta^{1/2} + (2 - S_{1t})]V_e^{1/2}\epsilon_{et} \\ \mu_t &= \mu_{t-1} + (S_{2t} - 1)V_\mu^{1/2}\epsilon_{\mu t} \end{aligned} \tag{2.2}$$

where ϵ_{et} and $\epsilon_{\mu t}$ are Gaussian errors with unit variance, and S_{1t} and S_{2t} are independent Bernoulli variables taking values in $\{1, 2\}$ with probability $\Pr(S_{\ell t} = 1) = \pi_{\ell 1}$, $\ell = 1, 2$. The irregular component e_t has variance V_e if $S_{1t} = 1$ and δV_e otherwise. The restriction $\delta > 1$ implies that $S_{1t} = 2$ corresponds to periods of shocks with larger variance. The level μ_t remains constant as long as $S_{2t} = 1$ and jumps to a new state when $S_{2t} = 2$.

Model (2.2) can be put into the state space format (2.1) by setting $\mathbf{x}_t = \mu_t$, $\mathbf{u}_t = (\epsilon_{et}, \epsilon_{\mu t})'$, $\mathbf{H}_t = 1$, $\mathbf{G}_t = ([(S_{1t} - 1)\delta^{1/2} + (2 - S_{1t})]V_e^{1/2}, 0)$, $\mathbf{F}_t = 1$, $\mathbf{R}_t = (0, (S_{2t} - 1)V_\mu^{1/2})$, and $\mathbf{a}_t = \mathbf{c}_t = 0$.

2.3 Example 2: US business cycle

Hamilton (1989) analysed the US business cycle using the Markov switching model:

$$\begin{aligned} y_t - \mu_t &= \sum_{j=1}^{p} \phi_j (y_{t-j} - \mu_{t-j}) + e_t \\ \mu_t &= \alpha_1 (2 - S_{1t}) + \alpha_2 (S_{1t} - 1) \end{aligned} \tag{2.3}$$

where y_t is the US real GNP growth rate and S_{1t} takes values in $\{1, 2\}$ according to a Markov process with transition probability $\pi_{1\,ij}$, $i, j = 1, 2$. The GNP growth switches between α_1 when $S_{1t} = 1$ and α_2 when $S_{1t} = 2$, where the restriction $\alpha_2 > \alpha_1$ identifies $S_{1t} = 2$ with periods of expansion. Hamilton used data up to 1984 but later studies have pointed out a subsequent decline in the series volatility, also known as the Great Moderation (see Kim and Nelson, 1999, and McConnell and Perez-Quiros, 2000). McConnell and Perez-Quiros introduce heteroskedasticity by assuming that the variance of the measurement shock takes two values according to a further Markov latent variable $S_{2t} = \{1, 2\}$ as in:

$$e_t = [(S_{2t} - 1)\delta^{1/2} + (2 - S_{2t})]\, V_e^{1/2} \epsilon_{et} \tag{2.4}$$

where ϵ_{et} is a standard normal variate. The transition probabilities for S_{2t} are denoted $\pi_{2\,ij}$, $i, j = 1, 2$. The restriction $0 < \delta < 1$ associates $S_{2t} = 2$ to the low volatility regime.

Model (2.3)-(2.4) can be put into the format (2.1) by setting $\mathbf{x}_t = (\mu_t, y_t - \mu_t, y_{t-1} - \mu_{t-1}, \cdots, y_{t-p+1} - \mu_{t-p+1})'$, $\mathbf{u}_t = e_t$, $\mathbf{H}_t = (1, 1, 0, \cdots, 0)$, $\mathbf{a}_t = (\alpha_1(2 - S_{1t}) + \alpha_2(S_{1t} - 1), 0, \cdots, 0)'$,

$$\mathbf{F}_t = \begin{pmatrix} 0 & 0 & & \cdots & 0 & 0 \\ 0 & \phi_1 & \phi_2 & \cdots & \phi_{p-1} & \phi_p \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ & & & \vdots & & \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

and $\mathbf{R}_t = V_e^{1/2}(0, (S_{2t} - 1)\delta^{1/2} + 2 - S_{2t}, 0, \cdots, 0)'$. The remaining quantities $\mathbf{G}_t$ and $\mathbf{c}_t$ are equal to zero.

3 Bayesian inference

3.1 Prior distributions

We further assume:

A6. All parameters in $\boldsymbol{\theta}$ are a priori independent, and $\boldsymbol{\theta}$ is a priori independent from $\boldsymbol{\pi}$.

For the elements of $\boldsymbol{\theta}$, three possible prior distributions are available: normal (`NT`), beta (`BE`), and inverse gamma (`IG`). These three distributions are flexible enough to characterize most beliefs. The normal distribution is characterized in terms of mean and variance. The `IG` distribution is typically used for scale parameters; it is parameterized as in Bauwens et al. (1999, p.292), i.e. $\theta_i \sim \texttt{IG}(s,\nu)$ implies $E(\theta_i) = s/(\nu-2)$ and $V(\theta_i) = 2s^2/\{(\nu-4)(\nu-2)^2\}$. Each parameter is defined over a finite support (a,b) that must be specified together with the distribution hyperparameters. Hence the beta-distributed variables are not restricted to the $(0,1)$ interval. For infinite supports, it will suffice to set the bounds to very large values.

The transition probabilities $\boldsymbol{\pi}$ are assumed to be Dirichlet distributed. For each sequence $\mathbf{S}_\ell = (S_{\ell 1}, \cdots, S_{\ell \mathtt{T}})$ with transition probability $\boldsymbol{\pi}_\ell$, the number of independent Dirichlet distributions that must be elicited on $\boldsymbol{\pi}_\ell$ depends on the process considered for $\mathbf{S}_\ell$. For instance, if $\mathbf{S}_\ell$ is a sequence of independent categorical variables, then $\boldsymbol{\pi}_\ell = (\pi_{\ell 1}, \cdots, \pi_{\ell \mathtt{ns}_\ell})$ follows a Dirichlet distribution. If $\mathbf{S}_\ell$ follows instead a Markov process, then $\boldsymbol{\pi}_\ell$ has $\mathtt{ns}_\ell^2$ elements (see Assumption A2), and $\mathtt{ns}_\ell$ independent Dirichlet distributions must be elicited on each subset $\boldsymbol{\pi}_{\ell j} = (\pi_{\ell 1 j}, \cdots, \pi_{\ell \mathtt{ns}_\ell j})$, $j = 1, \cdots, \mathtt{ns}_\ell$. Examples of prior elicitation are given in Section 5.

3.2 Posterior inference

DMM produces samples from the joint posterior distribution $f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}, \mathbf{x}|\mathbf{y})$, where for convenience we have omitted the exogenous variables $\mathbf{z}$ from the information set. For any variable $\mathbf{w}_t$ let $\mathbf{w}$ denote the vector $\mathbf{w} = (\mathbf{w}_1, \cdots, \mathbf{w}_\mathtt{T})$. We use the factorization:

$$f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}, \mathbf{x}|\mathbf{y}) = f(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}, \mathbf{y}) f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})$$

Posterior samples of the state vector $\mathbf{x}$ are drawn off-line using the simulation smoother proposed by Durbin and Koopman (2002). Samples from $f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})$ are obtained with the following Gibbs scheme:

$$f(\boldsymbol{\theta}|\mathbf{S}, \boldsymbol{\pi}, \mathbf{y}),\ \Pr(\mathbf{S}\ |\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{y}),\ f(\boldsymbol{\pi}|\boldsymbol{\theta}, \mathbf{S}, \mathbf{y})$$

Assumptions A4 and A6 imply that the first full conditional verifies:

$$f(\boldsymbol{\theta}|\mathbf{S}, \boldsymbol{\pi}, \mathbf{y}) \propto f(\mathbf{y}|\mathbf{S}, \boldsymbol{\theta})\ f(\boldsymbol{\theta})$$

The conditionally Gaussian hypothesis A1 makes possible the evaluation of the augmented likelihood $f(\mathbf{y}|\mathbf{S}, \boldsymbol{\theta})$ by Kalman filtering (see Kalman, 1960). For non-stationary state variables, diffuse initial conditions are handled as in Koopman (1997). For stationary state variables say $\mathbf{x}_t^*$, the recursions are initialized using the unconditional mean and covariance matrix $E(\mathbf{x}_1^*|\boldsymbol{\theta}, \mathbf{S})$

and $V(\mathbf{x}_1^*|\boldsymbol{\theta},\mathbf{S})$. The unconditional covariance matrix of the stationary elements of the state vector is calculated as in Kitagawa (1977). Program **DMM** draws $\boldsymbol{\theta}$ one parameter at a time from $f(\theta_i|\boldsymbol{\theta}_{-i},\mathbf{S},\mathbf{y})$ using the stepping out slice sampler proposed by Neal (2003).

The **S**-sequence is drawn with either the single-move algorithm devised by Gerlach et al. (GCK, 2000), or the multi-move adaptive MH sampler given in Fiorentini, Planas and Rossi (2012). The CGK sampler draws one variable at a time from the full conditional distribution $\Pr(\mathbf{S}_t|\mathbf{S}_{-t},\ \boldsymbol{\theta},\boldsymbol{\pi},\mathbf{y})$. When the discrete latent variables are conditionally dependent it is more efficient to sample them in blocks: given a block length h, the multi-move sampler draws from $\Pr(\mathbf{S}_t,\mathbf{S}_{t+1},\cdots,\mathbf{S}_{t+h-1}|\mathbf{S}_{-t,\cdots,t+h-1},\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{y})$. Both algorithms yield the full $\mathbf{S}$ sequence marginally to $\mathbf{x}$ in $O(\mathtt{T})$ operations.

It remains to describe the sampler of $\boldsymbol{\pi}$ given $\mathbf{S}$, $\boldsymbol{\theta}$, and $\mathbf{y}$. Assumptions A4 and A6 imply that given $\mathbf{S}$, $\boldsymbol{\pi}$ does not depend on $\boldsymbol{\theta}$ and $\mathbf{y}$: $f(\boldsymbol{\pi}|\boldsymbol{\theta},\mathbf{S},\mathbf{y}) \propto f(\mathbf{S}|\boldsymbol{\pi})f(\boldsymbol{\pi})$. We first consider the case of a sequence $\mathbf{S}_\ell$ of independent variables with $Pr(S_{\ell t}=k)=\pi_{\ell k}$ for $k=1,\cdots,\mathtt{ns}_\ell$. Under the assumption $\boldsymbol{\pi}_\ell=(\pi_{\ell 1},\cdots,\pi_{\ell\,\mathtt{ns}_\ell}) \sim Dirichlet(\alpha_{\ell\ 1},\cdots,\alpha_{\ell\ \mathtt{ns}_\ell})$, the full conditional $f(\boldsymbol{\pi}_\ell|\mathbf{S}_\ell)$ is a Dirichlet distribution with hyperparameters $(\alpha^*_{\ell\ 1},\cdots,\alpha^*_{\ell\ \mathtt{ns}_\ell})$ such that:

$$\alpha^*_{\ell\ j}=\alpha_{\ell\ j}+\sum_{t=1}^{\mathtt{T}}\mathbf{1}(S_{\ell t}=j), \qquad j=1,\cdots,\mathtt{ns}_\ell$$

where $\mathbf{1}(\cdot)$ is the indicator function.

When $\mathbf{S}_\ell$ is a Markov sequence with $Pr(S_{\ell t}=k|S_{\ell t-1}=j)=\pi_{\ell kj}$ for $k,j=1,\cdots,\mathtt{ns}_\ell$, then a set of transition probabilities $\boldsymbol{\pi}_{\ell j}=(\pi_{\ell 1 j},\cdots,\pi_{\ell\mathtt{ns}_\ell j})$ is attached to each state j. Since $j=1,\cdots,\mathtt{ns}_\ell$, the Markov process $\mathbf{S}_\ell$ is characterized by ns_ℓ^2 transition probabilities that are put together into the vector $\boldsymbol{\pi}_\ell=(\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell})$. The full conditional distribution of $\boldsymbol{\pi}_\ell$ is such as:

$$\begin{aligned}
f(\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell}|\mathbf{S}_\ell) &\propto \Pr(\mathbf{S}_\ell|\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell})\prod_{k=1}^{\mathtt{ns}_\ell}f(\boldsymbol{\pi}_{\ell k}) \\
&\propto \Pr(S_{\ell 1}|\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell})\prod_{t=2}^{\mathtt{T}}\Pr(S_{\ell t}|S_{\ell t-1},\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell})\prod_{k=1}^{\mathtt{ns}_\ell}f(\boldsymbol{\pi}_{\ell k}) \\
&\propto \Pr(S_{\ell 1}|\boldsymbol{\pi}_{\ell 1},\cdots,\boldsymbol{\pi}_{\ell\mathtt{ns}_\ell})\prod_{k=1}^{\mathtt{ns}_\ell}\prod_{t\in I_k}^{\mathtt{T}}\Pr(S_{\ell t}|S_{\ell t-1}=k,\boldsymbol{\pi}_{\ell k})f(\boldsymbol{\pi}_{\ell k})
\end{aligned}$$

where $I_k=\{t\geq 2: S_{\ell t-1}=k\}$. The term $\prod_{k=1}^{\mathtt{ns}_\ell}\prod_{t\in I_k}^{\mathtt{T}}\Pr(S_{\ell t}|S_{\ell t-1}=k,\boldsymbol{\pi}_{\ell k})f(\boldsymbol{\pi}_{\ell k})$ is proportional to the product of $\mathtt{ns}_\ell$ independent Dirichlet distributions. To remove dependence on the initial condition $S_{\ell 1}$, this product is taken as proposal in a MH step with acceptance probability given by $\min\{1,\Pr(S_{\ell 1}|\boldsymbol{\pi}^*_\ell)/\Pr(S_{\ell 1}|\boldsymbol{\pi}_\ell)\}$, where $\boldsymbol{\pi}^*_\ell$ is the candidate vector and $\boldsymbol{\pi}_\ell$ is the previously sampled value.

DMM produces forecasts if requested. For instance, one-step-ahead forecasts are obtained by

sampling $\mathbf{S}_{T+1}$ from $\Pr(\mathbf{S}_{T+1}|\mathbf{S}_T, \boldsymbol{\pi})$, $\mathbf{x}_{T+1}$ from $f(\mathbf{x}_{T+1}|\mathbf{S}_{T+1}, \mathbf{x}_T, \boldsymbol{\theta})$, and $\mathbf{y}_{T+1}$ from $f(\mathbf{y}_{T+1}|\mathbf{x}_{T+1}, \mathbf{S}_{T+1}, \boldsymbol{\theta})$, where $(\mathbf{x}_T, \mathbf{S}_T, \boldsymbol{\theta}, \boldsymbol{\pi})$ have been drawn from the joint distribution $f(\mathbf{x}_T, \mathbf{S}_T, \boldsymbol{\theta}, \boldsymbol{\pi}|\mathbf{y})$. Iterating forward this data augmentation scheme yields forecasts of $\mathbf{S}_t$, $\mathbf{x}_t$, and $\mathbf{y}_t$ until any desired horizon.

DMM simulates the missing observations if any. Under A1, the measurement equation (2.1) implies:

$$f(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\theta}, \mathbf{S}_t) = N(\mathbf{c}_t\mathbf{z}_t + \mathbf{H}_t\mathbf{x}_t, \mathbf{G}_t\mathbf{G}_t')$$

This density is used to simulate the missing observations given posterior draws of $\mathbf{x}_t, \boldsymbol{\theta}, \mathbf{S}_t$.

Other sampling strategies are of course possible. A first alternative would be to introduce the state vector $\mathbf{x}$ within the Gibbs loop and to sample $\boldsymbol{\theta}$ from $f(\boldsymbol{\theta}|\mathbf{x}, \mathbf{S}, \mathbf{y})$ by taking benefit of conjugate priors as in Planas, Fiorentini, and Rossi (2008). Model dependence makes however this strategy inadequate for general scope programs. Another possibility is to sample the discrete latent variable $\mathbf{S}$ given the state vector $\mathbf{x}$ as in Chib (1996). This possibility is not always feasible: for instance in Example 2, the distribution of S_{1t} given μ_t, α_0, and α_1 is degenerate.

3.3 Marginal likelihood estimators

DMM calculates two estimators for the data marginal likelihood: reciprocal importance sampling and bridge sampling. The first estimator proposed by Gelfand and Dey (1996) is based on:

$$f(\mathbf{y}) = \left\{ \int \frac{q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})}{f(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}) f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})} dF(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y}) \right\}^{-1}$$

where $F(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})$ refers to the cumulative posterior distribution. The importance function $q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$ stabilizes the estimator by excluding points with low likelihood values; otherwise the estimator has infinite variance (see Geweke, 1999, p.47). We use $q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}) = q_1(\boldsymbol{\theta})q_2(\boldsymbol{\pi})q_3(\mathbf{S})$, where $q_1(\boldsymbol{\theta})$ is a normal distribution with mean and covariance matrix estimated from the posterior samples and truncated at level $0\%, 5\%, 15\%, \cdots, 95\%$, $q_2(\boldsymbol{\pi})$ is a Dirichlet with parameters estimated from the posterior samples of $\boldsymbol{\pi}$, and $q_3(\mathbf{S})$ is a non-homogenous Markov chain as discussed in Fiorentini, Planas, and Rossi (2012).

The bridge sampling estimator devised by Meng and Wong (1996) re-weights both the importance function and the posterior density with a bridge function. Let I and I_q denote the support of the parameter posterior distribution and of the importance density $q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$, respectively. The bridge function, say $h(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$, is assumed to be defined over $I \bigcap I_q$. Let $Q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$ be the

cumulative distribution associated with $q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$. The estimator is obtained from the identity:

$$f(\mathbf{y}) = \frac{\int_{I_q} \frac{h(\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{S})}{q(\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{S})}\, dQ(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})}{\int_{I} \frac{h(\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{S})}{f(\mathbf{y}|\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{S})f(\boldsymbol{\theta},\boldsymbol{\pi},\mathbf{S})}\, dF(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})}$$

The bridge sampling formula generalizes importance sampling methods: for instance setting $h(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}) = q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S})$ yields the reciprocal importance sampling estimator. As optimal choice Meng and Wong propose a recursive procedure based on:

$$h(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}) \propto \frac{q(\boldsymbol{\theta}, \boldsymbol{\pi}\mathbf{S}) f(\boldsymbol{\theta}, \boldsymbol{\pi}\mathbf{S}|\mathbf{y})}{n_q q(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}) + n_y f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})}$$

where the constants n_q and n_y refer to the number of draws from the importance function and from the posterior density, respectively. The recursions enter through the term $f(\boldsymbol{\theta}, \boldsymbol{\pi}, \mathbf{S}|\mathbf{y})$ that involves a preliminary marginal likelihood estimate. **DMM** uses one estimate returned by reciprocal importance sampling. Marginal likelihood estimates are reported in logs, with standard deviations that are obtained with the delta-method.

3.4 Data simulation

Alternatively, **DMM** can be used to generate a set of observations together with the associated unobserved variables for given model parameters. No posterior analysis is performed when this option is activated.

4 Running DMM

To run **DMM**, two inputs files are needed: the first, say `example.nml`, contains the model settings, the prior distributions, and the data; the second, say `design.dll`, is a very simple DLL that defines the system matrices of the state space representation (2.1). The program is called from the MS-DOS prompt typing:

```
DMM example.nml
```

The path must be added if `example.nml` does not belong to the same directory as the `DMM` executable file. We detail below the contents of the two input files. The program output is described in Section 4.3.

4.1 The `example.nml` file

Inputs are organized in namelists. Each namelist appears after the ampersand '&' and closes with '&end', as in '&namelist1 arg1= ... arg2=... &end'. The name of the namelists is case-sensitive, but the order of the namelists and of the arguments they contain is irrelevant.

The namelists are described in Table 1. Four are compulsory: `ssm`, `prior`, `mcmc`, and `dataset`. To give an overview, the first, `ssm`, sets the system dimensions, the number of discrete latent variables `nv`, and specifies the name of the DLL file. It also includes a facility for verifying that the system matrices are correctly coded that can be activated by setting `check = Y`. The second, `prior`, states the number of model parameters, their prior densities, and the prior hyperparameters. The third, `mcmc`, controls the MCMC simulations and computes the marginal likelihood when `marglik = Y`. The fourth, `dataset`, contains the number of endogenous series `ny`, the number of exogenous series `nz`, the number of forecasts requested `nf`, the number of observations `T`, and the data. If `datasim = Y` the program simulates the data.

The dataset is loaded as a matrix of total dimension $(\texttt{T} + \texttt{nf}) \times (\texttt{ny} + \texttt{nz})$, where columns are separated by spaces. The missing observations on the endogenous series must be coded as '-99999'; no missing values in the first `d(1)` observations are allowed, where `d(1)` represents the model order of integration. Users must take care of inserting a column of ones as exogenous series if the measurement equation (2.1) includes a constant. If `nf` forecasts are required, then the exogenous series must also cover the forecasting period while the endogenous series must be completed from `T+1` to `T+nf` with any arbitrary value; for instance one may use '-99999'. If `datasim = Y`, then the observations are generated given the exogenous variables and some parameter values.

The namelists that describe the discrete latent processes are labeled `S1`, `S2`, $\cdots$, `S6`. Each carries information about the process dynamic (I=independent, M=Markov), the number of states, the Dirichlet hyperparameters, and the system matrices that switch according to the variable. Only `nv` namelists for the discrete latent processes must be specified.

The default values are `nv = 0`,`check = N`, `seed = 0`, `thin = 1`, `burnin = 1000`, `simulrec = 5000`, `hbl = 1`, `marglik = N`, `nf = 0`, and `datasim = N`. Two examples of input files are given in Section 5.

TABLE 1 **DMM** input namelists

Name	Function	Argument	Value
`ssm`	system dimensions	`nx`: dimension of $\mathbf{x}_t$	integer
		`d(1)`: maximum order of integration of $\mathbf{x}_t$	$0, 1, 2$
		`d(2)`: # of non-stationary elements in $\mathbf{x}_t$	$0, 1, \cdots$,`nx`
		`nu`: # of shocks	integer
		`nv`: # of discrete latent variables	$0, 1, \cdots, 6$
		`dllname`: path and name of the dll file	`design.dll`
		`check`: print the system matrices **c**,**H**,**G**,**a**,**F**,**R**	`Y`, `N`
`prior`	$\boldsymbol{\theta}$ prior	`nt`: dimension of $\boldsymbol{\theta}$	integer
		`pdftheta(1)`: prior distribution for θ_1	`NT`, `BE`, `IG`
		`hyptheta(1,1)`: prior hyperparameters for θ_1	four real numbers
		...	...
		`pdftheta(nt)`: prior distribution for $\theta_{\texttt{nt}}$	`NT`, `BE`, `IG`
		`hyptheta(1,nt)`: prior hyperparameters for $\theta_{\texttt{nt}}$	four real numbers
`mcmc`	MCMC options	`seed`: initialise the random number generator	from 0 to 999
		`thin`: record every j iterations	integer
		`burnin`: # of burn-in iterations	integer
		`simulrec`: # of simulations to record	integer
		`hbl`: block length for sampling **S**	$1, 2, \cdots$,`T`
		`marglik`: marginal likelihood	`Y`, `N`
`dataset`	data	`T`: # of observations	integer
		`ny`: # of endogenous series	integer
		`nz`: # of exogenous series	integer
		`nf`: # of forecasts	integer
		`datasim`: simulate data	`Y`, `N`
		`obs`: a matrix of dimension $(\texttt{T} + \texttt{nf}) \times (\texttt{ny} + \texttt{nx})$	real numbers
`S1`	S_1 process	`dynS1`: S_1 independent/Markov dynamics	I, M
		`ns1`: # of states for S_1	$2, 3, \cdots$
		`hypS1`: Dirichlet hyperparameters	see examples
		`matS1`: matrices impacted by S_1	**c**,**H**,**G**,**a**,**F**, or **R**
...	...	...	...
`S6`	S_6 process	`dynS6`: S_6 dynamics	I, M
		`ns6`: # of states for S_6	$2, 3, \cdots$
		`hypS6`: Dirichlet hyperparameters	see examples
		`matS6`: matrices impacted by S_6	**c**,**H**,**G**,**a**,**F**, or **R**

4.2 The `design.dll` file

The `design.dll` file receives $\boldsymbol{\theta}$ (`theta`) as input and returns the system matrices of the state space representation (2.1). To create the DLL users have to write a `Fortran` source code with following declarations:

```
SUBROUTINE DESIGN(ny,nz,nx,nu,ns,nt,theta,c,H,G,a,F,R)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : 'design_' ::  DESIGN
INTEGER ny,nz,nx,nu,ns(6),nt
DOUBLE PRECISION theta(nt)
DOUBLE PRECISION c(ny,max(1,nz),ns(1)),H(ny,nx,ns(2)),G(ny,nu,ns(3))
DOUBLE PRECISION a(nx,ns(4)),F(nx,nx,ns(5)),R(nx,nu,ns(6))
Body text:  matrix specification
RETURN
END
```

The first line is the classic subroutine declaration in `Fortran`. The second line defines the DLL interface to be called by **DMM**. We underline that the name, structure, inputs and outputs of the subroutine are fixed whereas, beside the file extension, the name of the file is free. The input parameters `ny,nz,nx,nu,ns,nt` take the values specified in the `example.nml`, while the vector of parameters `theta` is updated automatically by **DMM**. The users only need to fill the vector and matrices $\mathbf{c}, \mathbf{H}, \mathbf{G}, \mathbf{a}, \mathbf{F}, \mathbf{R}$ as a function of `theta(1)`,`theta(2)`,..., `theta(nt)`. The last two lines end the subroutine.

DMM requires that the state vector $\mathbf{x}_t$ contains first the non-stationary elements. For instance, if the order of integration of the model is one but there are two non-stationary elements, i.e. `d(1)=1` and `d(2)=2` in the namelist `ssm`, then the vector $\mathbf{x}_t$ must begin with the two elements that are non-stationary. This requisite arises because stationary and non-stationary elements of the state are treated differently in the Kalman filter initialization as explained in Section 3.

To create `design.dll` from `design.for`, use can be made of the GNU Fortran compiler for Windows which is freely donwloadable at *gcc.gnu.org/wiki/GFortran*. Once installed, the compiler can be called at the command prompt with the following command-line arguments:

```
gfortran -shared -o design.dll design.for
```

Examples of `design.for` files are given in Section 5.

The instruction `Check = Y` in the namelist `ssm` enables users to visualize the system matrices returned by the routine for some given values of the model parameters.

4.3 The output

DMM produces up to ten ASCII files. The filenames have in common the root of the `example.nml`. The files that contain simulated quantities also have the `seed` number appended. This makes possible comparing chains with different starting point as a simple convergence check. Assuming `seed` $= 0$, the output files are identified by the file extension according to:

- `example.pri`: reports the prior distributions with their hyperparameters.
- `example0.par`: the `simulrec` $\times$ (`nt` $+ dim(\boldsymbol{\pi})$) matrix of posterior samples of $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$. If `datasim = Y`, then `simulrec=1` and the file contains only the parameter values used for generating the endogenous series; $\boldsymbol{\theta}$ is printed first and $\boldsymbol{\pi}$ follows.
- `example0.unb`: the `simulrec` $\times$ (`nx` $\times$ `T`) matrix of posterior samples of the state vector. Each row contains the first state element from $t = 1$ to `T`, then the second state element from $t = 1$ to `T`, and so on until the `nx`-th state element. If `datasim = Y`, the file contains the simulated state elements in a `T` $\times$ `nx` matrix.
- `example0.inn`: the `simulrec` $\times$ (`ny` $\times$ `T`) matrix of innovations, i.e. $\mathbf{y}_t - E(\mathbf{y}_t|\mathbf{y}_1, \cdots, \mathbf{y}_{t-1})$. Each row contains the innovations on the first series from $t = 1$ to `T`, then the innovations on the second series, and so on. Zeros are reported in case of missing observations.
- `example0.mis`: the `simulrec` $\times$ `nmis` matrix of estimated missing observations, where `nmis` is the total number of missing observations. Each row contains the missing observation(s) in chronological order.
- `example0.dis`: the `simulrec` $\times$ `T` matrix of posterior sample of $\mathbf{Z}$, where $\mathbf{Z} = (\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_{\mathtt{nv}})$. The following mapping is used: if the model includes two discrete variables S_{1t} and S_{2t} taking values in $\{1, 2\}$, then $Z_t \in \{1, 2, 3, 4\}$ corresponds to $(S_{1t}, S_{2t}) \in \{(1, 1), (1, 2), (2, 1), (2, 2)\}$. If `datasim = Y`, the file contains the simulated discrete variable $\mathbf{Z}$ in one row.
- `example0.fst`: the `simulrec` $\times$ (`nf`(`nx` + `ny` + 1)) matrix of forecasts of the state, of the observations, and of the Z_t variable. Each row contains the first state element from `T` + 1 to `T` + `nf`, then the second state element from `T` + 1 to `T+nf`, and so on until the `nx`-th state element. Then the series forecasts are written with the same order for series 1 to `ny`. Each row ends with the `nf` forecasts of $\mathbf{Z}_t$.

- `example0.dat`: the `T` $\times$ `ny` matrix of simulated series. It is produced only if `simulation = Y`.

- `example0.ml`: the log-marginal likelihood estimates with associated standard errors. It is produced only if `Marglik` = Y. The reciprocal importance sampling estimates are reported in logs for truncation level at 0%, 5%,$\cdots$, and 95%. Two bridge sampling estimates are displayed: the first one is obtained without iterating and the second one after ten iterations. All estimates are given with a standard error.

- `example.chk`: the file shows the system matrices associated to a given value of `theta` simulated from the prior distribution. It is produced only if `Check = Y`.

These ASCII files can be imported for further analysis in any common statistical package such as R or MatLab. A MatLab code that produces main summary statistics and plots of the posterior distributions is available upon request to the authors.

5 Examples

5.1 Nile riverflow

The model specification (2.2) is completed with the following priors on $\boldsymbol{\theta} = (V_e, V_\mu, \delta)$ and $\boldsymbol{\pi} = (\pi_{11}, \pi_{21})$: $V_j \sim \texttt{IG}(6 \times 10^4, 6) \times I_{(0,\ 5\times10^4)}, j = e, \epsilon,\ \delta \sim \texttt{BE}(2,4) \times I_{(1,\ 20)}$, and $\pi_{j1} \sim Dirichlet(16, 2),\ j = 1, 2$. The `nile.nml` file contains the following namelists:

```
&ssm
nx=1 nu=2 d=1 1 nv=2 dllname=nile.dll
&end

&prior
nt = 3
pdftheta(1) = IG hyptheta(1,1) = 60000 6 0 50000
pdftheta(2) = IG hyptheta(1,2) = 60000 6 0 50000
pdftheta(3) = BE hyptheta(1,3) = 2 4 1 20
&end

&S1
dynS1=I nS1=2 hypS1(1,1)=16 2 matS1=G
&end

&S2
dynS2=I nS2=2 hypS2(1,1)=16 2 matS2=R
&end

&mcmc
seed=0 thin=1 burnin=1000 simulrec=5000
&end

&dataset
T=100 ny=1 nz=0 nf=10 obs =
1120
1160
...
740
&end
```

The namelist `prior` sets the prior distributions and hyperparameters for `theta`. For instance, `pdftheta(1) = IG` states that `theta(1)` is IG-distributed with scale, degree of freedom, and support that are specified via the instruction `hyptheta(1,1)`. Notice that the first dimension of

hyptheta is fixed to one, while the second dimension corresponds to the position of the parameter in the vector theta.

The namelist S1 contains four entries: dynS1=I declares that $\mathbf{S}_1$ is a sequence of independent variables with nS1 = 2 states that impact the matrix matS1 = $\mathbf{G}$. The parameters of the categorical distribution $\Pr(S_{1t} = i)$, $i = 1, 2$, are Dirichlet distributed with hyperparameters $(16, 2)$ as entered via the argument hypS1(1,1). The namelist S2 is filled similarly.

The design subroutine written in the Nile.for file is reported below.

The state space format discussed in Section 2.2 is implemented in the Nile.for file as shown below.

```
SUBROUTINE DESIGN(ny,nz,nx,nu,ns,nt,theta,c,H,G,a,F,R)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : 'design_' ::  DESIGN
INTEGER ny,nz,nx,nu,ns(6),nt
DOUBLE PRECISION theta(nt)
DOUBLE PRECISION c(ny,max(1,nz),ns(1)),H(ny,nx,ns(2)),G(ny,nu,ns(3))
DOUBLE PRECISION a(nx,ns(4)),F(nx,nx,ns(5)),R(nx,nu,ns(6))

 c(1,1,1)     = 0.D0
 H(1,1,1)     = 1.D0
 G(1,1,1)     = DSQRT(theta(1))
 G(1,1,2)     = DSQRT(theta(1)*theta(3))
 G(1,2,1:2) = 0.D0
 a(1,1)       = 0.D0
 F(1,1,1)     = 1.D0
 R(1,1:2,1)   = 0.D0
 R(1,1,2)     = 0.D0
 R(1,2,2)     = DSQRT(theta(2))

RETURN
END
```

In agreement with the namelists dynS1 and dynS2, the matrix $\mathbf{G}$ takes two values, $\mathbf{G}(:,:,1)$ and $\mathbf{G}(:,:,2)$, depending on S_{1t}. Similarly, $\mathbf{R}$ switches between $\mathbf{R}(:,:,1)$ and $\mathbf{R}(:,:,2)$ depending on S_{2t}. The model parameters are ordered as theta(1) = V_e, theta(2) = V_μ, and theta(3) = δ. The DSQRT instruction is the Fortran square root double precision function; other Fortran intrinsic functions that may turn useful are DEXP, DLOG, DCOS, DSIN - for more details, see the GNU Fortran reference manual at *http://gcc.gnu.org/onlinedocs/gcc-4.6.0/gfortran/*. To built the DLL from the Fortran file use can be made of the GNU Fortran compiler typing at the MS-DOS prompt:

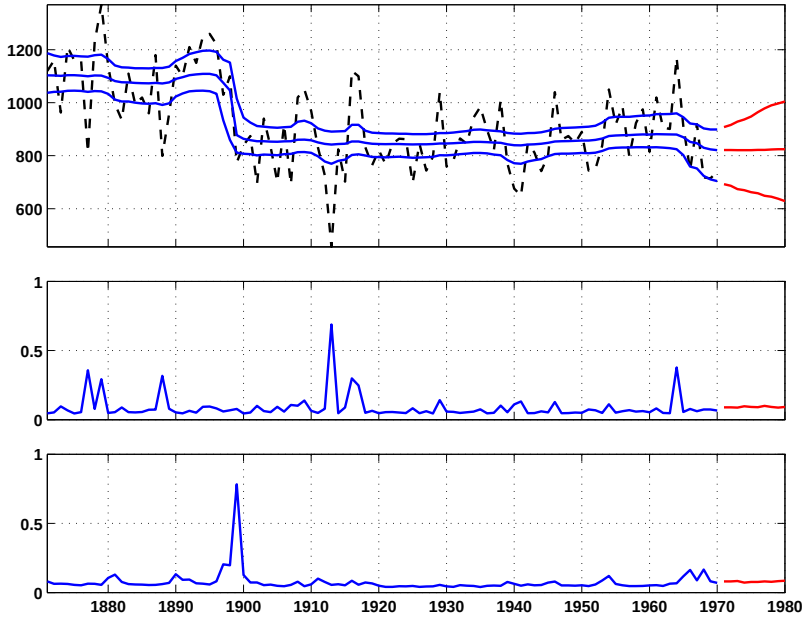
```
gfortran -shared -o Nile.dll Nile.for
```

Assuming `DMM.exe`, `Nile.dll` and `Nile.nml` are located in the same directory, the program is executed with the command:

```
DMM Nile.nml
```

To show some results, Figure 1 top panel displays the observed series in dashed, plus the trend posterior mean and forecasts together with the 5% and 95% percentiles. The middle and bottom panels report the posterior probability of an outlier on the irregular component, i.e. $\Pr(S_{1t} = 2|\mathbf{y})$, and of a level shift, i.e. $\Pr(S_{2t} = 2|\mathbf{y})$. As can be seen, the dating agrees with the results in Mendelssohn (2011).

FIGURE 1 Latent variables in Nile example



Notes: top panel: observed series (dashed), trend posterior mean and forecasts with 5th and 95th percentiles; middle panel: posterior probability of an additive outlier $\Pr(S_{1t} = 2|\mathbf{y})$; bottom panel: posterior probability of a level shift $\Pr(S_{2t} = 2|\mathbf{y})$.

Table 2 below shows the posterior mode of the parameters together with the 90% highest posterior density region (HPD). It may be noticed that, while the posterior mode of the trend shock variance V_ϵ is larger than that reported in Bell (2011), our trend estimates is smoother than that in Bell. This is due to the properties of the trend process in (2.2) which is by construction smoother than a random walk where shocks occur at every period.

TABLE 2 Parameter posterior mode with 90% HPD for Nile example

	V_e ($\times 10^4$)	V_ϵ ($\times 10^4$)	δ	π_{11}	π_{21}
Mode	1.27	0.91	3.77	0.94	0.95
90% HPD	(0.47, 1.67)	(0.30,2.62)	(1.18,10.12)	(0.58,0.97)	(0.58,0.97)

5.2 US business cycle

We fit the model (2.3)-(2.4) to the quarterly real GDP data for the period 1953:II to 1999:II. We choose the autoregressive order $p = 1$ and set the following prior distributions on $\boldsymbol{\theta} = (\alpha_1, \alpha_2, \phi_1, \delta, V_e)$: $\alpha_1 \sim N(-.5, .2) \times I_{(-1,0)}$, $\alpha_2 \sim N(1, .2) \times I_{(0,2)}$, $\phi_1 \sim N(0, .1) \times I_{(-.9,.9)}$, $\delta \sim$ BE$(2, 4) \times I_{(0,1)}$, and $V_e \sim$ IG$(5, 6) \times I_{(0,5)}$. The Markov transition probabilities are distributed as: $(\pi_{\ell 11}, \pi_{\ell 21}) \sim Dirichlet(6, 2)$, and $(\pi_{\ell 12}, \pi_{\ell 22}) \sim Dirichlet(2, 18)$, $\ell = 1, 2$. The `mswitch.nml` file enters the data and the model information as shown below.

```
&ssm
nx=2 nu=1 d=0 0 nv=2 dllname=mswitch.dll
&end

&prior
nt = 5
pdftheta(1) = NT hyptheta(1,1) = -.5 .2 -1 0
pdftheta(2) = NT hyptheta(1,2) = 1 .2 0 2
pdftheta(3) = NT hyptheta(1,3) = 0 .1 -.9 .9
pdftheta(4) = BE hyptheta(1,4) = 2 4 0 1
pdftheta(5) = IG hyptheta(1,5) = 5 6 0 5
&end

&S1
dynS1=M nS1=2 hypS1(1,1)=6 2 hypS1(1,2)=2 18 matS1=a
&end

&S2
dynS2=M nS2=2 hypS2(1,1)=6 2 hypS2(1,2)=2 18 matS2=R
&end

&mcmc
seed=0 thin=1 burnin=1000 simulrec=5000
&end

&dataset
T=185 ny=1 nz=0 nf=10 obs =
0.7559
-0.6071
...
0.8242
&end
```

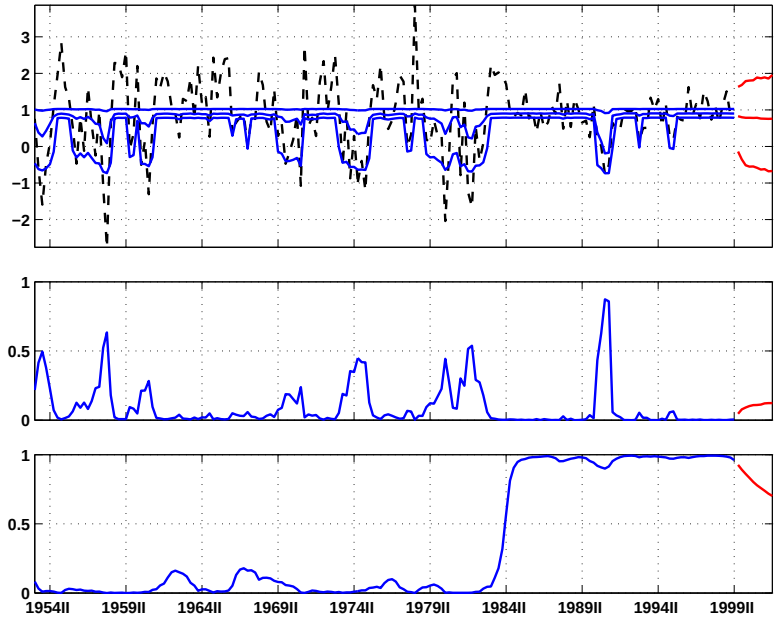
In this example, `dynS1=M` states that the dynamics of $\mathbf{S}_1$ is Markov. Since `nS1=2`, the transition matrix involves four parameters, $(\pi_{1\,11}, \pi_{1\,21})$ and $(\pi_{1\,12}, \pi_{1\,22})$, for which two independent Dirichlet distributions are specified with hyperparameters (6,2) and (2,18) as specified in the arguments `hypS1(1,1)` and `hypS1(1,2)`. The namelist `S2` is filled similarly. Notice that, as for the entry `hyptheta(1,·)`, the first index in `hypS1(1,·)` is fixed to one while the second index takes values from 1 to `nS1` since `nS1` Dirichlet distributions need to be specified. The state space model described in Section 2.3 is implemented in the `mswitch.for` file as shown below.

```
SUBROUTINE DESIGN(ny,nz,nx,nu,ns,nt,theta,c,H,G,a,F,R)
!DEC$ ATTRIBUTES DLLEXPORT, ALIAS : 'design_' ::  DESIGN
INTEGER ny,nz,nx,nu,ns(6),nt
DOUBLE PRECISION theta(nt)
DOUBLE PRECISION c(ny,max(1,nz),ns(1)),H(ny,nx,ns(2)),G(ny,nu,ns(3))
DOUBLE PRECISION a(nx,ns(4)),F(nx,nx,ns(5)),R(nx,nu,ns(6))

  c(1,1,1)     = 0.D0
  H(1,1:2,1)   = 1.D0
  G(1,1,1)     = 0.D0
  a(1,1)       = theta(1)
  a(1,2)       = theta(2)
  a(2,1:2)     = 0.D0
  F(1:2,1:2,1) = 0.D0
  F(2,2,1)     = theta(3)
  R(1,1,1:2)   = 0.D0
  R(2,1,1)     = DSQRT(theta(5))
  R(2,1,2)     = DSQRT(theta(4)*theta(5))

RETURN
END
```

Both $\mathbf{a}_t$ and $\mathbf{R}_t$ have one additional dimension as they are the arrays that are switching according to the discrete latent variables. The DLL is created via the command `gfortran -shared -o mswitch.dll mswitch.for`. Running `DMM mswitch.nml` yields the posterior samples. Figure 2 below displays the posterior mean of the discrete and continuous latent variables together with the data in dashed. In the last two decades, the model detects a low growth in 1981 and in 1991, in agreement with the NBER official business cycle dating. The evidence of a volatility break in 1984 is remarkably strong.

Table 3 below shows the posterior mode of the model parameters together with the 90% HPD. These results are in broad agreement with those in McConnell and Perez-Quiros (2000) in spite of a different estimation technique and a slightly different modelling.

FIGURE 2 Latent variables in US GDP example



Notes: top panel: observed series (dashed), trend posterior mean and forecasts with 5th and 95th percentiles; middle panel: posterior probability of a recession $\Pr(S_{1t} = 1|\mathbf{y})$; bottom panel: posterior probability of a low variance regime $\Pr(S_{2t} = 2|\mathbf{y})$.

TABLE 3 Parameter posterior mode and 90% HPD for US GDP

	α_1	α_2	ϕ_1	δ	V_e
Mode	-0.30	0.89	0.22	0.16	0.97
90% HPD	(-0.99,-0.07)	(0.59,0.99)	(-0.11,0.34)	(0.07,0.24)	(0.64,1.25)
	$\pi_{1\,11}$	$\pi_{1\,12}$	$\pi_{2\,11}$	$\pi_{2\,12}$	
Mode	0.74	0.03	0.98	0.02	
90% HPD	(0.16,0.83)	(0.00,0.07)	(0.84,0.99)	(0.00,0.08)	

6 Remarks

Sometimes it is useful to set parameters to constant values. One simple way to do so is to equalize the lower and upper bounds in the `hyptheta` argument of the `prior` namelist, for instance the instruction `hyptheta(1,1) = 0 1 .2 .2` fixes the parameter to .2. This also serves at setting parameter values when the data are to be generated by the program.

Parameterization is one important aspect of the prior elicitation. For instance complex roots in the AR(2) polynomial $\phi(L) = 1 - \phi_1 L - \phi_2 L^2$ may be imposed by writing $\phi_1 = 2A\cos 2\pi/\tau$ and $\phi_2 = -A^2$, where A and τ represent the cycle amplitude and period (see Planas et al., 2008). Also, stationarity of AR(p) processes may be imposed by expressing the AR coefficients in terms of partial correlations (see Barnett, Kohn, and Sheather, 1996). One important advantage of

writing a model-specific `design.for` file is that any parameterization of interest can be easily implemented.

Users may wish to elicitate a non-informative prior on a scale parameter V *à la* Jeffreys (1961); this can be approximately achieved by setting $\psi = \log V$ in the `design` subroutine and specifying a uniform distribution as $\psi \sim \texttt{BE}(1, 1)$ over a large set.

Several packages offer estimation of state space models; see the *Journal of Statistical Software* special issue 'Statistical Software for State Space Models' edited by Commandeur, Koopman, and Ooms (2011). Up to our knowledge only `S+FinMetrics` by Zivot and Wang (2006) considers classical analysis of Markov switching state space models. Estimation is performed by maximizing the approximated likelihood devised in Kim (1994).

7 Conclusion

DMM is a program for Bayesian analysis of dynamic mixture models. It handles multivariate series that may be non-stationary, with missing observations, and linked to some exogenous variables. The program implements up-to-date techniques for sampling the discrete latent variable in O(`T`) operations (GCK, 2000, Fiorentini et al., 2012), for exact initialization of the Kalman recursions (Koopman, 1997), for drawing model parameters efficiently via a slice sampler (Neal, 2003), and for computing the marginal likelihood. The prior distributions do not need to be conjugate.

Besides an ASCII file that contains the system dimensions, the prior information, and the data set, users are asked to write an extremely simple Fortran program that gives the state space representation. Although this may be seen an extra cost, such an architecture makes possible fitting any model that belongs to the wide class of conditionally Gaussian state space systems. It also gives complete freedom for the model parameterization. The program benefits from the computational speed advantage of low-level languages, which is particularly relevant when MCMC algorithms are employed.

References

Barnett G., Kohn R. and Sheater S. (1996), "Bayesian estimation of autoregressive model using Monte chain Monte Carlo", *Journal of Econometrics*, 74, 237-254.

Bauwens L., Lubrano M. and Richard J. (1999), *Bayesian Inference in Dynamic Econometric Models*, Oxford University Press.

Bell W.R. (2011), 'REGCMPNT - A Fortran program for regression models with ARIMA component errors', *Journal of Statistical Software*, 41, 7, 1-23.

Chib S. (1996), "Calculating posterior distributions and modal estimates in Markov mixture models", *Journal of Econometrics*, 75, 79-97.

Cobb G.W. (1978), 'The problem of the Nile: conditional solutions to a changepoint problem', *Biometrika*, 65, 2, 243-251.

Commandeur J., Koopman S., Ooms M. (2011), "Statistical Software for State Space Methods", *Journal of Statistical Software*, 41, 1, 1-18.

Durbin J. and Koopman S. (2002), 'A simple and efficient simulation smoother for state space time series analysis', *Biometrika*, 89, 603-615.

Fiorentini G., Planas C, and Rossi A. (2012), 'Efficient MCMC sampling in dynamic mixture models', *Statistics and Computing*, forthcoming.

Fruhwirth-Schnatter S. (2006), *Finite Mixture and Markov Switching Models*, New York, Springer.

Gelfand A. and Dey D. (1994), "Bayesian model choice: asymptotics and exact calculations", *Journal of the Royal Statistical Society, B*, 56, 501-514.

Gerlach R., Carter C., and Kohn, R. (2000), "Efficient Bayesian inference for dynamic mixture models", *Journal of the American Statistical Association*, 95, 819-828.

Geweke J. (1999), "Using simulation methods for Bayesian econometric models: inference, development, and communication", *Econometric Reviews*, 18, 1, 1-73.

Giordani P., Kohn R., and van Dijk D. (2007), "A unified approach to nonlinearity, structural change, and outliers", *Journal of Econometrics*, 137, 112-133.

Hamilton J.D. (1989), "A new approach to the economic analysis of nonstationary time series and the business cycle", *Econometrica*, 57, 357-384.

Harvey A.C. (1989), *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge University Press: Cambridge.

Jeffreys H. (1961), *The Theory of Probability* (*3rd edn*), Clarendon Press, Oxford.

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

Kim C.-J. (1994), "Dynamic linear models with Markov-switching", *Journal of Econometrics*, 60, 1-22.

Kim C.-J. and Nelson C.R. (1999), *State-space Models with Regime Switching: Classical and Gibbs Sampling Approaches with Applications*, Massachussets Institute of Technology Press, Cambridge.

Kitagawa G., (1977), 'An algorithm for solving the matrix equation X = F X F' + S', *International Journal of Control*, 25, 5, 745753.

Koopman S. (1997), "Exact initial Kalman filtering and smoothing for nonstationary time series

model", *Journal of American Statistical Association*, 92, 440, 1630-1638.

McConnell M.M. and Perez-Quiros G. (2000), "Output fluctuations in the United States: what has changed since the early 1980's", *American Economic Review*, 90, 5, 1464-1476.

Mendelssohn R. (2011), 'The STAMP Software for State Space Models', *Journal of Statistical Software*, 41, 2, 1-19.

Meng X.L. and Wong W.H. (1996), "Simulating ratios of normalizing constants via a simple identity: a theoretical exploration", *Statistica Sinica*, 6, 831-860.

Neal R.M., (2003), 'Slice sampling', *Annals of Statistics*, 31, 3, 705767.

Planas C., Rossi A. and Fiorentini G. (2008), "Bayesian analysis of the output gap", *Journal of Business & Economic Statistics*, 26, 1, 18-32.

Scott S.L. (2002), "Bayesian methods for hidden Markov models: recursive computing in the 21st century", *Journal of the American Statistical Association*, 97, 457, 337-351.

West M. and Harrison J. (1997), *Bayesian Forecasting and Dynamic Models*, Springer-Verlag, New York.

Zivot E. and Wang J. (2006), *Modelling financial time series with S-PLUS*, Springer-Verlag, New York.