Massively Parallel Sequential Monte Carlo for Bayesian Inference

Garland Durham* and John Geweke†‡

December 7, 2011

Abstract

This paper reconsiders sequential Monte Carlo approaches to Bayesian inference in the light of massively parallel desktop computing capabilities now well within the reach of individual academics. It first develops an algorithm that is well suited to parallel computing in general and for which convergence results have been established in the sequential Monte Carlo literature but that tends to require manual tuning in practical application. It then introduces endogenous adaptations in the algorithm that obviate the need for tuning, using a new approach based on the structure of parallel computing to show that convergence properties are preserved and to provide reliable assessment of simulation error in the approximation of posterior moments. The algorithm is generic, requiring only code for simulation from the prior distribution and evaluation of the prior and data densities, thereby shortening development cycles for new models. Through its use of data point tempering it is robust to irregular posteriors, including multimodal distributions. The sequential structure of the algorithm leads to reliable and generic computation of marginal likelihood as a by-product. The paper includes three detailed examples taken from state-of-the-art substantive research applications. These examples illustrate the many desirable properties of the algorithm.

Keywords: graphics processing unit; particle filter; posterior simulation; single instruction multiple data

*Leeds School of Business, University of Colorado (USA); Garland.Durham@colorado.edu

†University of Technology Sydney (Australia), Erasmus University (The Netherlands) and University of Colorado (USA), John.Geweke@uts.edu.au. Geweke acknowledges partial financial support from Australian Research Council grant DP110104372.

‡We acknowledge useful comments from Nicolas Chopin and William C. McCausland, discussions with Ron Gallant, and tutorials in CUDA programming from Rob Richmond. We bear sole responsibility for the content of the paper.

1 Introduction

Bayesian approaches have inherent advantages in solving inference and decision problems, but practical applications pose challenges for computation. As these challenges have been met Bayesian approaches have proliferated and contributed to the solution of applied problems. McGrayne (2011) has recently conveyed these facts to a wide audience.

Until the 1970's computation was limited to essentially deterministic closed-form solutions adapted either to quite specific applications or to statistical models generally applicable but rather simple. Zellner (1971) concisely represents the state of the art at that juncture. Posterior simulation evolved shortly thereafter. For example, the importance sampling methods pioneered by Kloek and van Dijk (1978) with the theory articulated in Geweke (1989) found widespread application in economics and business. Beginning with the revolutionary work of Gelfand and Smith (1990) Markov chain Monte Carlo (MCMC) greatly expanded the application of Bayesian inference, for example by making practical the innovations of Lindley and Smith (1972) in hierarchical models.

The evolution of Bayesian computation over the past half-century has conformed with exponential increases in speed and decreases in the cost of computing. The influence on algorithms, models, and the way that substantive problems are formulated for statistical inference can be subtle but is hard to over-state. Successful, innovative basic and applied research recognizes the characteristics of the tools of implementation from the outset and tailors approaches to those tools.

A number of Bayesian statisticians, ourselves included, find that we are now on the cusp of a major change in the computing environment. We anticipate that this change will strongly affect the evolution of Bayesian methodology and applications, and will improve models and decision-making.

1.1 Background

The methods developed here are motivated by recent innovations in computer hardware and software that bring massively parallel desktop computing within the reach of the academic research sector. The innovation in hardware consists of a *device* with many *cores* (processing units) connected to a *host* (conventional desktop or laptop computer with one or a few central processing units) through a *bus* (for data transfer between host and device). The device used for the applications in this paper is a graphics processing unit (GPU), which provides hundreds of cores at a cost well under one dollar (US) each. Multiple devices can be attached to a single host, making possible a desktop system with literally thousands of cores available at a cost within the budget of the typical academic researcher. Accompanying innovations in software efficiently manage host-device data transfers and device instruction code in ways that are largely transparent to the programmer. The software used in the applications in this paper is the CUDA extension of the C programming language (Nvidia, 2011), which is freely available. Section 2.1 provides more detail on relevant aspects of hardware and software (see also Hendeby et

al., 2010; Lee et al., 2010; Suchard et al., 2010).

The device is designed for *single instruction multiple data* (SIMD) processing. In its purest form this means that exactly the same machine instructions are executed in lockstep on the device processors, but the data on which the instructions operate differ from one device core to another. Some algorithms, for example linear algebra, are well suited to this restriction. Algorithms that are sequential or incorporate data dependent branching, like acceptance sampling and Gibbs sampling, are not well suited and can lead to bottlenecks that preclude any significant gains from parallel computing (Amdahl, 1967).

The sophistication of current hardware and software, including GPU's and the CUDA extension, significantly reduces the cost of some bottlenecks. Nevertheless, in order to realize the potential gains in computing speed on the order of several orders of magnitude made possible by the massively parallel environment, algorithms that conform to SIMD constraints in the bulk of their computations are needed. We refer to such algorithms and code as *SIMD-compatible*. The algorithm presented in this paper conforms to this standard and realizes the attendant increases in computing speed.

1.2 Approach

The approach taken here draws on a vibrant literature on sequential Monte Carlo methods for filtering and smoothing in state space models, adapting those methods to generic Bayesian inference in ways that are well suited to desktop parallel computing. It incorporates sound theoretical foundations, numerical efficiency for a wide variety of models and data, and requirements for implementation suited to applied scientists who do not specialize in computational approaches to Bayesian inference. The algorithm explored in this paper builds on existing work, and we provide evidence and specific recommendations regarding aspects that we have found to be particularly effective in the very demanding applications we provide as illustrations.

When implemented for applications at a level of complexity comparable to those presented in this paper, the algorithm is highly computationally intensive. It is the algorithm's amenability to implementation in the massively parallel CUDA environment that makes it particularly appealing.

The algorithm starts with a sample of parameter vectors, called particles, drawn from the prior distribution. It then introduces data incrementally, representing the updated posterior density by means of a sample of particles and importance weights. The essential computation in updating particles and weights is the evaluation of the likelihood function for each particle. Predictive likelihoods, and therefore marginal likelihood, are by-products of the importance weights. Expectations of other functions of interest are also easily obtained, and the applications in Sections 3–5 provide illustrations.

Each cycle of the algorithm moves through three successive phases: correction (C), selection (S), and mutation (M), that modify the particles. This C, S, M nomenclature is due to Chopin (2004), which is most similar to the algorithm developed here.

The C phase of each cycle successively adds observations to the information set.

At each observation predictive densities are evaluated and importance weights are updated. Evaluations of the predictive likelihood are SIMD-compatible and executed on the device (each particle corresponds to a thread which can be evaluated on a single core independently of all other particles). The updated weights provide a means to estimate the effective sample size (*ESS*; Kong et al., 1994; Liu and Chen, 1995) subsequent to each added observation. Transition to the S phase (CS transition) occurs when the *ESS* falls below a pre-specified threshold or when the data are exhausted, a criterion well-established in the literature (e.g., Del Moral et al., 2006; Chopin and Jacob, 2010). The *ESS* itself is computed on the device. Evaluating the decision rule requires minimal data transfer to the host and constitutes a trivial fraction of the total computational burden.

Low *ESS* indicates that the importance weights have become unbalanced with the addition of information into the posterior: a few particles have large weights while others have little weight. The S phase uses residual resampling (Baker, 1985, 1987; Liu and Chen, 1998) to extract an identically but not independently distributed sample of unweighted particles from the existing collection of particles. This phase is performed on the device and involves little computational effort relative to the balance of the algorithm.

After resampling, the particles are equally weighted, but there are multiple copies of particles that had large weights before resampling, and the number of distinct particles has been reduced (some particles with low weights disappear from the sample). The M phase addresses this attrition problem, which is ubiquitous with particle filters (Doucet et al., 2000; Fearnhead, 2002; Andrieu et al., 2010), by iterating over Metropolis steps with target density equal to the posterior density of the parameter vector at the current observation (Metropolis et al., 1953; Gilks and Berzuini, 2001; Chopin, 2002). The particles already represent a sample from the target distribution upon entry into the M phase; the goal here is diversification of the sample (mixing). Diagnostics indicating the degree to which this mixing has been successful are available at each iteration of the Metropolis step.

Each Metropolis iteration involves evaluating the posterior density kernel of candidate particles. This is where most of the computational cost of the algorithm occurs. However, this work is performed on the device, with each particle corresponding to a thread that can be evaluated independently (as in the C phase). The precise nature of the sampler used in the Metropolis steps is not important in principle (although it makes a great deal of difference in practice). Our approach uses a Gaussian random-walk sampler whose variance matrix is that of the existing collection of particles scaled by a step-size factor that is updated adaptively. This approach is simple, robust and generic, making it possible to use the algorithm in a wide variety of models with little effort. Generally speaking more efficient variants can be tailored to any given applications, but we conjecture that the approach introduced here will prove sufficient for most, as is the case for the applications in Sections 3 through 5.

Transition to the next cycle (MC transition) occurs at the end of the M phase if the sample is not yet exhausted. The particles at the end of the last cycle provide the final

representation of the posterior distribution conditional on all available information.

One of the innovations introduced in this paper is the classification of particles into several distinct groups which are operated on largely independently of each other (as detailed in Section 2.3). This strategy is natural in the massively parallel computing environment in which we are working, and has the important by-product of providing reliable numerical standard errors for posterior moments, log marginal likelihoods, and other quantities of interest as an inherent element of the procedure. Reporting information about the numerical errors associated with simulation-based statistical methods is important, yet such information can be costly to obtain and is often neglected.

There are several tuning parameters involved in the algorithm: number of groups, number of particles per group, number of Metropolis steps per M phase, and CS transition threshold. In application one executes the algorithm with fixed settings. If the numerical standard errors in the approximation of functions of interest are too large for the purposes at hand, one executes the algorithm again with revised settings. Furthermore, there are trade-offs involving some of these parameters. For example, one might try using fewer particles but mixing more thoroughly (more Metropolis iterations) in the M phase to achieve sufficient accuracy at lower computational cost. We provide some experimental results in the application sections and make recommendations regarding typical settings.

Section 2.3 presents a simplified version of this algorithm, with no adaptations dependent on the distribution of particles realized in a particular run of the algorithm, and uses results in Chopin (2004) to show that the distribution of the particles at the end of the last cycle satisfies a central limit theorem. It also shows how the particle grouping can be used to approximate the variance in this central limit theorem. A central difficulty in the literature is the paucity of limit theorems for adaptive sequential Monte Carlo. Section 2.4 exploits the structure of the parallel computing environment and a novel but simple argument that obviate these *lacunae* and place the adaptive algorithm on a sound theoretical footing.

1.3 Context and contribution

This paper pursues an agenda well established in the literature on sequential Monte Carlo methods for Bayesian inference. Its goal is to provide a generic approach to Bayesian inference with solid theoretical foundations that is practical and well suited to massively parallel desktop computing. It does this building on the work of many others. For some of the relevant lines of research in this field the contribution of this paper is incremental and for others it is discrete. We believe it is the first paper to tie these strands together in pursuit of this goal.

1. *The algorithm is generic.* The desirability of an algorithm that is reliable and ready to use, or "black box," has long been recognized in this literature, dating back to at least Liu and West (2001) and Chopin (2002) through very recent work like Fulop and Li (2011) and Chopin et al. (2011).

The algorithm presented here obviates the need for tedious tuning, blocking and monitoring that can be obstacles to applied Bayesian inference, and our implementation of the algorithm using the massively parallel CUDA framework greatly enhances computational tractability. These claims are supported by Section 2.4 and the examples in Sections 3, 4 and 5.

The algorithm has minimal requirements for its application.

(a) The prior distribution must be proper and there must be code that simulates from this distribution and evaluates the prior density. Given a proper prior distribution, writing code is typically trivial and there is little reason for it to be SIMD-compatible. The algorithm is incompatible with improper prior distributions, an irrelevant limitation from a subjective Bayesian perspective.

(b) There must be SIMD-compatible code that evaluates the conditional data density. This is a straightforward matter for likelihood-based inference that constitutes the great bulk of day-to-day application. This standard can also be met for models in which likelihood function evaluations are themselves subject to simulation error, as illustrated in the many papers taking up full Bayesian inference for nonlinear state space models (e.g., Liu and West, 2001; Carvalho et al., 2010) and by the example in Section 4 of this paper.

We believe that the approach explored in this paper will significantly shorten the development cycle for Bayesian inference in new models, and that it will prove more reliable and easier to implement than the optimization methods that are the foundation of non-Bayesian approaches and will thereby accelerate the adoption of Bayesian inference in applied statistics.

2. *There is a sound limiting distribution theory that can be used in a practical way to reliably assess the accuracy of sequential Monte Carlo approximations of posterior moments.* There is an established literature on limiting distributions (e.g., Gilks and Berzuini, 2001; Chopin, 2004; Künsch, 2005; Del Moral et al., 2006, 2011), and we draw specifically on Chopin (2004) in this work. This paper builds on this literature in two directions.

   (a) It is universally recognized and perhaps tautological that a generic algorithm must be adaptive. The transitions from the C to the S phase and from the S to the M phase described in the previous section are both adaptive. However, adaptation raises significant challenges for the derivation of central limit theorems. Chopin (2004) deals with resampling in the S phase and Del Moral et al. (2011) provides a central limit theorem for the case in which effective sample size drives the C to S phase transition, but to our knowledge these are the only such contributions. Moreover, the prospect that substantial theoretical work would be required for each variant on adaptation is daunting. This work introduces, in Section 2.4, an approach to formulating sound limiting

distribution theory that obviates this prospect and grows out of the parallel computing environment. We believe this approach is novel.

(b) Practical application of a central limit theorem requires consistent estimation of its variance term. This is straightforward for importance sampling (Geweke, 1989), but proved substantially more difficult for Markov chain Monte Carlo (e.g., Flegal and Jones, 2010). This question has been addressed only occasionally in the sequential Monte Carlo literature, e.g., Gilks and Berzuini (2001). Chopin (2004) discusses but does not implement the main idea used here, which is to enforce conditional independence across groups of particles in a way that an elementary central limit theorem and attendant variance approximation can be used to assess numerical accuracy. Section 2.3 gives the details and Sections 3 through 5 provide examples.

The paper supplements these contributions with carefully stated conditions sufficient for practical assessment of numerical accuracy based on the limiting distribution theory. These conditions are not necessary, but the set of cases in which the conditions we state are violated and yet these procedures are still reliable is not likely to be of much practical relevance, and users are more likely to check the simpler stated conditions.

We believe that sequential Monte Carlo methods are likely to grow in importance and application along with the penetration of massively parallel desktop computing, and it is important to avoid implementations that lead to users simply spinning numbers rather than computing reliable results.

3. *The algorithm is robust to irregular posterior distributions.* Sequential Monte Carlo methods inherently provide data point tempering (Chopin, 2002, 2004; Jasra et al., 2007b) through the sequential accumulation of information as the algorithm passes through the sample. Modifications of these algorithms lead to classical tempering that preserves this robustness (Jasra et al., 2007a; Duan and Fulop, 2011). While this is well established in the sequential Monte Carlo literature, it is less well recognized in the significant applied Bayesian community that uses Markov chain Monte Carlo and importance sampling methods almost exclusively. This community should be receptive to a generic algorithm that routinely handles irregularities like multimodality or weak identification. The applications in Sections 3 and 5 are chosen in part to illustrate the potential of sequential Monte Carlo in these dimensions.

4. *The algorithm provides marginal likelihoods and other indicators of model performance as routine byproducts.* Importance sampling produces marginal likelihood as a byproduct (Kloek and van Dijk, 1978; Geweke, 2005, Section 8.2.2) and the C phase inherits this property, as widely recognized in the sequential Monte Carlo literature (e.g., Del Moral et al., 2006). This is important for applied work because marginal likelihood computations are more awkward for Markov chain

Monte Carlo. Section 2.5 details how this is accomplished and shows that with modest supplementary computation predictive likelihoods and probability integral transform diagnostics can also be produced as byproducts of the algorithm. The examples in Sections 3 through 5 provide illustrations and show that the numerical accuracy is substantially lower than that generally reported for other purely computational approaches to these measures.

The relevant evaluation of these advantages resides in application to problems that are characteristic of the scale and complexity of serious disciplinary work. The balance of the paper provides three such applications, reflecting the backgrounds of the authors in economics and finance. The first application (Section 3) is a suite of extended exponential generalized autoregressive conditional heteroskedasticity (EGARCH) models (Nelson, 1991; Durham and Geweke, 2011) applied to over twenty years of daily financial asset returns, a sample of roughly 5,000 observations. Asset return models of this kind are widely used in the academic and financial sectors for assessing volatility and pricing derivative contracts like options. The distribution of conditional returns is described by a mixture of normals with up to four mixture components. Models include up to four EGARCH factors. The larger models exhibit strong multimodality and many parameters are weakly identified, features that can pose difficulties for conventional methods. Since it is the first in our series of examples, we use it to illustrate some of the aspects of the algorithm that are common to all applications. The second application (Section 4) is the dynamic multinomial probit model (Geweke et al., 1993, 1997; Rossi et al., 2005), using an artificial longitudinal data set for 500 individuals and 4 time periods. Models like this are used, for example, in understanding discrete choices of individuals over their lives. We include it here in part to show that the algorithm remains effective when likelihoods cannot be evaluated exactly but unbiased simulation approximations are available. The final application (Section 5) is the vector autoregression (VAR) model for predicting macroeconomic activity (Sims, 1980). This model is highly competitive with alternatives, and is widely applied by central banks and private forecasting firms. We include it here in part to explore the performance of the algorithm in a model with hundreds of parameters.

2 A generic algorithm for Bayesian inference

This section details the methodological contribution of this work. It begins (Section 2.1) with the features of the computing environment that guide the design of all aspects of the algorithm. That section also indicates the specific hardware and software environment of the applications that follow in Sections 3 through 5. Section 2.2 sets out much of the notation used in the balance of the section. It also enumerates a set of conditions that can be checked readily and are sufficient for the algorithm to reliably represent the posterior distribution as well as the accuracy of that representation. Section 2.3 provides a first version of the algorithm that is congruent with established limit theory

for sequential Monte Carlo but impractical for application. It lays the groundwork for claim 2b of Section 1.3. Section 2.4 modifies the algorithm in several ways that render it flexible and robust, supporting claim 1. There are imposing analytical obstacles to the application of limit theory that can be applied directly to this modified algorithm. Section 2.4 shows how this limitation can be circumvented so as to render established limit theory applicable in a massively parallel computing environment.

2.1 Computing environment

The particular device used in the applications in this paper is the graphics processing unit (GPU). As a practical matter several GPU's can be incorporated in a platform with no significant complications, and desktop computers that can accommodate up to eight GPU's (four cards with two GPU's each) are readily available. The single- and multiple-GPU environments are equivalent for our purposes. A single GPU consists of several multiprocessors, each with several cores. The GPU has global memory shared by its multiprocessors, typically one to several gigabytes (GB) in size, and local memory specific to each multiprocessor, typically on the order of 50 to 100 kilobytes (KB) per multiprocessor. (For example, this research uses a single Nvidia GTX 570 GPU with 15 multiprocessors, each with 32 cores. The GPU has 1.36 GB of local memory, and each multiprocessor has 49 KB of memory and 32 KB of registers shared by its cores.) The bus that transfers data between GPU global and local memory is significantly faster than the bus that transfers data between host and device, and accessing local memory on the multiprocessor is faster yet. Optimizing memory access patterns is critical for achieving good performance in this hardware environment.

This hardware has become attractive for scientific computing with the extension of scientific programming languages to allocate the execution of instructions between host and device and facilitate data transfer between them. Of these the most significant has been the computation unified device architecture (CUDA) extension of the C programming language (Nvidia, 2011). CUDA abstracts the host-device communication in a way that is convenient to the programmer yet faithful to the aspects of the hardware important for writing efficient code.

Code executed on the device is contained in special functions called kernels that are invoked by the host code. Specific CUDA instructions move the data on which the code operates from host to device memory and instruct the device to organize the execution of the code into a certain number of blocks with a certain number of threads each. The allocation into blocks and threads is the virtual analogue of the organization of a GPU into multiprocessors and cores.

While the most flexible way to develop applications that make use of GPU parallelization is through C/C++ code with direct calls to the vendor-supplied interface functions, it is also possible to work at a higher level of abstraction. For example, a growing number of mathematical libraries have been ported to GPU hardware (e.g., Innovative Computing Laboratory, 2011). Such libraries are easily called from standard scientific programming languages and can yield substantial increases in performance for

some applications. In addition, Matlab (2011) provides a library of kernels, provides interfaces for calling user-written kernels, and provides functions for host-device data transfer from within the Matlab workspace.

Both single-precision (four-byte) and double-precision (eight-byte) arithmetic are available, though double precision is several times slower than single (depending on the exact hardware used). The applications in this paper use a mix of single and double precision based on storage constraints and performance considerations.

2.2 Notation and conditions

We augment standard notation for data, parameters and models. The relevant observable random vectors are Y_t $(t = 1, \ldots, T)$ and $Y_{t_1:t_2}$ denotes the collection $\{Y_{t_1}, \ldots, Y_{t_2}\}$. The observation of Y_t is y_t, $y_{t_1:t_2}$ denotes the collection $\{y_{t_1}, \ldots, y_{t_2}\}$, and therefore $y_{1:T}$ denotes the data. This notation assumes ordered observations, which is natural for time series. If $\{Y_t\}$ is independent and identically distributed (i.i.d.) the ordering is arbitrary.

A model for Bayesian inference specifies a $k \times 1$ unobservable parameter vector $\theta \in \Theta$ and a conditional density

$$p\left(Y_{1:T} \mid \theta\right) = \prod_{t=1}^{T} p\left(Y_t \mid Y_{1:t-1}, \theta\right) \tag{1}$$

with respect to an appropriate measure for $Y_{1:T}$. The model also specifies a prior density $p\left(\theta\right)$ with respect to a measure ν. The posterior density $p\left(\theta \mid y_{1:T}\right)$ follows in the usual way from (1) and $p\left(\theta\right)$.

Central to the algorithm is a collection of particles θ_{jn} $(n = 1, \ldots, N; j = 1, \ldots, J)$, where J indicates the number of groups and N is the number of particles per group. The algorithm executes L cycles (indexed by ℓ). Cycle ℓ processes observations $y_{t_{\ell-1}+1}$ through y_{t_ℓ}, where $0 = t_0 < t_1 < \cdots < t_{L-1} < t_L = T$. Each cycle operates on the set of particles in three phases: C, S, and M. The C phase adds observations and updates importance weights. The S phase resamples. The M phase of cycle ℓ performs R_ℓ Metropolis iterations (indexed by r).

In the C phase of cycle ℓ, denote the particles $\theta_{jn}^{(\ell-1)}$. In the first cycle, $\ell = 1$, $\theta_{jn}^{(0)} \overset{iid}{\sim} p\left(\theta\right)$; in subsequent cycles, $\ell > 1$, the particles $\theta_{jn}^{(\ell-1)} \sim p\left(\theta \mid y_{1:t_{\ell-1}}\right)$ are identically but not independently distributed. When the C phase terminates, these are the particles that enter into the S phase of cycle ℓ. After execution of the S phase denote the selected particles $\theta_{jn}^{(\ell,0)}$. These are the particles that enter the M phase. Note that for each $\theta_{jn}^{(\ell,0)}$ there exists some n' such that $\theta_{jn}^{(\ell,0)} = \theta_{jn'}^{(\ell-1)}$. At the completion of iteration r of the Metropolis steps in the M phase of cycle ℓ denote the particles $\theta_{jn}^{(\ell,r)}$. The final step of this phase (step R_ℓ) defines $\theta_{jn}^{(\ell)} = \theta_{jn}^{(\ell,R_\ell)}$. If $\ell < L$ then $\theta_{jn}^{(\ell)}$ is carried to the start of the next cycle. If $\ell = L$ then the entire data set has been processed; in this case define $\theta_{jn} = \theta_{jn}^{(L)}$. The collection $\{\theta_{jn}\}$ represents the posterior distribution conditional on the

full sample. Occasionally reference is made to a generic moment or function of interest $g(\theta)$, and we then denote $g_{jn} = g(\theta_{jn})$, $g_{jn}^{(\ell)} = g\left(\theta_{jn}^{(\ell)}\right)$ and $g_{jn}^{(\ell,r)} = g\left(\theta_{jn}^{(\ell,r)}\right)$.

Several conditions come into play at various stages in the development of the algorithm and its application.

Condition 1 *(Prior distribution). The model specifies a proper prior distribution. The prior density kernel can be evaluated with SIMD-compatible code. Simulation from the prior distribution must be practical but need not be SIMD-compatible.*

It is well understood that a model must take a stance on the distribution of likely outcomes *a priori* if it is to have any standing in formal Bayesian model comparison. This requirement is functionally related to the successive predictive structure of many sequential Monte Carlo algorithms, including the one developed here, because these algorithms require a distribution for θ before the first observation is introduced. Given a proper prior distribution the evaluation and simulation conditions are weak.

Condition 2 *(Likelihood function evaluation) The sequence of conditional densities*

$$p(y_t \mid y_{1:t-1}, \theta) \quad (t = 1, \ldots, T) \tag{2}$$

can be evaluated with SIMD-compatible code. Alternatively, if this cannot be done, there exists a random variable $u \in \mathcal{U}$, jointly distributed with θ and $Y_{1:T}$, with the property

$$\int_{\mathcal{U}} p(Y_{1:t}, u, \theta)\, du = p(Y_{1:t}, \theta) \;\; \forall \;\; (\theta, Y_{1:t}) \quad (t = 1, \ldots, T). \tag{3}$$

Condition 2 is important to computational efficiency because evaluation of (2) constitutes almost all of the floating point operations in typical applications of the algorithm. In many cases the sequence $Y_{1:T}$ is completely observed and there are analytical expressions for (2) that can be evaluated with SIMD-compatible code. The alternative (3) introduces the possibility of simulated likelihood so long as the simulations are unbiased, a point also recognized by Chopin et al. (2011) and Fulop and Li (2011). Section 2.3 discusses the handling of simulated likelihood in the algorithm and Section 4 provides a worked example.

Condition 3 *(Bounded likelihood) The sequence of densities (2) is bounded above by $\overline{p} < \infty$ for all $\theta \in \Theta$.*

This is one of two sufficient conditions for the central limit theorem invoked in Section 2.3. It is not trivial, and in particular it is violated in mixture of normals models and other mixture models unless the support of the prior distribution is suitably truncated. On the other hand these conditions are sufficient, not necessary, and we have worked with cases in which Condition 3 is violated but the central limit theorem is still a reliable indicator of the accuracy of simulation approximations of posterior moments. The example in Section 3 is one such instance.

Condition 4 *(Existence of prior moments) If the algorithm is used to approximate* $\mathrm{E}\left[g\left(\theta\right) \mid y_{1:T}\right]$*, then* $\mathrm{E}\left[g\left(\theta\right)^{2+\delta}\right] < \infty$ *for some* $\delta > 0$*.*

In any careful implementation of posterior simulation the existence of certain posterior moments must be verified analytically. The condition $\mathrm{E}\left[g\left(\theta\right)^{2+\delta}\right] < \infty$, together with Condition 3, is sufficient for the existence of $\mathrm{E}\left[g\left(\theta\right) \mid y_{1:T}\right]$. Condition 4 also comes into play in establishing a central limit theorem.

Condition 5 *(Parameterization) The support* Θ *of* θ *is* $\Theta = \mathbb{R}^k$ *and* $\mathrm{var}\left(\theta\right) < \infty$*.*

This is a desirable but not an essential condition. Because the M phase employs a Metropolis random walk with a Gaussian proposal, it is likely to be more efficient if Condition 5 is satisfied than if it is not. In applications where the model under consideration does not satisfy Condition 5, we have found it useful to transform the parameter vector so that it conforms with this condition. The principle analytical demands in these cases are finding a parameterization with unbounded support but finite variance, and evaluating the Jacobian of the transformation. These are typically routine tasks.

2.3 Sequential Monte Carlo with predetermined sampling

This preliminary variant of the algorithm assumes predetermined values for some of the constants involved in adapting the algorithm to the particular data and model under consideration. In particular, L, $\{t_\ell,\, \ell = 1, \ldots, L\}$, $\{R_\ell,\, \ell = 1, \ldots, L\}$, and a doubly-indexed sequence of $k \times k$ variance matrices $\{\Sigma_{\ell r},\, \ell = 1, \ldots, L,\, r = 1, \ldots, R_\ell\}$ are all predetermined. That is, these constants are all determined independently of the simulation algorithm that generates the particles. This is the sense in which sampling is predetermined in this algorithm.

The following pseudo-code defines the algorithm. It employs the notation $\phi\left(x; \mu, \Sigma\right)$ for the multivariate $N\left(\mu, \Sigma\right)$ probability density function and $MH\left(p, q\right)$ for a Metropolis-Hastings step with proposal density q and target density p. The parameter vector θ should be transformed, if need be, so that it satisfies Condition 5.

$\theta_{jn}^{(0)} \overset{iid}{\sim} p\left(\theta\right)\ \ (n = 1, \ldots, N; j = 1, \ldots, J)\,;\ \ell = 0$

For $\ell = 1 : L$

Correction (C) phase, applied to all particles $\theta_{jn}\ \ (n = 1, \ldots, N; j = 1, \ldots, J)$

$w_{t_{\ell-1}}\left(\theta_{jn}^{(\ell-1)}\right) = 1$

For $s = t_{\ell-1} + 1 : t_\ell$

$$w_s\left(\theta_{jn}^{(\ell-1)}\right) = w_{s-1}\left(\theta_{jn}^{(\ell-1)}\right) \cdot p\left(y_s \mid y_{1:s-1}, \theta_{jn}^{(\ell-1)}\right) \tag{4}$$

End

$w\left(\theta_{jn}^{(\ell-1)}\right) := w_{t_\ell}\left(\theta_{jn}^{(\ell-1)}\right)$

Selection (S) phase, applied to each group $j = 1, \ldots, J$

Using residual sampling based on $\left\{w\left(\theta_{jn}^{(\ell-1)}\right) \quad (n = 1, \ldots, N)\right\}$, select

$$\left\{\theta_{jn}^{(\ell,0)} \quad (n = 1, \ldots, N)\right\} \text{ from } \left\{\theta_{jn}^{(\ell-1)} \quad (n = 1, \ldots, N)\right\}.$$

Mutation (M) phase, applied to all particles $\theta_{jn} \quad (n = 1, \ldots, N; j = 1, \ldots, J)$

For $r = 1 : R_\ell$

$$\theta_{jn}^{(\ell,r)} \sim MH\left(p\left(\theta|\, \mathbf{y}_{1:t_\ell}\right), \phi\left(\theta;\, \theta_{jn}^{(\ell,r-1)}, \Sigma_{\ell r}\right)\right) \tag{5}$$

End

$\theta_{jn}^{(\ell)} := \theta_{jn}^{(\ell,R_\ell)} \quad (n = 1, \ldots, N; j = 1, \ldots, J)$

End

$\theta_{jn} := \theta_{jn}^{(L)} \quad (n = 1, \ldots, N; j = 1, \ldots, J)$

Floating point operations are almost entirely concentrated at points (4) and (5) in the algorithm. From Condition 2 these computations are SIMD-compatible. As a consequence computational speed scales well according to the number of cores available.

The C phase operates on a group of observations rather than a single observation, an idea that dates at least to Chopin (2002), which terms this procedure "iterated batch importance sampling." The S phase has its origins in the sampling-importance-resampling algorithm of Rubin (1988). Note that resampling is carried out independently within each group rather than across all particles, an idea that may be traced to Chopin (2002).

This algorithm is close to being a special case of the algorithm in Theorem 2 of Chopin (2004). There are three points of difference.

1. In our notation Chopin's algorithm takes $L = T$ and $t_\ell = \ell$ ($\ell = 1, \ldots, L$). The formulation here amounts to scarcely more than a change in notation in the statement and proof of Chopin's Theorem 2.

2. The S phase of this algorithm is applied separately within each of J groups rather than to the entire set of particles. As discussed in Chopin (2004) the essential feature of the S phase is the independence of the random components of residual selection, and that is preserved here.

3. The final representation of the posterior distribution in Chopin (2004) consists of the particles $\theta_{jn}^{(L-1)}$ and weights $w\left(\theta_{jn}^{(L-1)}\right)$ at the end of the C phase of cycle L. The limiting normal distribution is not compromised by the final S and M phases of our algorithm.

From Theorem 2 of Chopin (2004), the Metropolis random walk algorithm in the M phase can be replaced with any MCMC algorithm with a unique invariant distribution whose p.d.f. is $p\left(\theta \mid y_{1:t_\ell}\right)$. The algorithm just described uses the Metropolis random walk because it is generic and has worked well for us in a wide variety of applications. We have found that when k is large (i.e., θ is of high dimension) then it may be advantageous to execute Metropolis steps sequentially over subsets of parameters. Section 5 provides an example.

In the case of unbiased likelihood simulation, the contingency addressed in Condition 2, let u index the sequence of random variables used to simulate the likelihood function, for example, the seed of the random number generator. This leads to no essential changes in the algorithm as recognized in the literature for the case of importance sampling (Shephard and Pitt, 1997; Koopman et al., 2009) and the Metropolis-Hastings algorithm (Andrieu et al., 2010; Flury and Shephard, 2008). Indeed, (3) implies

$$\int_{\mathcal{U}} p\left(\theta, u \mid Y_{1:t} = y_{1:t}\right) du = \frac{\int_{\mathcal{U}} p\left(y_{1:t}, u, \theta\right) du}{\int_{\Theta}\int_{\mathcal{U}} p\left(y_{1:t}, u, \theta\right) du d\theta} = \frac{p\left(y_{1:t}, \theta\right)}{p\left(y_{1:t}\right)} = p\left(\theta \mid y_{1:t}\right), \quad (6)$$

so the seed u of the random number generator is, functionally, the same as that of a particularly simple latent variable. Hence the range of posterior simulation methods available is undiminished when the likelihood function cannot be evaluated exactly but an unbiased simulator is available.

Operationally the particles θ_{jn} are each augmented with a seed for the random number generator $u_{jn}^{(\ell)}$, with $u_{jn}^{(0)}$ drawn from a uniform distribution at the start of the algorithm. For simplicity, suppose that the unit interval $(0, 1)$ represents the set of potential seeds. Like $\theta_{jn}^{(\ell)}$, the seed $u_{jn}^{(\ell)}$ associated with a given particle does not change during the C phase of the algorithm. The S phase is likewise unmodified from the pseudo-code presented above. In the M phase, each Metropolis step r consists of two sub-steps, the first operating on seeds

$$MH\left(p\left(u \mid \mathbf{y}_{1:t_\ell}, \theta_{jn}^{(\ell,r-1)}\right), I_{(0,1)}\left(u\right)\right) \quad (7)$$

and the second on parameters

$$MH\left(p\left(\theta \mid \mathbf{y}_{1:t_\ell}, u_{jn}^{(\ell,r)}\right), \phi\left(\theta; \theta_{jn}^{(\ell,r-1)}, \Sigma_{\ell r}\right)\right). \quad (8)$$

Thus, the seed associated with a particle may or may not change during the M phase. The acceptance rate can vary as the algorithm passes through the sample because the suitability of $U(0, 1)$ as a candidate distribution changes systematically. Section 4.2 elaborates on these technical points in the context of a particular example.

Conditions 3 and 4 imply the conditions of Chopin's Theorem 2. Hence for any function of interest g satisfying Condition 4 we may define

$$\overline{g} = \mathrm{E}\left[g\left(\theta\right) \mid y_{1:T}\right], \qquad \overline{g}^{(J,N)} = (JN)^{-1} \sum_{j=1}^{J} \sum_{n=1}^{N} g\left(\theta_{jn}\right),$$

and conclude

$$(JN)^{1/2} \left(\overline{g}^{(J,N)} - \overline{g}\right) \xrightarrow{d} N\left(0, v\right). \tag{9}$$

The three points of difference between this algorithm and Chopin's, just enumerated, all imply that v in (9) does not share the analytical expression of Chopin (2004) for the variance parameter of the central limit theorem. We do not have a similar analytic result for v in (9), but these expressions provide little foundation for numerical approximation of v in any event.

The algorithm organizes particles into groups in order to address this issue. Consider the $N \times 1$ vectors θ_j^* $(j = 1, \ldots, J)$, each comprised of the N particles in group j, in any phase or cycle of the algorithm. The J vectors $\{\theta_1^*, \ldots, \theta_J^*\}$ are mutually independent in every phase of every cycle, and in particular this is so at the conclusion of the algorithm. Define the within-group sample means

$$\overline{g}_j^N = N^{-1} \sum_{n=1}^{N} g\left(\theta_{jn}\right) \qquad (j = 1, \ldots, J). \tag{10}$$

The same central limit theorem applies to each $\overline{g}_j^N$ as to $\overline{g}^{(J,N)}$, but with variance Jv. Hence a reasonable estimate of v is

$$\widehat{v}^{(J,N)} = N \sum_{j=1}^{J} \left[\overline{g}_j^N - \overline{g}^{(J,N)}\right]^2 / \left(J - 1\right) \tag{11}$$

and more precisely, as $N \to \infty$,

$$\left(J - 1\right) \widehat{v}^{(J,N)} / v \xrightarrow{d} \chi^2\left(J - 1\right). \tag{12}$$

Thus $\mathrm{var}\left(\overline{g}^{(J,N)}\right) = \mathrm{E}\left[\overline{g}^{(J,N)} - \overline{g}\right]^2 \cong \widehat{v}^{(J,N)}/(JN)$. The *numerical standard error* (*NSE*) of $\overline{g}^{(J,N)}$ is $\left[\widehat{v}^{(J,N)}/(JN)\right]^{1/2}$. The numerical approximation of $\mathrm{var}\left[g\left(\theta\right) \mid y_{1:T}\right]$ is

$$(JN)^{-1} \sum_{j=1}^{J} \sum_{n=1}^{N} \left[g\left(\theta_{jn}\right) - \overline{g}^{(J,N)}\right]^2.$$

Therefore the *relative numerical efficiency* (*RNE*; Geweke, 1989) of $\overline{g}^{(J,N)}$ is

$$\sum_{j=1}^{J} \sum_{n=1}^{N} \left[g\left(\theta_{jn}\right) - \overline{g}^{(J,N)}\right]^2 / \widehat{v}^{(J,N)}.$$

While the approximation $\overline{g}^{(J,N)}$ of $\overline{g}$ is simulation (JN) consistent, from (12) $\widehat{v}^{(J,N)}$ is not a consistent estimator of v. We do not think this is of any practical concern. The assessment of numerical accuracy in this algorithm, based on independent replications, is based on the Lindberg-Levy central limit theorem. Consistent with our experience this suggests that the assessment of numerical accuracy (11) is substantially more reliable than similar approximations in either importance sampling or single-chain MCMC.

2.4 Sequential Monte Carlo with adaptive sampling

The sequential Monte Carlo algorithm just described does not work, as a practical matter. While a given set of the constants and sequences indicated in the first two paragraphs of the previous section might work well for one particular model and data set, they cannot work well for all (or even very many). The need to adapt sequential Monte Carlo algorithms to the characteristics of each application is well recognized in the literature. Many established particle filtering algorithms, for example, are comprised of a C phase and an S phase, monitoring the particles as t advances to determine whether to proceed with C or a CS transition. This transition is often couched in terms of an increasing particle attenuation problem and the need for renewal, and the transition is usually based on simple properties of the particles like the ESS criterion of Liu and Chen (1995).

Adaptations of this kind complicate the convergence theory. A series of papers, including Chopin (2002, 2004), Künsch (2005), Douc and Moulines (2008) and Andrieu et al. (2010), have wrestled with the loss of independence induced by adaptive transitions in various settings. Chopin (2004) introduces the M phase but at the cost of deterministic CS transitions, which in practical terms means that renewal typically takes place well before it is warranted by particle attrition. There is an even richer variety of adaptations that can take place in the M phase, as indicated for the algorithm summarized in Section 1.2. It is well recognized that such adaptations are critical to the efficiency of sequential Monte Carlo.

There is a simple and straightforward way to bridge this gap between theory and practice that, to our knowledge, has not been exploited in the literature. First, modify the algorithm presented in Section 2.3 so that some or all of the constants and sequences are adaptive, with values chosen based on the distribution of particles at the time each element of a sequence comes into play. (That is what the algorithm summarized in Section 1.2 does.) Record all of the constants and sequences chosen in this way. Then, repeat the exercise (with different seeds for the random number generator) using exactly the algorithm described in Section 2.3. In this repetition the constants and sequences are predetermined, precisely as required. While the adaptive algorithm is not theoretically defensible, the repetition with fixed constants and sequences is. Moreover, given the large values of N that are feasible in the context of massively parallel desktop computing, the outcome from the repetition is likely to be of practical value—that is, numerical standard errors will be sufficiently small for the purposes at hand.

Of course, given the parallel computing environment this exercise need not be carried

out sequentially: half the blocks could be adaptive, with the other half utilizing the elements of the sequences $\{t_\ell\}$, $\{R_\ell\}$, and $\{\Sigma_{\ell r}\}$ as they become available. This preserves the SIMD-compatibility at points (4) and (5) and leads to negligible reduction in efficiency of the entire algorithm. At the end, one could confine attention to the outcome from the non-adaptive blocks. One can also compare the results from the adaptive and non-adaptive blocks. We have carried out this exercise for a number of applications, including those presented in Sections 3 through 5. The results are not only similar, but in many cases a test of the null hypothesis of identical results in the two sets of blocks fails to be rejected at conventional significance levels. Less formally, differences of approximations of posterior moments are within the range implied by numerical standard errors—and these are small, given the number of particles now feasible with desktop parallel computing.

This approach to the extension of relatively rigid sequential Monte Carlo algorithms, for which limiting distribution theory has been established, to adaptive versions that are essential to a generic approach but for which the theory would be more challenging to establish directly, is itself generic. In particular it applies not only to the specific adaptive scheme outlined in the rest of this section, but also to variants on this scheme including improvements that should emerge in future research. Likewise, it need not begin with Chopin (2004) but could just as well build on other limit theory like that in Del Moral et al. (2011). This is the essence of claim 2a made in Section 1.3.

We have experimented with many possible adaptations for the determination of L and the sequences $\{t_\ell\}$, $\{R_\ell\}$ and $\{\Sigma_{\ell r}\}$. These experiments have been conducted in the context of scaled-up applications typical of substantive research, like those in Sections 3 through 5. In comparing different adaptations we find that those that are better or worse tend to be uniformly so across these applications. We have found the following to be effective.

1. Following each computation of (4) in the C phase, determine the effective sample size
$$ESS = \frac{\left[\sum_{j=1}^{J}\sum_{n=1}^{N} w_s\left(\theta_{jn}^{(\ell-1)}\right)\right]^2}{\sum_{j=1}^{J}\sum_{n=1}^{N} w_s\left(\theta_{jn}^{(\ell-1)}\right)^2}$$
and proceed to the CS transition if $ESS/(JN) < D_1$, where $D_1 \in [0,1]$ is prespecified and fixed. This transition in each cycle determines the sequence $\{t_\ell\}$ and the number of cycles L. We have found that any choice $D_1 \in [0.1, 0.9]$ makes little difference. Larger (smaller) values of D_1 increase (decrease) L but reduce (increase) the number of Metropolis steps required to achieve a given level of numerical accuracy. Section 3.2 illustrates this trade-off.

2. In the S phase, residual resampling (Baker, 1985, 1987; Liu and Chen, 1998) and systematic resampling (Whitley, 1994; Carpenter, et al., 1999) are equally efficient and are superior to multinomial resampling (Gordon et al., 1993). Since limit

theory for systematic resampling is problematic (Chopin, 2004) we use residual resampling.

3. In the M phase, for some fixed $R > 0$ and $D_2 \in [0, D_1)$ execute $R_\ell = R$ Metropolis random walk steps if $ESS/(JN) > D_2$ at the preceding CS transition, and execute $R_\ell = 3R$ steps otherwise. The efficiency of the algorithm is not particularly sensitive to either the choice of D_2 or the number of additional Metropolis steps taken if this criterion is triggered. For the applications in this paper, this is a factor only in the VAR model, as described in Section 5.2.

4. Immediately preceding each Metropolis step r, compute the sample variance of the current collection of particles $\{\theta_{jn}^{(\ell,r)}\}$ and let $\Sigma_{\ell r}$ be equal to this matrix times a scaling factor, $h_{\ell r}$. The scaling factor is initialized at $h_{11} = 0.5$. After the Metropolis step is completed, compute the acceptance rate across all particles in all groups. If this exceeds 0.25, increase the scaling factor by 0.01, and otherwise reduce it by 0.01, unless this change would cause the scaling factor to be less than 0.10 or exceed 1.0. In the applications in Sections 3 through 5 these barriers are rarely operational.

The adaptive algorithm requires that the constants N, J, D_1, D_2 and R be specified. For the applications we always use $J = 16$ groups of $N = 2^{12}$ particles each. As documented in Sections 3 through 5 we find that the choices $D_1 = 0.5$ and $D_2 = 0.2$ work well across applications, whereas the accuracy of the simulation approximation is sensitive to the choice of R.

2.5 Model evaluation and comparison

The algorithm provides the marginal likelihood (ML) of the model directly. This stems from the expression

$$ML = \prod_{\ell=1}^{L} p\left(y_{t_{\ell-1}+1:t_\ell} \mid y_{1:t_{\ell-1}}\right)$$

(Geweke, 2005, Section 2.6.2). At the conclusion of the C phase in cycle ℓ, $w\left(\theta_{jn}^{(\ell-1)}\right) = p\left(y_{t_{\ell-1}+1:t_\ell} \mid y_{1:t_{\ell-1}}, \theta_{jn}^{(\ell-1)}\right)$, $\mathrm{E}\left[w\left(\theta_{jn}^{(\ell-1)}\right)\right] = p\left(y_{t_{\ell-1}+1:t_\ell} \mid y_{1:t_{\ell-1}}\right)$, and

$$N^{-1}\sum_{n=1}^{N} w\left(\theta_{jn}^{(\ell-1)}\right) \xrightarrow{as} \int_\Theta p\left(y_{t_{\ell-1}+1:t_\ell} \mid y_{1:t_{\ell-1}}, \theta\right) d\nu\left(\theta\right) = p\left(y_{t_{\ell-1}+1:t_\ell} \mid y_{1:t_{\ell-1}}\right).$$

A simulation-unbiased and simulation-consistent approximation of ML is

$$ML^{(J,N)} = J^{-1}\sum_{j=1}^{J} ML_j^N, \text{ where } ML_j^N = \prod_{\ell=1}^{L}\left[N^{-1}\sum_{n=1}^{N} w\left(\theta_{jn}^{(\ell-1)}\right)\right].$$

Then following the procedure of Section 2.3, the numerical standard error associated with this approximation is $\left(\widehat{v}^{(J,N)}\right)^{1/2}$ with

$$\widehat{v}^{(J,N)} = \sum_{j=1}^{J} \left(ML_j^N - ML^{(J,N)}\right)^2 / (J-1).$$

The alternative approximation

$$ML^{*N} = \prod_{\ell=1}^{L} \left[(JN)^{-1} \sum_{j=1}^{J} \sum_{n=1}^{N} w\left(\theta_{jn}^{(\ell-1)}\right) \right]$$

is also simulation-unbiased and simulation-consistent and has the same asymptotic variance. If one were to report marginal likelihood there would be little to choose between ML^N and ML^{*N}, and numerical standard error would be $\left(\widehat{v}^{(J,N)}/(JN)\right)^{1/2}$ in either case.

It is much more common to report log marginal likelihoods, and these reports are sometimes used in subsequent model comparison. By Jensen's inequality the approximation $\log ML^N$ or $\log ML^{*N}$ of $\log ML$ incurs negative simulation bias. A standard Taylor series expansion shows that the bias in $\log\left(ML^{*N}\right)$ is approximately $-\widehat{v}^{(J,N)}/\left[2JN\left(ML\right)^2\right]$ and that the numerical variance (due to Monte Carlo simulation) is approximately $\widehat{v}^{(J,N)}/\left[JN\left(ML\right)^2\right]$. The latter variance can be estimated from $\log\left(ML_j^N\right)$ $(j = 1, \ldots, J)$ in the same way as was described in Section 2.3 for any function of interest. The bias adjustment for $\log\left(ML_N^*\right)$ is then simply to add half its numerical variance.

A simple side computation at this point in the algorithm provides simulation-consistent approximations to the sequence of predictive likelihoods.

$$\frac{\sum_{j=1}^{J}\sum_{n=1}^{N} w_s\left(\theta_{jn}^{(\ell-1)}\right)}{\sum_{j=1}^{J}\sum_{n=1}^{N} w_{s-1}\left(\theta_{jn}^{(\ell-1)}\right)} = \frac{\sum_{j=1}^{J}\sum_{n=1}^{N} w_{s-1}\left(\theta_{jn}^{(\ell-1)}\right) p\left(y_s \mid y_{1:s-1}, \theta_{jn}^{(\ell-1)}\right)}{\sum_{j=1}^{J}\sum_{n=1}^{N} w_{s-1}\left(\theta_{jn}^{(\ell-1)}\right)} \xrightarrow{as} p(y_s \mid y_{1:s-1}). \tag{13}$$

These are central to the optimal pooling of models to maximize the log score of prediction (Geweke and Amisano, 2011; Durham and Geweke, 2011). Numerical standard errors for these quantities, and numerical standard errors and bias adjustment for their logarithms, follow in exactly the same way as for marginal likelihood and log marginal likelihood.

The C phase of the algorithm, in the computation (4), represents each posterior distribution $p\left(\theta \mid y_{1:s}\right)$ $(s = 1, \ldots, T)$ by means of an importance sample. If at this point one executes the auxiliary simulations $Y_{sjn}^{(\ell-1)} \sim p\left(Y_s \mid y_{1:s-1}, \theta_{jn}^{(\ell-1)}\right)$ $(n = 1, \ldots, N; j = 1, \ldots, J)$ then as simulation-consistent approximation of the cumulative distribution of a function of interest $f\left(Y_s\right)$, evaluated at the observed value $f\left(y_s\right)$, is

$$\frac{\sum_{j=1}^{J}\sum_{n=1}^{N} w_s\left(\theta_{jn}^{(\ell-1)}\right) I_{(-\infty, f(y_s)]}\left(f\left(Y_{sjn}^{(\ell-1)}\right)\right)}{\sum_{j=1}^{J}\sum_{n=1}^{N} w_s\left(\theta_{jn}^{(\ell-1)}\right)}.$$

These evaluations are the essential element of a probability integral transform test of model specification (Rosenblatt, 1952; Smith, 1985; Diebold et al., 1998; Berkowitz, 2001; Geweke and Amisano, 2010).

3 Example: Exponential generalized autoregressive conditional heteroskedasticity model

The predictive distributions of returns to financial assets are central to the pricing of their derivatives like futures contracts and options. The literature modeling asset return sequences as stochastic processes is enormous and has been a focus and motivation for Bayesian modelling in general and application of sequential Monte Carlo methods in particular. One of these models is the exponential generalized autoregressive conditional heteroskedasticity (EGARCH) model introduced by Nelson (1991). The example in this section works with a family of extensions developed in Durham and Geweke (2011) that is highly competitive with many stochastic volatility models.

In the context of this paper the EGARCH model is also of interest because its likelihood function is relatively intractable. The volatility in the model is the sum of several factors that are exchangeable in the posterior distribution. The return innovation is a mixture of normal distributions that are also exchangeable in the posterior distribution. Both features are essential to the superior performance of the model (Durham and Geweke, 2011). Permutations in the ordering of factors and mixture components induce multimodal distributions. Models with these characteristics have been widely used as a drilling ground to assess the performance of simulation approaches to Bayesian inference with ill-conditioned posterior distributions (e.g., Jasra et al., 2007b). The models studied in this section have up to $(4!)^2 = 576$ permutations, and potentially as many local modes. We make no efforts to optimize the algorithm for these characteristics. Nonetheless, the irregularity of the posteriors turns out to pose no difficulties for the algorithm, illustrating the robustness that derives from its implicit data point tempering.

Most important, in our view, this example illustrates the potential large savings in development time and intellectual energy afforded by the algorithm presented in this paper, compared with other approaches that might be taken. We believe that other existing approaches, including importance sampling and conventional variants on Markov chain Monte Carlo, would be substantially more difficult. At the very least they would require experimentation with tuning parameters by Bayesian statisticians with particular skills in these numerical methods, even after using the approach of Geweke (2007) to deal with dimensions of posterior intractability driven by exchangeable parameters in the posterior distribution. The algorithm replaces this effort with the learning embedded in data point tempering, thereby releasing the time of investigators for more productive and substantive efforts.

Table 1: Parameters and prior distributions for the EGARCH models

All parameters have Gaussian priors with means and standard deviations indicated below (the prior distribution of θ_{8i} is truncated below at -3.0). Indices i and k take on the values $i = 1, \ldots, I$ and $k = 1, \ldots, K$.

	Mean	Std Dev	Transformation
θ_1	0	1	$\mu_Y = \theta_1/1000$
θ_2	$\log(0.01)$	1	$\sigma_Y = \exp(\theta_2)$
θ_{3k}	$\tanh^{-1}(0.95)$	1	$\alpha_k = \tanh(\theta_{3k})$
θ_{4k}	$\log(0.10)$	1	$\beta_k = \exp(\theta_{4k})$
θ_{5k}	0	0.2	$\gamma_k = \theta_{5k}$
θ_{6i}	0	1	$p_i^* = \tanh(\theta_{6i}) + 1$
θ_{7i}	0	1	$\mu_i^* = \theta_{7i}$
θ_{8i}	0	1	$\sigma_i^* = \exp(\theta_{8i})$

3.1 Model and data

An EGARCH model for a sequence of asset returns $\{y_t\}$ has the form

$$v_{kt} = \alpha_k v_{k,t-1} + \beta_k \left(|\varepsilon_{t-1}| - (2/\pi)^{1/2} \right) + \gamma_k \varepsilon_{t-1} \quad (k = 1, \ldots, K), \tag{14}$$

$$y_t = \mu_Y + \sigma_Y \exp\left(\sum_{k=1}^{K} v_{kt}/2 \right) \varepsilon_t. \tag{15}$$

The return disturbance term ε_t is distributed as a mixture of I normal distributions,

$$p(\varepsilon_t) = \sum_{i=1}^{I} p_i \phi(\epsilon_t; \mu_i, \sigma_i^2) \tag{16}$$

where $\phi(\,\cdot\,; \mu, \sigma^2)$ is the Gaussian density with mean μ and variance σ^2, $p_i > 0$ $(i = 1, \ldots, I)$ and $\sum_{i=1}^{I} p_i = 1$. The parameters of the model are identified by the conditions $\mathrm{E}(\varepsilon_t) = 0$ and $\mathrm{var}(\varepsilon_t) = 1$; equivalently,

$$\sum_{i=1}^{I} p_i \mu_i = 0, \qquad \sum_{i=1}^{I} p_i (\mu_i^2 + \sigma_i^2) = 1. \tag{17}$$

The models are indexed by K, the number of volatility factors, and I, the number of components in the return disturbance normal mixture, and we refer to the specification (14)–(15) as `egarch_KI`. The original form of the EGARCH model (Nelson, 1991) is (14)–(15) with $I = K = 1$.

The algorithm operates on transformed parameters, as described in Condition 5 of Section 2.2. The vector of transformed parameters is denoted θ. The prior distributions for the elements of θ are all Gaussian. Priors and transformations are detailed in Table

1. The intermediate parameters p_i^*, μ_i^* and σ_i^* are employed to enforce the normalization (17) and undergo the further transformations

$$p_i = p_i^* / \sum_{i=1}^{I} p_i^*, \;\; \mu_i^{**} = \mu_i^* - \sum_{i=1}^{I} p_i \mu_i^*, \;\; c = \left[\sum_{i=1}^{I} p_i ((\mu_i^{**})^2 + (\sigma_i^*)^2) \right]^{-1/2}, \tag{18}$$

$$\mu_i = c\mu_i^{**}, \;\; \sigma_i = c\sigma_i^* \;\; (i = 1, \ldots, I). \tag{19}$$

The truncation of the parameters θ_{8i} $(i = 1, \ldots, I)$ bounds the likelihood function above, thus satisfying Condition 3. The initial simulation from the prior distribution is trivial, as is evaluation of the prior density.

Evaluation of $p(y_t \mid y_{1:t-1}, \theta)$ entails the following steps, which can readily be expressed in SIMD-compatible code, satisfying Condition 2.

1. Transform the parameter vector θ to the parameters of the model (14)–(15) using the fourth column of Table 1 and (18)–(19).

2. Compute v_{kt} $(k = 1, \ldots, K)$ using (14), noting that ε_{t-1} and $v_{k,t-1}$ $(k = 1, \ldots, K)$ are available from the evaluation of $p(y_{t-1} \mid y_{1:t-2}, \theta)$. As is conventional in these models the volatility states are initialized at $v_{k0} = 0$ $(k = 1, \ldots, K)$.

3. Compute $h_t = \sigma_Y \exp\left(\sum_{k=1}^{K} v_{kt}/2\right)$ and $\varepsilon_t = (y_t - \mu_Y)/h_t$.

4. Evaluate $p(y_t \mid y_{1:t-1}, \theta) = (2\pi)^{-1/2} h_t^{-1} \sum_{i=1}^{I} \left\{ p_i \frac{1}{\sigma_i} \exp\left[-(\varepsilon_t - \mu_i)^2 / 2\sigma_i^2 \right] \right\}$.

The observed returns are $y_t = \log(p_t/p_{t-1})$ $(t = 1, \ldots, T)$ where p_t is the closing Standard and Poors 500 index on trading day t. We use returns beginning January 3, 1990 $(t = 1)$ and ending March 31, 2010 $(t = T = 5100)$.

3.2 Performance

All of the inference for the EGARCH models is based on $2^{16} = 65,536$ particles in $J = 2^4 = 16$ groups of $N = 2^{12} = 4,096$ particles each. Except where specified otherwise, $R = 55$ (the base number of Metropolis iterations in each M phase), $D_1 = 0.50$ (*ESS* threshold for *CS* transition), and $D_2 = 0.20$ (the *ESS* threshold below which additional Metropolis steps are taken). The exercise begins by comparing the log marginal likelihood of the 10 variants of this model indicated in column 1 of Table 2. The format of this table is similar to that of several that follow. The number of cycles (L) varies from application to application because it is determined by the sequential Monte Carlo algorithm with adaptive sampling detailed in Section 2.4. The log marginal likelihood is bias adjusted and its numerical standard error (*NSE*) is computed as described in Section 2.5. "Log score" is the log predictive likelihood $p(y_{505:5100} \mid y_{1:504})$; observation 505 is the return on the first trading day of 1992. Log score is also bias adjusted.

Table 2: Comparison of EGARCH models

Model	Compute time (seconds)	Cycles L	Log Marginal Likelihood	Numerical Standard Error	Log Score	Numerical Standard Error
`egarch_11`	177	53	16,641.84	0.04	15,009.21	0.04
`egarch_12`	318	68	16,712.50	0.08	15,075.43	0.07
`egarch_21`	231	60	16,669.34	0.07	15.038.38	0.08
`egarch_22`	384	76	16,735.80	0.10	15,099.73	0.09
`egarch_23`	499	77	16,750.83	0.13	15,114.28	0.11
`egarch_32`	447	75	16,733.99	0.12	15,099.44	0.11
`egarch_33`	561	77	16,748.81	0.13	15,113.76	0.14
`egarch_34`	748	82	16,748.42	0.10	15,113.33	0.08
`egarch_43`	673	78	16,745.50	0.10	15,112.17	0.10
`egarch_44`	881	81	16,745.19	0.12	15,111.94	0.11
$J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$.						

Bayes factors strongly favor the `egarch_23` model, as do log scores. In comparison with all the models that it nests, the Bayes factor is at least $\exp(15)$. In comparison with all the models that nest `egarch_23`, the Bayes factor ranges from $\exp(2)$ to over $\exp(5)$. This pattern is classic: the data provide strong diagnostics of underfitting, while the evidence of overfitting is weaker because it is driven primarily by the prior distribution. The 95% confidence intervals for log Bayes factors are generally shorter than 0.3. As will be seen, the length of these confidence intervals can be reduced by increasing R.

Compute time increases substantially as a function of I and K. This is mainly due to the greater number of floating point operations required to evaluate likelihood functions in the M phase of the algorithm. To a lesser extent it is caused by faster reduction of effective sample size in the C phase, increasing the number of cycles L and thereby the number of passes through the M phase. Going forward, all examples in this section utilize the `egarch_23` model.

Table 3 compares compute time and *NSE* for log marginal likelihood and log score for some alternative choices of R. The M phase accounts for most of the compute time when R is large, over 90% for $R > 50$. The number of cycles is little affected by the size of R.

Reported values for *NSE* should be regarded as rough approximations when the number of groups J is as small as it is in the examples in this paper ($J = 16$). From one execution of the algorithm to another (with identical settings but different seeds for the random number generator), the distribution of the ratio of squared *NSE*'s is approximately $F(15, 15)$, implying that the probability of a 2:1 ratio of *NSE*'s in the two executions is about 6%. As J is increased, while reducing N proportionately to keep the total number of particles JN fixed, the approximate normal distribution of within group sample means $\overline{g}_j$ in (10) becomes less reliable. At this point we do not have a generic recommendation for combinations of J and N that produce the most reliable

Table 3: Sensitivity to number of Metropolis steps R in phase M

Metropolis steps R	Compute time (seconds)	Cycles L	Log Marginal Likelihood	Numerical Standard Error	Log Score	Numerical Standard Error
5	88	73	16,750.85	0.70	15,114.28	0.70
8	114	75	16,750.53	0.39	15,113.99	0.39
13	157	76	16,750.89	0.25	15,114.35	0.26
21	225	76	16,750.79	0.14	15,113.31	0.15
34	335	76	16,751.19	0.14	15,114.56	0.14
55	500	74	16,750.90	0.10	15,114.40	0.09
89	800	76	16,750.92	0.07	15,114.28	0.06
144	1358	76	16,750.80	0.04	15,114.19	0.05
`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$.						

NSE's.

Potential improvements in *NSE* are subject to diminishing returns with respect to the computational cost associated with increasing the value of R. As $R \to \infty$, the *NSE* for the Monte Carlo approximation of $\mathrm{E}\left[g\left(\theta\right) \mid y_{1:T}\right]$ approaches $\left\{\mathrm{var}\left[g\left(\theta\right) \mid y_{1:T}\right]/(JN)\right\}^{1/2}$. At some point it becomes more efficient to increase N rather than R in pursuit of greater accuracy. This point is model-dependent and cannot be identified with much precision with only $J = 16$ groups. But the evidence in Table 3 suggests that for the moments studied there the break-even point is above $R = 100$.

Table 4 illustrates the impact of changing the threshold D_1 at which CS transition is triggered. (All of the results in this Table use $D_2 = 0$; that is, the number of Metropolis iterations in each M phase is always R.) A weaker criterion (smaller D_1) lengthens the C phase, resulting in fewer cycles L. But the number of distinct points resampled in the S phase is then smaller, implying that for fixed R the expected *NSE* will be higher. There is a trade-off between smaller values of D_1 and larger values of R that keep *NSE* about the same. Among pairs D_1 and R that yield similar *NSE*, there is not much difference in compute time. For example, from Table 4, $(D_1 = 0.8, R = 34)$, $(D_1 = 0.6, R = 55)$ and $(D_1 = 0.4, R = 89)$ all have a compute time of about 600 seconds and *NSE* around 0.07. This is characteristic of all the models and functions of interest we have studied. For this reason we maintain a fixed value of D_1 (0.5) and do not provide further detail on the impact of changing D_1 going forward.

In many situations—in particular, those that satisfy the conditions for asymptotic normality of posterior distributions (Chen, 1985; Sweeting and Adekola, 1987; Geweke, 2005, Theorem 3.4.3)—an additional observation has less impact on the posterior distribution when sample size is large than when it is small. At the start of the algorithm the contribution of each observation to information about the parameter vector θ is relatively large, substantially reducing the probability assigned to large regions of Θ by the prior. Consequently $ESS/(JN)$ falls below the threshold D_1 triggering S and M phases more frequently in the early part of the sample than it does later on. Figure

Table 4: Interaction between choices of D_1 and R

ESS threshold D_1	Metropolis steps R	Compute time (seconds)	Cycles L	Log Marginal Likelihood	Numerical Standard Error	Log Score	Numerical Standard Error
0.2	34	180	32	16,751.12	0.26	15,114.57	0.26
0.4	34	261	56	16,751.01	0.11	15,114.43	0.12
0.6	34	389	91	16,750.70	0.09	15,114.14	0.08
0.8	34	737	184	16,750.87	0.07	15,114.31	0.07
0.2	55	250	31	16,750.92	0.15	15,114.32	0.15
0.4	55	393	56	16,750.10	0.12	15,114.48	0.13
0.6	55	618	93	16,750.86	0.08	15,114.24	0.07
0.8	55	1111	178	16,750.88	0.04	15,114.32	0.03
0.2	89	387	32	16,750.82	0.12	15,114.22	0.11
0.4	89	608	56	16,750.78	0.07	15,114.19	0.09
0.6	89	967	93	16,750.79	0.08	15,114.22	0.06
0.8	89	1835	183	16,750.74	0.05	15,114.21	0.04
`egarch_23`, $J = 16$, $N = 4096$, $D_2 = 0.0$.							

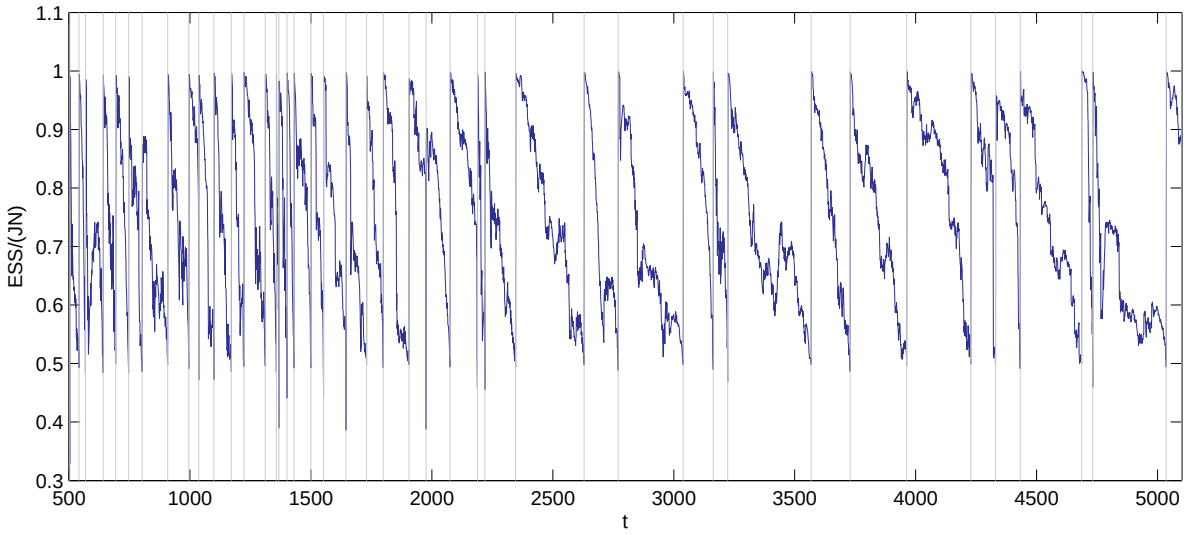

Figure 1: Effective sample size in the C phase as a function of sample size t. Vertical lines indicate CS transition (`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$).

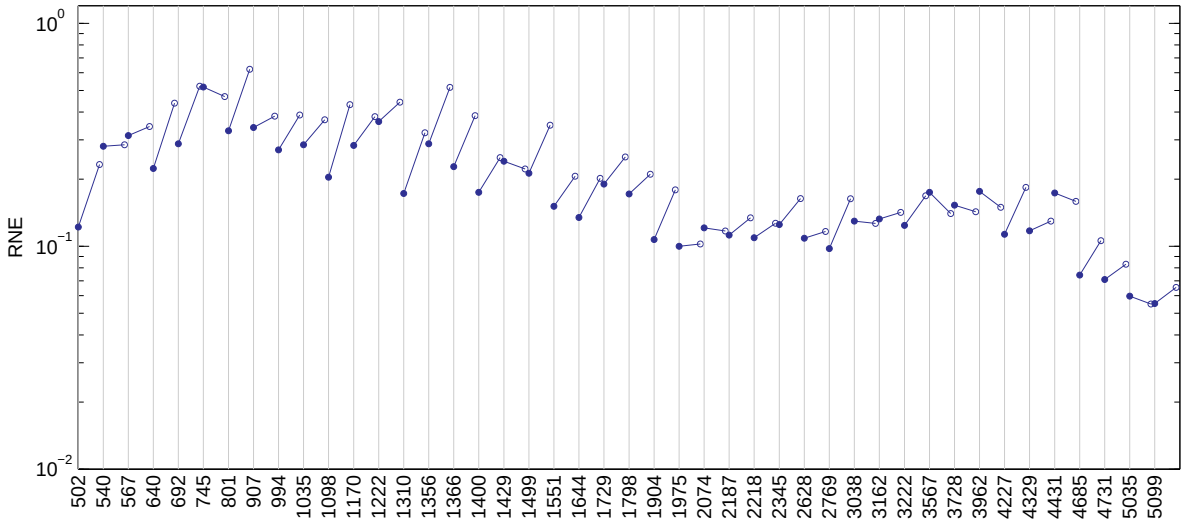

Figure 2: Relative numerical efficiency for the function of interest $p\left(y_{1:t_\ell} \mid \theta\right)$ in the M phase. A pair of points connected by a line segment shows the average RNE in the first 10 Metropolis steps (left point) and in the last 10 Metropolis steps (right point) of the M phase executed following the observation date indicated along the lower edge of the figure (`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$).

1 illustrates this for the `egarch_23` model and the 20-year daily return data set. It is characteristic of patterns in the other applications in this paper, and has been noted in the literature (e.g., Chopin, 2002; Fulop and Li, 2011). In general, cycle length is an increasing function of the number of observations t in the sample. For a given function form—for example, if the posterior distribution is nearly Gaussian—it is a decreasing function of the number of parameters in the model.

Although S and M phases occur less frequently as sample size increases, the computational cost of each successive M phase increases roughly in proportion to sample size. Thus, in the example shown in Figure 1 there are 7 passes through the M phase as the sample grows from 500 to 1000 observations and only 2 as it grows from 4500 to 5000, but this latter portion of the algorithm requires more compute time than the former.

The Metropolis steps in the M phase reduce the dependence between particles and thereby provide a more efficient representation of the posterior distribution. This is an expectation, not a feature of every realized step of the random-walk Metropolis stochastic process. Expected relative numerical efficiency (RNE) increases, with a limiting value of 1 as $R \to \infty$. Using the same values of J, N and R as in Table 2 and Figure 1, Figure 2 shows some aspects of this pattern within each M phase, exhibiting the RNE for Monte Carlo approximation of the posterior expectation of a function of θ that is calculated as part of the Metropolis arithmetic and therefore can be monitored at very little additional cost in computing time.

Although RNE tends to increase within each M phase, Figure 2 shows a systematic trend toward lower RNE as the sample progresses. This can be mitigated by increasing

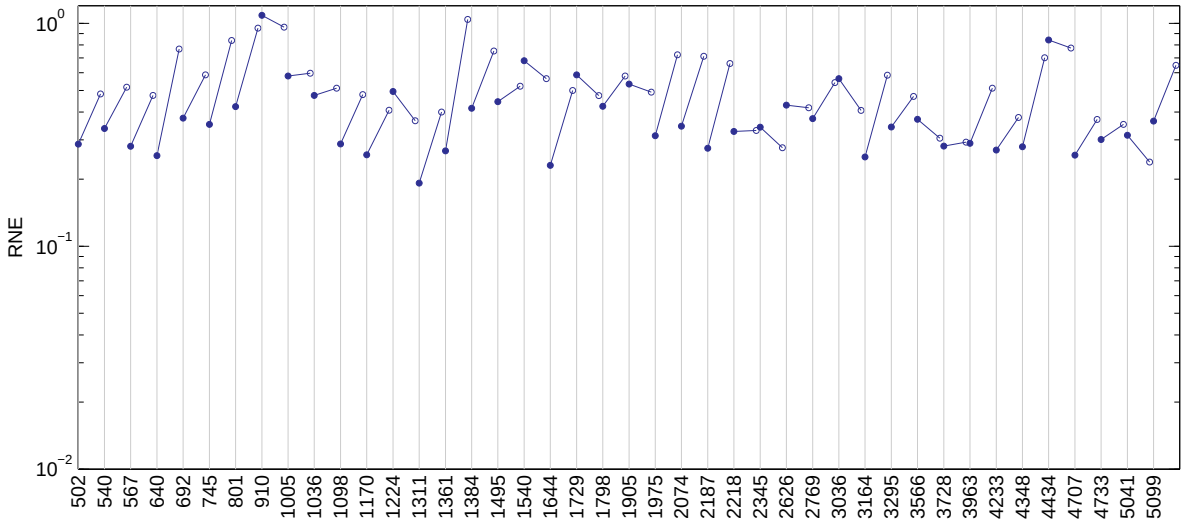

Figure 3: Relative numerical efficiency for the function of interest $p\left(y_{1:t_\ell} \mid \theta\right)$ in the M phase. A pair of points connected by a line segment shows the average RNE in the first 10 Metropolis steps (left point) and in the last 10 Metropolis steps (right point) of the M phase executed following the observation date indicated along the lower edge of the figure (`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 144$).

R. Figure 3 reports exactly the same computations for a different execution of the posterior simulator, except that here $R = 144$ rather than $R = 55$ as was the case in Figure 2. The patterns of RNE in these two figures are consistent with the comparisons made in Table 3. In both cases the user has reliable information about the accuracy of the approximation. The user who finds the degree of accuracy with $R = 55$ insufficient can improve the approximation by increasing either R or the number of particles. In principle the algorithm can be modified to monitor RNE and adjust the number of Metropolis iterations R_ℓ executed in each cycle so as to meet a target for RNE or numerical accuracy. We have experimented with various such modifications, but for this paper we use the simple rule described in Section 2.4: $R_\ell = R$ if $ESS/(JN) > D_2 = 0.2$ and $R_\ell = 3R$ otherwise.

3.3 Some specific aspects of the application

Models for asset returns, like the EGARCH models considered here, are primarily of interest for their predictive distributions. The cumulative predictive distribution function of an asset return is relevant for the pricing of derivative assets like options. To illustrate this application, consider the conditional probability of a return of less than -3% on date $t+1$ conditional on information through observation t. This conditional probability is obtained by Monte Carlo approximation of the expectation $\mathrm{E}\left[g(\theta) \mid y_{1:t}\right]$ over the posterior distribution of θ with $g(\theta) = P\left(y_{t+1} < -0.03 \mid \theta\right)$ as described in Section 2.3. Table 5 reports details on this approximation for t corresponding to two

Table 5: Moment of interest, $100 \cdot \mathrm{E}\left[P\left(Y_{t+1} < -0.03 \mid y_{1:t}, \theta\right)\right]$

	$t =$ March 31, 2009			$t =$ March 31, 2010		
	Posterior			Posterior		
R	mean	*NSE*	*RNE*	mean	*NSE*	*RNE*
5	9.6044	0.1098	0.0008	0.0627	0.0023	0.0019
8	9.7580	0.0714	0.0014	0.0672	0.0016	0.0042
13	9.7181	0.0371	0.0053	0.0697	0.0015	0.0057
21	9.7241	0.0370	0.0052	0.0742	0.0010	0.0122
34	9.7536	0.0192	0.0192	0.0757	0.0008	0.0216
55	9.7678	0.0165	0.0272	0.0763	0.0003	0.1623
89	9.7793	0.0096	0.0811	0.0766	0.0003	0.1284
144	9.7712	0.0087	0.0987	0.0768	0.0003	0.1633
`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.0$.						

different days in the sample: March 31, 2010 (the last day of the sample) and March 31, 2009. Volatility is much higher on the earlier date than it is on the later date.

These approximations are based on the collection of particles operational at the time the expectation is computed. After the last observation of the sample, we always execute S and M phases. In accordance with the discussion in Section 2.3, we also execute S and M phases immediately before evaluating the expectation for the earlier date.

In both cases, the *NSE* of the approximation declines substantially as R increases. The results for $t =$ March 31, 2009, are mostly unremarkable. Posterior moments are the same, up to *NSE*, for all values of R. Pairing the results with the compute times from Table 3, increases in R up to 89 are efficient, but the incremental compute time required for $R = 144$ would have been better allocated to more particles.

The results for $t =$ March 31, 2010 are more striking. The posterior means increase monontonically with R, and values for $R = 5$, 8 and 13 are too small even after taking into account limitations on the reliability of *NSE* due to just $J = 16$ groups of particles. To understand this feature, note first from the *RNE* entries in Table 5 that the effective sample sizes are about 8 to 25 particles per group for these small values of R. Second, most of the probability associated with very low returns in the less volatile environment of March 2010 arises from tails of the posterior distribution of θ. The result is a positively skewed distribution of the function of interest $g\left(\theta\right) = P\left(y_{t+1} < -0.03 \mid y_{1:t}, \theta\right)$, with the important right tail poorly (if at all) represented with these very small effective sample sizes. This does not imply a systematic bias in posterior moment approximation, but it does imply that more often than not the computed posterior moment will be too low. (The *NSE* will be low as well.) This phenomenon arises again in Section 5.2 in a different context. In general, very small effective sample sizes imply skewed sampling distributions of computed posterior moments when the corresponding functions of interest themselves have skewed posterior distributions. Since the shape of the posterior distribution of a function of interest can be less than obvious, the practical implication is that the user should beware very small effective sample sizes.

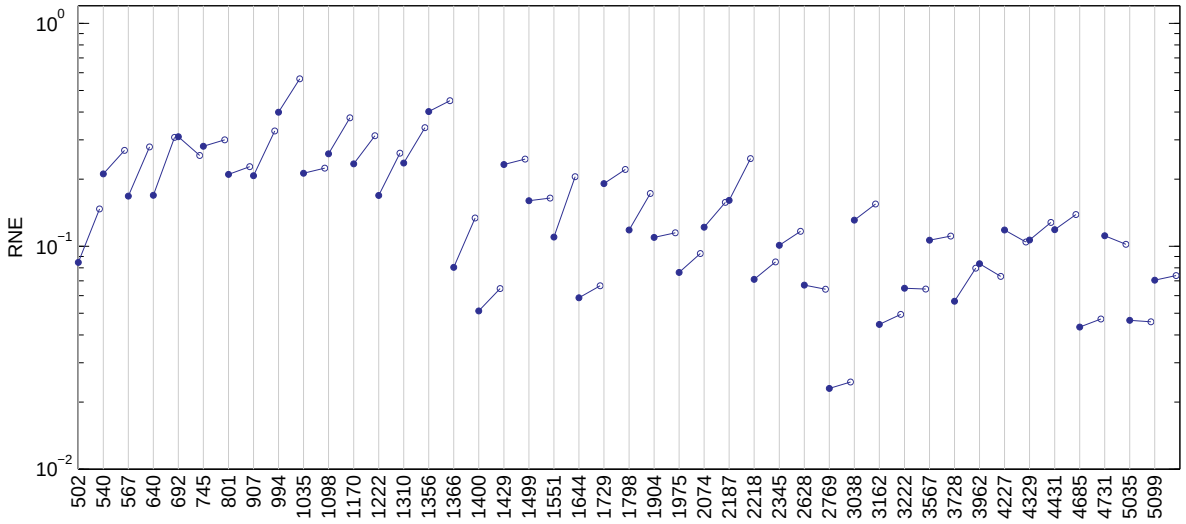

Figure 4: Relative numerical efficiency for the posterior moment $\mathrm{E}\left[P\left(y_{t+1<-0.03} \mid y_{1:t}, \theta\right)\right]$ in the M phase. A pair of points connected by a line segment shows the average RNE in the first 10 Metropolis steps (left point) and in the last 10 Metropolis steps (right point) of the M phase executed following the observation date indicated along the lower edge of the figure (`egarch_23`, $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$).

Figure 4 shows RNE for the function of interest $\mathrm{E}\left[P\left(y_{t+1} < -0.03 \mid y_{1:t}, \theta\right)\right]$ in the same way that Figure 2 did for $p\left(y_{1:t} \mid \theta\right)$. Again, the patterns are similar, and again the RNE of other functions of interest we have studied behaves in much the same way.

In the `egarch_23` model there are 2 permutations of the factors v_{kt} and 6 permutations of the components of the normal mixture probability distribution function of ε_t. This presents a severe challenge for single-chain Markov chain Monte Carlo as discussed by Celeux et al. (2000) and Jasra et al. (2007b), and for similar reasons importance sampling is also problematic. The problem can be mitigated (Fruhwirth-Schnatter, 2001) or avoided entirely (Geweke, 2007) by exploiting the special "mirror image" structure of the posterior distribution. But these models are still interesting as representatives of multimodal and ill-behaved posterior distributions in the context of generic posterior simulators. We focus here on the 6 permutations of the normal mixture in `egarch_23`.

Consider a parameter vector θ with three distinct values of the triplets (p_s, μ_s, σ_s) $(s = A, B, C)$. There are six distinct ways in which these values could be assigned to components $i = 1, 2, 3$ of the normal mixture (16). These permutations define six points θ_u $(u = 1, \ldots, 6)$. For all sample sizes t, the posterior densities $p\left(\theta_u \mid y_{1:t}\right)$ at these six points are identical. Let θ' be a different parameter vector with analogous permutations θ'_u $(u = 1, \ldots, 6)$. As the sample adds evidence $p\left(\theta_u \mid y_{1:t}\right) / p\left(\theta_{u'} \mid y_{1:t}\right) \overset{as}{\to} 0$ or $p\left(\theta_u \mid y_{1:t}\right) / p\left(\theta_{u'} \mid y_{1:t}\right) \overset{as}{\to} \infty$. Thus, a specific triplet set of triplets (p_s, μ_s, σ_s) $(s = A, B, C)$ and its permutations will emerge as pseudo-true values of the parameters (Geweke, 2005, Section 3.4).

These properties will be exhibited in the marginal distributions, as well. Consider the

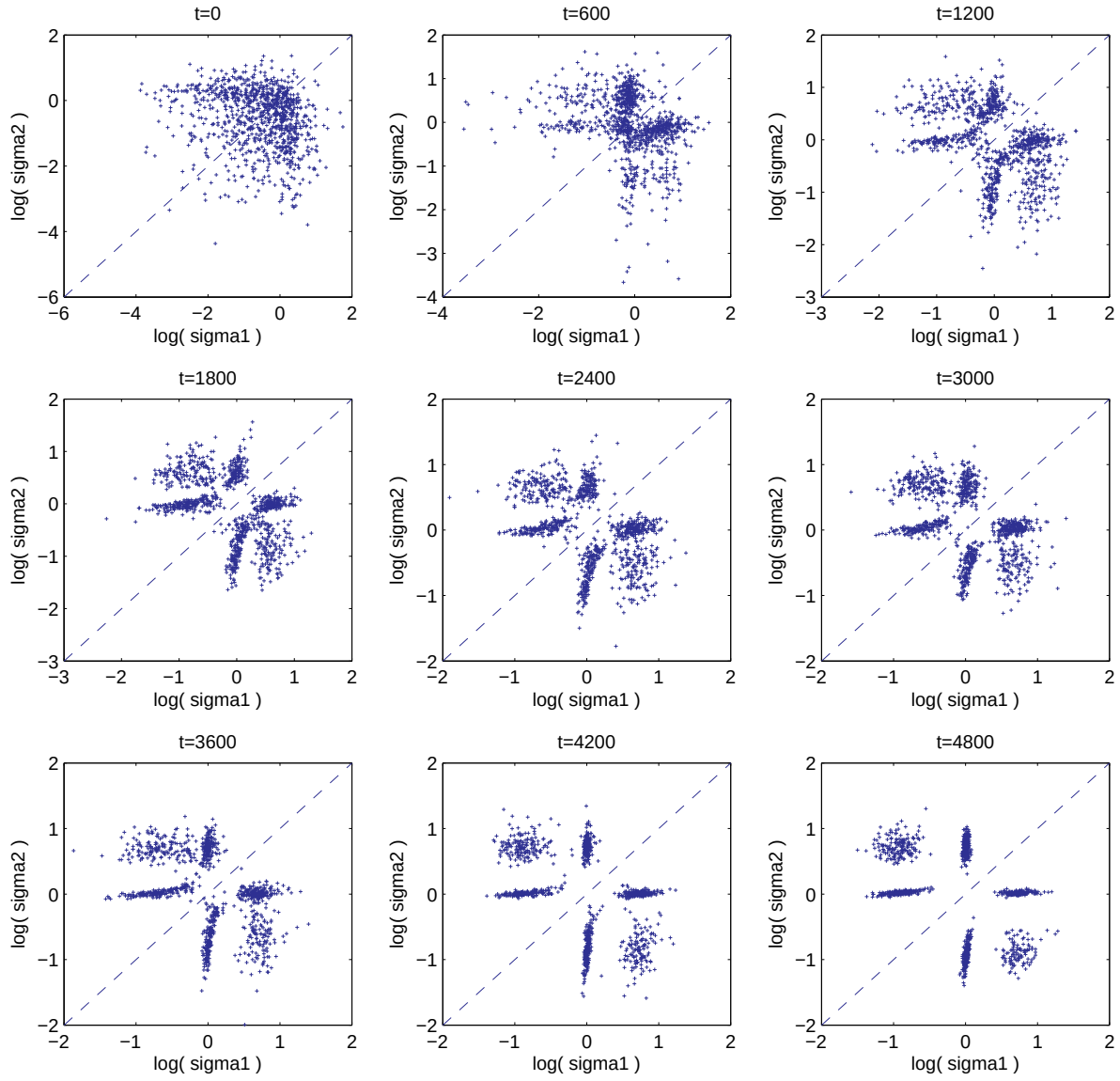

Figure 5: Scatterplots of a subset of particles $(\log \sigma_1, \log \sigma_2)$ from selected posterior distributions conditional on $y_{1:t}$ ($J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 34$).

pair (σ_1, σ_2), which is the case portrayed in Figure 5. The scatterplot is symmetric (up to the random sampling in the simulator) about the axis $\sigma_1 = \sigma_2$ in all cases. As sample size increases six distinct and symmetric modes in the distribution gradually emerge. These reflect the full posterior distribution for the normal mixture components of the `egarch_23` model (i.e., marginalizing on all other parameters) that is roughly centered on the components $(p = 0.17,\ \mu = 0.16,\ \sigma = 0.40)$, $(p = 0.85,\ \mu = 0.01,\ \sigma = 1.01)$ and $(p = 0.01,\ \mu = -1.36,\ \sigma = 1.96)$. The progressive decrease in entropy with increasing t illustrates how the algorithm copes with ill-behaved posterior distributions. Particles gradually migrate toward concentrations governed by the evidence in the sample. Unlike Markov chain Monte Carlo there is no need for particles to migrate between modes, and unlike importance sampling there is no need to sample over regions eliminated by the data (on the one hand) or to attempt to construct multimodal source distributions (on the other). The algorithm proposed in this paper adapts to these situations without specific intervention on a case-by-case basis.

Figure 6 presents the same analysis, for the probability components of the normal mixture. Symmetry and increasing concentration with t are again evident, as are six distinct modes at $t = 4800$. Similar study of the parameters μ of the normal mixture also shows symmetry, but six modes are not so evident.

4 Example: Dynamic multinomial probit model

This model describes choices made by individuals in a sequence of time periods, for example whether to commute between home and work by automobile, public transportation or autonomously (walking or cycling). As applied to human behavior this model has the key strategic advantage of not imposing independence of irrelevant alternatives, which is inconsistent with microeconomic theory and is a property of the simpler and more widely applied multinomial logit model. Evaluating the likelihood function entails computing the probability that a vector with a multivariate normal distribution will simultaneously satisfy several linear inequality constraints, a problem that can only be solved by simulation except in simple illustrative cases not typical of research applications. Alternatively, likelihood function evaluation can be avoided by augmenting the data with the latent utilities in Bayesian approaches to inference, at the cost of being unable to assess marginal or predictive likelihoods. Both approaches demand substantial computing time as detailed in Geweke et al. (1997).

This section applies the algorithm of Section 2 using a reliable simulator to assess the intractable probabilities in the likelihood function. The simulator provides an unbiased evaluation of the entire likelihood function. As detailed in Section 2.3 this property is sufficient for quite a few approaches to posterior simulation, including the one taken here. In common with other unbiased probability simulators this one requires that the user select a tuning parameter (here, in the form of the number of iterations), but the only consequence of an imprudent choice is a poor approximation that will be indicated reliably by the numerical standard errors provided by the algorithm. The illustration

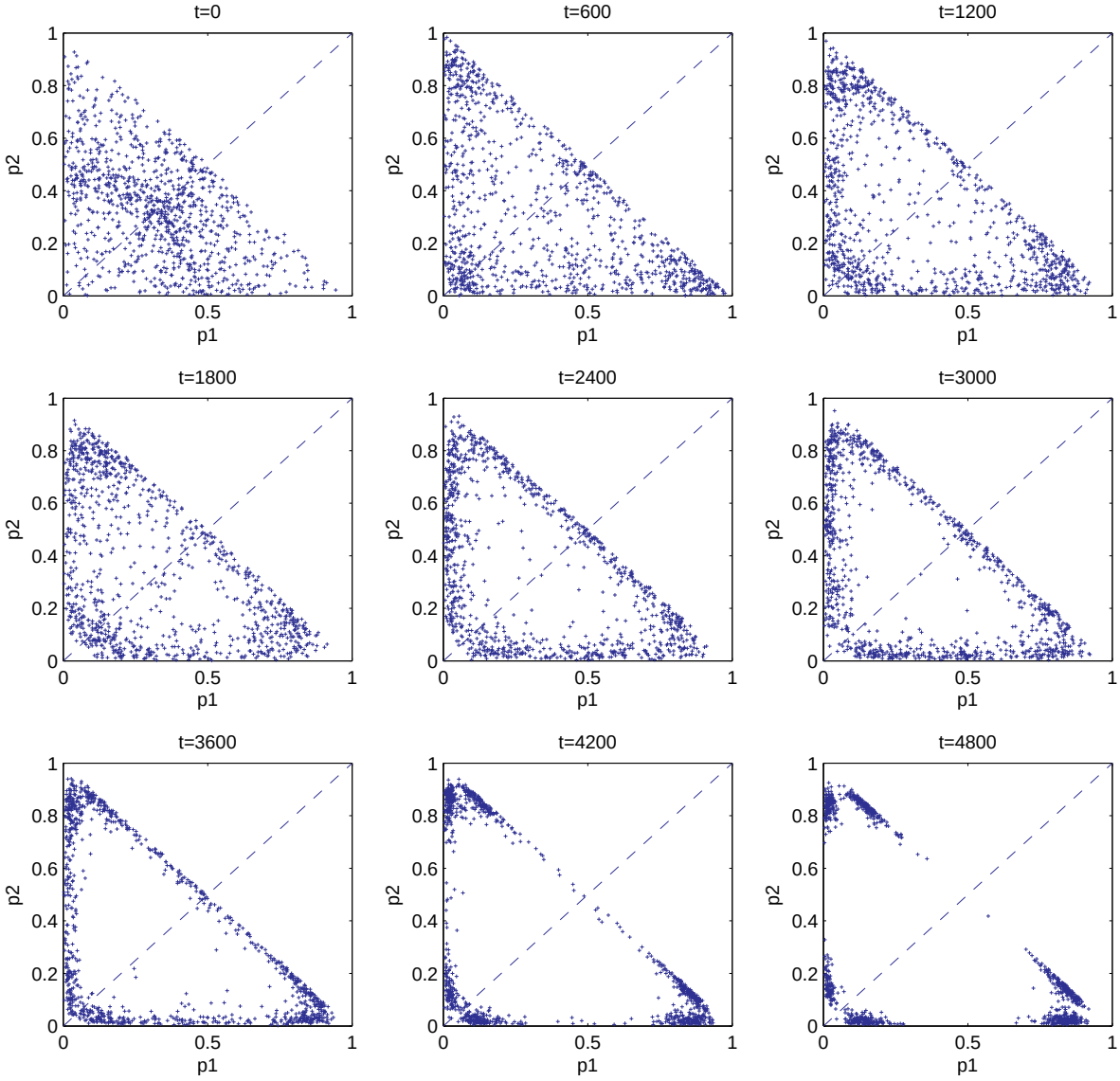

Figure 6: Scatterplots of a subset of particles (p_1, p_2) from selected posterior distributions conditional on $y_{1:t}$ ($J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 34$).

here supports the assertion of claim 1 in Section 1.3 that the algorithm can be extended to situations in which direct evaluation of the likelihood function is not possible.

4.1 Model and data

The dynamic multinomial probit model pertains to a set of individuals $i = 1, \ldots, I$ observed in time periods $t = 1, \ldots, T$. Each individual i in each time period t chooses exclusively from alternatives $c = 1, \ldots, C$. For each individual i and time period t there is an observed personal characteristic x_{it} (e.g., income) and observed choice characteristics z_{ict} $(c = 1, \ldots, C)$ (e.g., prices of the alternative choices) and an observed choice $y_{it} \in \{1, \ldots, C\}$. Choices are conditionally independent across individuals i but conditionally dependent across time periods t for each individual i. The fundamental unit of observation is the individual, i, and in this example i plays the same role as t in the description of the algorithm in Section 2. As the algorithm proceeds all T observations for an individual are always introduced as a group.

Consistent with microeconomic theory, individual i derives utility u_{ict} from choice c at time t, and makes that choice y_{it} that maximizes utility. It will turn out shortly that $P(u_{ict} = u_{ic't}) = 0$, so we may write

$$y_{it} = \arg\max_c u_{ict} \iff u_{i,y_{it},t} \geq u_{ic't} \, (c' = 1, \ldots, C) \, . \tag{20}$$

A linear probability model specifies

$$\mu_{ict} = \beta_{0c} + \beta_{1c} x_{it} + \gamma_c z_{ict} - \gamma_C z_{iCt}, \qquad u_{ict} = \mu_{ict} + \varepsilon_{ict} \tag{21}$$

for $i = 1, \ldots, I$; $t = 1, \ldots, T$; $c = 1, \ldots, C-1$ and $u_{iCt} = 0$. The asymmetric treatment of the last alternative is a conventional normalization recognizing the fact that (20) is invariant to the addition of terms a_{it} to u_{ict}.

The multinomial probit model specifies

$$\varepsilon_{it} = (\varepsilon_{i1t}, \ldots, \varepsilon_{i,C-1,t})' \sim N(0, \Sigma^*) \, ,$$

with

$$\Sigma^* = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & 1 \end{bmatrix} \tag{22}$$

in the case $C = 4$. The asymmetric treatment of $\mathrm{var}(\varepsilon_{i,C-1,t})$ is a conventional normalization recognizing the fact that (20) is invariant to the multiplication of u_{ict} by positive constants b_{it}. The dynamic multinomial probit model (Geweke et al., 1997) specifies

$$\underset{T \times T}{R} = [r_{st}] \, , \;\; r_{st} = \rho^{|s-t|}, \;\; \Sigma = R \otimes \Sigma^*, \;\; \varepsilon_i = (\varepsilon_{i1}', \ldots, \varepsilon_{iT}')' \sim N(0, \Sigma) \, . \tag{23}$$

The choices y_{it} are observed but the utilities u_{ict} are latent. Conditional on covariates x_{it} and z_{ict} the choice sequences $y_i = (y_{i1}, \ldots, y_{iT})'$ are mutually independent whereas

y_{is} and y_{it} $(s \neq t)$ are not independent unless $\rho = 0$. Conditional on covariates,

$$P(y_i) = \int_{S_i} \phi(\varepsilon; 0, \Sigma)\, d\varepsilon \tag{24}$$

where

$$S_i = \{\varepsilon_{it} : u_{i,y_{it},t} \geq u_{ict} \;\; (c = 1, \ldots, C; t = 1, \ldots, T)\}.$$

From (20) and (21) S_i is a function of covariates and the parameters β_{0c}, β_{1c} and γ_c $(c = 1, \ldots, C)$. Hence the expression (24) is a function of all of the parameters of the model.

The integral (24) cannot be evaluated analytically, and quite a few simulation algorithms have been proposed for its approximation. Of these the Geweke-Hajivassilliou-Keane (*GHK*) probability simulator (Börsch-Supan and Hajivassiliou, 1993; Geweke and Keane, 2001) is generally considered the most reliable (Hajivassilliou et al., 1996). The *GHK* simulator works with a linear transformation of the random variables u_{ict} to a random vector v in which the set S_i becomes a hyper-rectangle defined by the endpoints (a_c, b_c) $(c = 1, \ldots, C)$. It draws $\widetilde{v}_c \sim p(v_c \mid \widetilde{v}_1, \ldots, \widetilde{v}_{c-1})$ $(c = 1, \ldots, C-1)$ recursively, and then computes the random variable

$$\widetilde{p} = P(a_1 < v_1 \leq b_1)\, \Pi_{c=2}^{C} P(a_c < v_c \leq b_c \mid \widetilde{v}_1, \ldots, \widetilde{v}_{c-1}).$$

This describes a single iteration of the *GHK* probability simulator. Since

$$\mathrm{E}(\widetilde{p}) = P[a_c < v_c \leq b_c \;\; (c = 1, \ldots, C)] \tag{25}$$

(Börsch-Supan and Hajivassiliou, 1993; Geweke and Keane, 2001), the mean of independent executions of the simulator is strongly consistent for $P(y_i)$ in (24). In the context of the description of the algorithm in Section 2, u in (3) is the randomly selected seed of the random number generator at the time a new proposal u_{jn} is drawn in (7). The property (25), and the fact that the likelihood function is the product of the terms (24) over $i = 1, \ldots, I$, together imply (3).

The number of iterations H used in the *GHK* simulator is an additional design parameter in implementing the entire algorithm. This is true of simulated likelihood methods generally, as is the fact that the accuracy of approximation and execution time both increase with the number of iterations. The methods introduced in Section 2.3 for evaluating the accuracy of approximation of posterior moments remain valid. Numerical standard errors calculated in this way appropriately reflect the additional uncertainty introduced by the simulation noise surrounding likelihood function approximation. Increasing the number of simulations, especially when that number is small, leads to a verifiable reduction in the numerical standard errors of the algorithm's approximation of posterior moments.

The algorithm operates on transformed parameters θ. Priors and transformations are detailed in Table 6. Priors are all Gaussian with the indicated means and standard deviations. The evaluation of the likelihood function closely follows Rossi et al. (2005) and in

Table 6: Parameters and prior distribution in the multinomial probit model.

	Mean	Std Dev	Transformation	
θ_{1c}	0	0.1	$\beta_{0c} = \theta_{1c}$	$(c = 1, \ldots, C-1)$
θ_{2c}	0	0.1	$\beta_{1c} = \theta_{2c}$	$(c = 1, \ldots, C-1)$
θ_{3c}	$-2^{-1/2}$	0.1	$\gamma_c = \theta_{3c}$	$(c = 1, \ldots, C)$
θ_{4c}	0	0.1	$\sigma_{cc} = \exp(\theta_{4c})$	$(c = 1, \ldots, C-1)$
θ_{5cd}	$\tanh^{-1}(0.5)$	0.1	$\sigma_{cd} = \tanh(\theta_{5cd})$	$(c, d = 1, \ldots, C; c \neq d)$
θ_6	$\tanh^{-1}(0.5)$	0.1	$\rho = \tanh(\theta_6)$	

particular the `baysm R` package that accompanies that book. The essential operations in the *GHK* algorithm involve linear algebra, generation of normal random numbers, and the evaluation of the univariate normal cumulative distribution function, but there is no data-dependent branching. All three of these operations are SIMD-compatible.

The data set is constructed. The unconditional distribution of all covariates x_{it} and z_{ijt} is standard normal. Covariates are independent across individuals and choices, and all covariates follow first-order autoregressive processes with correlation coefficient 0.5. The choice vectors y_i are generated using (20)–(23) with $\beta_{0c} = \beta_{1c} = 0$, $\gamma_c = -2^{-1/2}$ $(c = 1, \ldots, C)$, $\sigma_{ii} = 1$ $(i = 1, \ldots, C-1)$, $\sigma_{ij} = 0.5$ $(i \neq j)$, and $\rho = 0.5$, following the experimental design in Geweke et al. (1997). There are $C = 4$ choices, $T = 4$ time periods and $I = 500$ individuals in the artificial sample.

4.2 Performance

All of the inference for the MNP model is based on $2^{16} = 65,536$ particles in $J = 2^4 = 16$ groups of $N = 2^{12} = 4,096$ particles each. Table 7 compares execution time and algorithm performance in computing log marginal likelihood for some alternative choices of R and H. As is the case generally, cycle length increases with the size of the sample. In this example effective sample size deteriorates more rapidly in the C phase than was the case for the EGARCH example in Section 3, resulting in more cycles relative to the sample size. From the compute times in Table 7 it is easy to see that the M phase accounts for almost all the compute time, even when $R = 1$.

In the approximation of log marginal likelihood the algorithm performs poorly when the number of iterations H of the *GHK* simulator is small. (This is consistent with conventional wisdom about the *GHK* algorithm, noted as long ago as Geweke et al., 1994.) In particular there appears to be a noted downward bias in reported log marginal likelihood for $H = 25$, reduced but still evident for $H = 50$, and undetectable for $H = 100$. Fulop and Li (2011) have noted a similar phenomenon for a simulated likelihood function in a nonlinear state space model. It is important to understand systematic anomalies of this kind in any algorithm.

Figures 7 and 8 provide further evidence. As described in Section 2.3, each Metropolis iteration in the M phase is executed in two steps: the first step (7) operates on seeds for the random number generator used in the *GHK* simulator and the second step (8)

Table 7: Sensitivity to number of Metropolis steps R and *GHK* iterations H

Metropolis steps R	*GHK* iterations H	Compute time (seconds)	Cycles L	Log Marginal Likelihood	Numerical Standard Error
1	25	746	77	-1892.83	0.35
1	50	1060	68	-1890.55	0.24
1	100	1683	61	-1890.00	0.18
1	200	2724	55	-1888.95	0.29
2	25	1548	79	-1892.11	0.25
2	50	2089	68	-1890.05	0.24
2	100	3197	61	-1889.48	0.16
2	200	5142	55	-1889.74	0.13
3	25	2408	80	-1891.99	0.19
3	50	3127	68	-1889.84	0.18
3	100	4878	62	-1889.88	0.08
3	200	7784	55	-1889.69	0.07
5	25	4111	82	-1891.92	0.17
5	50	5346	69	-1889.98	0.14
5	100	8067	62	-1889.71	0.07
5	200	13,255	57	-1889.75	0.05
8	25	6749	83	-1890.09	0.27
8	50	8535	69	-1889.62	0.13
8	100	13,363	63	-1889.75	0.06
8	200	21,056	57	-1889.64	0.05
13	25	10,692	81	-1891.10	0.16
13	50	14,237	70	-1889.75	0.15
13	100	20,805	62	-1889.67	0.05
13	200	34,049	57	-1889.71	0.04
21	25	18,059	82	-1890.52	0.21
21	50	22,940	69	-1889.99	0.04
21	100	35,333	63	-1889.78	0.04
21	200	54,205	56	-1889.72	0.05
$J = 16$, $N = 4096$, $D_1 = 0.50$, $D_2 = 0.20$.					

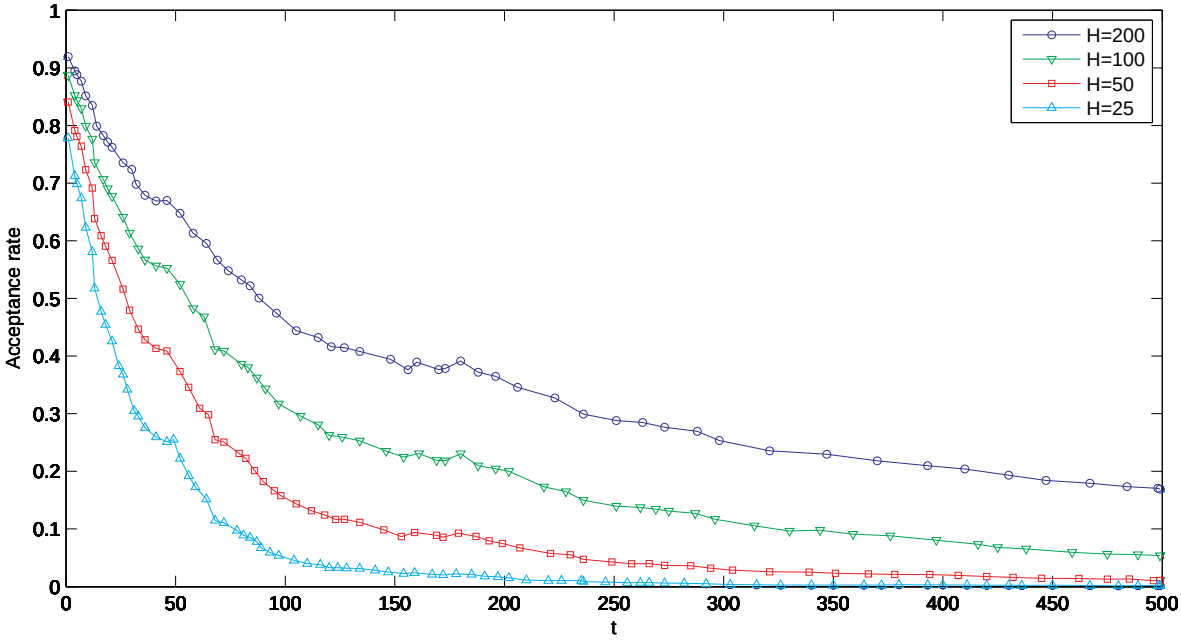

Figure 7: Average acceptance rates in Metropolis steps operating on *GHK* seeds. Markers indicate observations at which M phases were executed ($J = 16$, $N = 4096$, $D_1 = 0.50$, $D_2 = 0.20$, $R = 5$).

operates on parameter vectors. As depicted in Figure 7, the acceptance rate in the first step (over seeds) deteriorates steadily as t increases. For $H = 25$ acceptance rates drop to near zero in the last third of the sample. Acceptance rates improve with larger values of H, though at the end of the sample the acceptance rate is still below 0.2 even for $H = 200$. Figure 8 documents the implications of this phenomenon for the number of unique seeds at time t in one group of 4096 particles. The S phase resamples multiple copies of particles that have high weights. Each copy has the same seed. If acceptance rates in the Metropolis steps are low, the number of unique seeds deteriorates over successive S phases. There may still be many unique parameter vectors. This depends on acceptance rates in the second step of the Metropolis iterations, where we experience no difficulties in maintaining the target acceptance rate of 0.25.

To obtain good approximations of the likelihood function (6) either large H or a large sample of unique seeds is needed. As $H \to \infty$ it makes little difference if each particle uses the same seed. When H is small, a single seed may produce a poor approximation of the likelihood surface, but the problem is mitigated when we are able to integrate across many seeds.

In the extreme situation where all particles use the same seed, the algorithm produces a sample of parameter vectors from the posterior distribution corresponding to the approximate likelihood associated with that particular seed. If, as is the case here, the exact likelihood reflects the true data generating process, the approximate likeli-

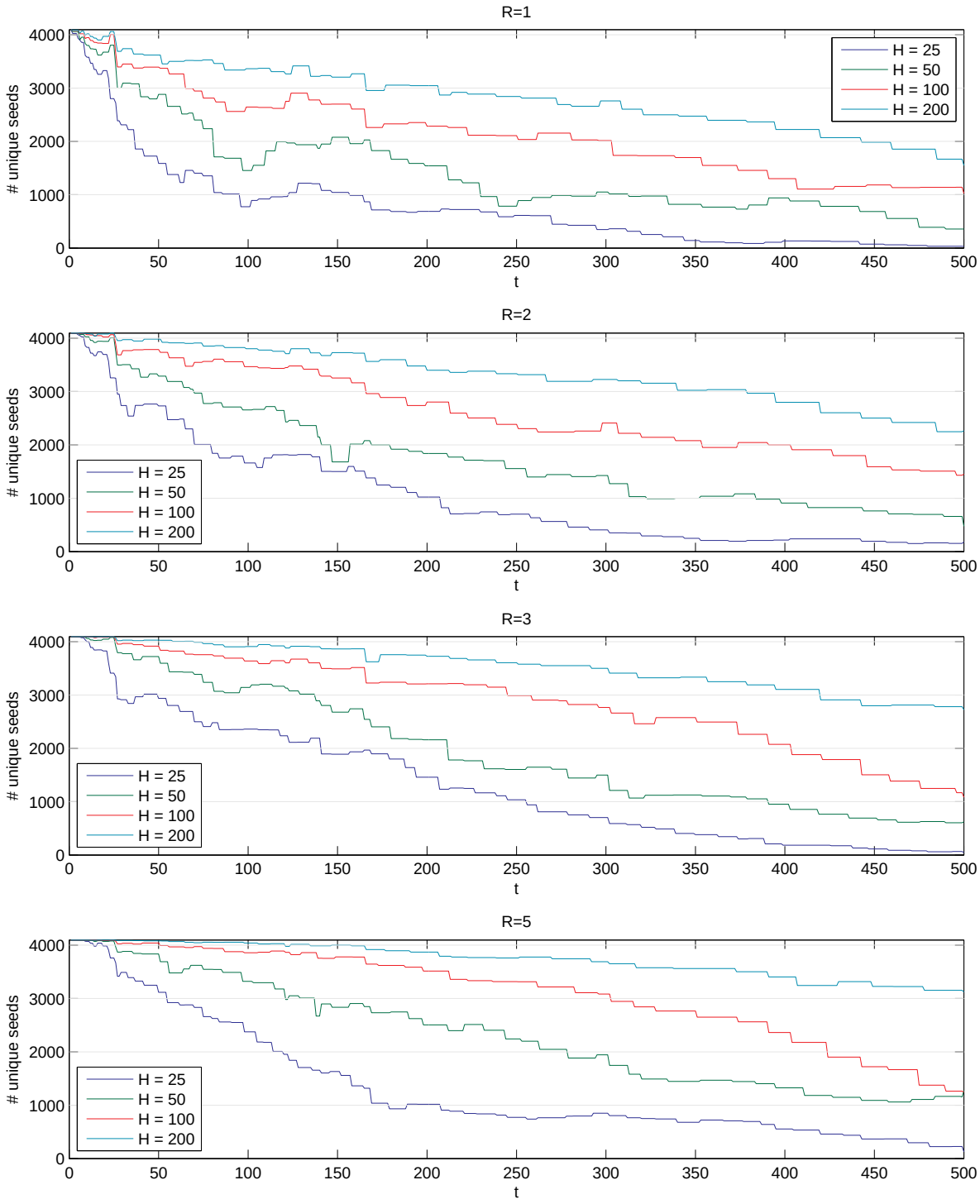

Figure 8: Number of unique *GHK* seeds u_{jn} in a single group of $N = 4096$ particles at observation t ($J = 16$, $N = 4096$, $D_1 = 0.50$, $D_2 = 0.20$).

hood corresponds to a misspecified model and the reported marginal likelihood will be systematically too low.

Ideally, one would like to adapt the Metropolis sampler to maintain some target acceptance rate. While we are able to do this when sampling parameter vectors (as described in Section 2.4), this does not appear to be feasible when sampling seeds. We do not at present know of any solution to this problem other than to increase the number of *GHK* simulations to attain acceptance rates sufficient to maintain an adequate diversity of seeds. By design, random number generators are highly sensitive to the choice of seed.

Fortunately this does not present an overwhelming problem for applications. At runtime the most important diagnostic for this problem is the number of unique particles and acceptance rates, which our software provides for both the parameter component of the particle, θ_{jn}, and the seed component of the particle, u_{jn}. The user effectively has access to the information in Figures 7 and 8 and can discern, at early cycles, that the selected combination of R and H will not be adequate for the evaluation of log marginal likelihood in the problem at hand. Furthermore, the problem appears confined largely to this specific task. As illustrated in the next section, it seems virtually nonexistent for posterior moments.

4.3 Some specific aspects of the application

The MNP model described in Section 4.1 is often used to predict the impact on choice probabilities of changes in alternative-specific characteristics z_{ict}. For example, in an application to retail sales the model could be used to predict the effect on market share of a change in the price in one or more of the alternative products. The prediction depends on the values of all of the covariates as well as the parameters of the model. Here we use the model to infer these effects from the posterior distribution of the change in choice probabilities corresponding to the covariate values in the sample. Whether or not this would be reasonable in an actual application depends on whether the covariate values in the sample are representative of the population, as might in fact be true if the sample were randomly selected from that population. Otherwise the application would be complicated by the need to model the population distribution of covariates.

Covariate distributions and model parameters are symmetric across the $C = 4$ alternatives in our artificial data set, and therefore these choice probabilities are 0.25 for each alternative, up to sampling variability in the covariates. We include the choice probability for the first alternative as our function of interest (a) in this section. A second function of interest (b) is the choice probability for the first alternative if its price z_{i1t} is reduced by one unit for each individual i in each time period t, as might the the case (for example) if a coupon were available each period. This change will increase the choice probability from its value in the first function of interest. A third function of interest (c) is the same choice probability if the price z_{i2t} of the second alternative were also reduced at the same time. This will reduce the choice probability, compared with the second function. In the fourth function (d) the prices of the first three alternatives are all reduced by one unit, further reducing the choice probability for the first alternative.

In the fifth and final function (e) prices of all four alternatives are reduced by one unit. By symmetry the population choice probabilities return to their symmetric values of 0.25, but because the new configuration of prices is less typical of the sample we expect the posterior variance to be greater in the fifth case than in the first.

Since the functions of interest all involve probabilities—as will generally be the case with this model—it is necessary to approximate the integral (24) associated with each particle θ_{jn}. In this evaluation there is no need to use the final seed u_{jn} associated with θ_{jn} in the sequential Monte Carlo algorithm, because (1) the choice of seed is now not part of the evolution of particles in the algorithm and (2) the distribution of seeds u_{jn} is not particularly attractive for this purpose, especially for small H and R. Instead we run the random number generator continuously in evaluating functions of interest, so that the approximation error is now independent across particles. There is also no particular reason to use the same number of *GHK* iterations H as in the algorithm, but we do that for the results reported here.

Table 8 documents the relevant aspects of posterior moment approximation in the case $R = 5$, for alternative values of H. Numerical variance due to simulation error in the approximation of choice probability is roughly proportional to $1/H$. Even as $H \to \infty$ simulation error due to finite J, N and R will remain, so returns to increasing H are diminishing. The evidence in this table shows substantial returns in moving from $H = 25$ to $H = 50$, but suggests that returns to increasing H are largely exhausted by $H = 100$. As anticipated the probability of the first choice increases in moving from alternative (a) or (e) to (d), again from (d) to (c), and again from (c) to (b). Price promotion for the first choice alone increases market share from 25% to about 46%. Market shares for (a) and (e) are about the same, but (again, as anticipated) the posterior standard deviation is substantially greater for case (e) than for case (a). These findings are the same across all variants in H and R of the algorithm explored here, as one would expect given that they are all simulation-consistent.

Table 9 provides the same information for alternative values of R, using the value $H = 100$ suggested by these results. In any application of the algorithm a sufficiently large number of iterations R in the M phase will eventually generate particles that are very nearly independently as well as identically distributed. Further iterations cannot improve this situation, and in every application there will come a point, before the particles are nearly independently distributed, where the returns to computing time in pursuit of accuracy are greater from increasing the number of particles JN than from increasing R. With $J = 16$ groups this point is not particularly well identified due to the error in approximating the relevant simulation variance as detailed in Section 2.3. In this example, Table 9 clearly indicates that there is little point in increasing R beyond 21, given that relative numerical efficiency at that point is close to the value 1.00 associated with independently distributed particles.

Table 8: Probability of Choice 1 given some alternative conditions

GHK iterations *H*	Compute Time (seconds)	Posterior Mean	Posterior Std Dev	*NSE*	*RNE*
(a) No change in price					
25	4111	0.26058	0.00998	0.00025	0.02406
50	5346	0.25988	0.00997	0.00012	0.10573
100	8067	0.26000	0.00982	0.00012	0.10930
200	13,255	0.25997	0.00987	0.00011	0.12805
(b) Price of choice 1 reduced by 1 unit					
25	4111	0.46117	0.01491	0.00033	0.03176
50	5346	0.46031	0.01487	0.00016	0.13220
100	8067	0.46044	0.01464	0.00017	0.11181
200	13,255	0.04606	0.01476	0.00016	0.12987
(c) Price of choices 1 and 2 reduced by 1 unit					
25	4111	0.36567	0.01612	0.00042	0.02293
50	5346	0.36507	0.01603	0.00023	0.07490
100	8067	0.36495	0.01581	0.00015	0.17491
200	13,255	0.36528	0.01607	0.00016	0.15433
(d) Price of choices 1, 2 and 3 reduced by 1 unit					
25	4111	0.29761	0.01168	0.00048	0.01853
50	5346	0.29704	0.01672	0.00023	0.07835
100	8067	0.29689	0.01649	0.00014	0.20679
200	13,255	0.29728	0.01671	0.00016	0.17172
(e) Price of all choices reduced by 1 unit					
25	4111	0.25251	0.01699	0.00054	0.01493
50	5346	0.25211	0.01678	0.00027	0.05772
100	8067	0.25195	0.01656	0.00014	0.22741
200	13,255	0.25234	0.01680	0.00016	0.16134
$J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 5$.					

Table 9: Probability of Choice 1 given some alternative conditions

Metropolis steps R	Compute Time (seconds)	Posterior Mean	Posterior Std Dev	*NSE*	*RNE*
(a) No change in price					
5	8067	0.2600	0.0982	0.00012	0.10930
8	13,363	0.25999	0.00993	0.00007	0.35273
13	20,805	0.25987	0.00991	0.00004	0.806290.
21	35,333	0.25984	0.00988	0.00005	0.72275
(b) Price of choice 1 reduced by 1 unit					
5	8067	0.46044	0.01464	0.00017	0.11181
8	13,363	0.46048	0.01477	0.00009	0.39364
13	20,805	0.46038	0.01475	0.00009	0.43056
21	35,333	0.46039	0.01476	0.00005	1.37777
(c) Price of choices 1 and 2 reduced by 1 unit					
5	8067	0.36495	0.01581	0.00014	0.17491
8	13,363	0.36513	0.01595	0.00009	0.53050
13	20,805	0.36501	0.01596	0.0009	0.48796
21	35,333	0.36507	0.01591	0.00006	1.11836
(d) Price of choices 1, 2 and 3 reduced by 1 unit					
5	8067	0.29689	0.01649	0.00014	0.20679
8	13,363	0.29710	0.01664	0.00009	0.52617
13	20,805	0.29605	0.01662	0.00010	0.38526
21	35,333	0.29704	0.01654	0.00007	0.92350
(e) Price of all choices reduced by 1 unit					
5	8067	0.25195	0.01656	0.00014	0.22741
8	13,363	0.25218	0.01673	0.00009	0.52617
13	20,805	0.25198	0.01670	0.00010	0.46086
21	35,333	0.25209	0.01662	0.00007	0.99389
$J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $H = 100$.					

5 Example: Vector autoregression model

This model was proposed by Sims (1980) for research in support of macroeconomic policy, including prediction of macroeconomic aggregates. Many central banks include the vector autoregression model in a suite of models that are regularly updated and use it to produce predictions over horizons of one or two years. The model is linear but typically has many parameters relative to the data available for inference. The model studied in this section has 231 parameters. The data in the application consists of seven time series, all with strong serial correlation, observed for 243 quarters.

The structure of the model is simple, that of a multivariate regression, but classical point estimates of the parameters produce inferior predictions due to the meagre data set and the failure to allow for parameter uncertainty. There is also substantive prior information about the model parameters, and even approximate Bayesian inference incorporating this information leads to competitive forecasts, as detailed in Litterman (1986) and Geweke and Whiteman (2006, Section 4). Contemporary approaches usually employ Gibbs sampling (Primiceri, 2005), which is attractive given the structure of the posterior distribution.

This section takes up the challenge of applying the algorithm developed in Section 2 to a model with a large number of parameters in a situation where it is not clear that it would be superior to a conventional MCMC algorithm. The important finding in this section is that the generic approach set forth in Section 2 works well, with some minor modification. As by-products it produces accurate evaluations of marginal likelihood as well as the evolution of the posterior distribution through the sample.

5.1 Model and data

A vector autoregression specifies the distribution of a K-dimensional time series $\{y_t\}$ conditional on its history:

$$y_t = c + \sum_{s=1}^{S} A_s y_{t-s} + \varepsilon_t; \quad \varepsilon_t \overset{iid}{\sim} N(0, \Sigma). \tag{26}$$

The matrices $A_s = [a_{kis}]$ are $K \times K$ and altogether the model has $K(1 + KS) + K(K+1)/2$ parameters. In the application here $K = 7$ and $S = 4$. The model admits the possibility of nonstationarity, and therefore the distribution of y_t $(t = 1, \ldots, T)$ conditions on S presample observations $y_{1-S}, \ldots, y_0$. Because the same lagged values of y_t appear on the right side of all K equations in (26) the likelihood function is that of a multivariate regression model, which is bounded above.

The prior distribution is a modification of the "Minnesota prior" proposed in Doan et al. (1984) and Litterman (1986) with two components. The prior distribution of Σ is Wishart,

$$\Sigma \sim W_K(D, \nu), \tag{27}$$

where D is a $K \times K$ diagonal matrix and ν is the degrees of freedom parameter. The prior distribution of the $K\,(1 + KS)$ coefficients, conditional on Σ, is comprised of the independent components

$$c_k \sim N\left(0, \kappa^2 \sigma_{kk}\right) \qquad (k = 1, \ldots, K) \tag{28}$$

$$a_{kks} \sim N\left(0, (\lambda/s)^2\right) \qquad (s = 1, \ldots, S;\ k = 1, \ldots, K) \tag{29}$$

$$a_{kis} \sim N\left(0, (\gamma\lambda/s)^2 \cdot (\sigma_{kk}/\sigma_{ii})\right) \qquad (k \neq i). \tag{30}$$

The conditional variances of the coefficients in (29)–(30) specify that effects of lagged covariates decay with time. Consistent with prior distributions employed in the literature and in practice, the intercept tightness hyperparameter in (28) is $\kappa = 1$, the overall tightness hyperparameter in (29)–(30) is $\lambda = 0.2$, and the relative tightness hyperparameter in (30) is $\gamma = 0.7$. These hyperparameters are typical of those used in applications of this model (Doan et al., 1984; Litterman, 1986; Geweke and Whiteman, 2006, Section 4). In (27) $\nu = 30$ and D/ν is a diagonal matrix with entries detailed below.

The algorithm operates on the vector of transformed parameters θ. The diagonal elements of the variance matrix Σ are mapped to $\frac{1}{2}\log(\sigma_{kk})$ $(k = 1, \ldots, K)$ and the off-diagonals are mapped to $\operatorname{atanh}(\sigma_{ki}/\sqrt{\sigma_{kk}\sigma_{ii}})$ $(k = 2, \ldots, K;\ i = 1, \ldots, k-1)$. The coefficients require no transformation. The algorithm requires the prior density evaluated with respect to θ. This is a straightforward calculation given the prior density with respect to the model parameters and the Jacobian of the transformation. Since $p(y_{1:t} \mid \theta)$ is the likelihood function of a multivariate regression model, evaluation of it is SIMD-compatible, as are the relatively minor computations involved in the transformation from θ to Σ.

A direct implementation of the algorithm described in Section 2.4 involves a random walk sampler over the 231-dimensional parameter vector θ in the M phase. We have found it more efficient to subdivide the parameter vector into several components and then subject each component, in turn, to a random walk Metropolis update, the variance of each component being computed in the usual way from the JN particles. For the results reported here the parameter vector was divided into eight components: the coefficients in each of the seven equations of the model, and (as the eighth component) the parameters of the variance matrix Σ. There is nothing especially important about this particular division into components; other choices may have performed just as well.

The composition of y_t is conventional for applications to the United States economy, following Smets and Wouters (2007): the logarithms of per capita real consumption, investment and gross domestic product (GDP); the logarithm of per capita weekly hours worked; the logarithms of the GDP deflator (price index) and the average real wage; and the federal funds interest rate. The data are quarterly, extending from 1951:1 through 2010:3; $t = 1$ corresponds to 1951:1 and $t = T = 239$ corresponds to 2010:3. All series except per capita hours worked and the federal funds rate are first differenced. Figure 9 presents the data from four of the time series. Inflation peaked in 1980:4 ($t = 120$), the "dot com bubble" peaked in 2000:1 ($t = 197$) and burst late in that quarter, and 2008:4–2009:2 ($t = 232$–234) marked the depth of the global financial crisis. Volatility

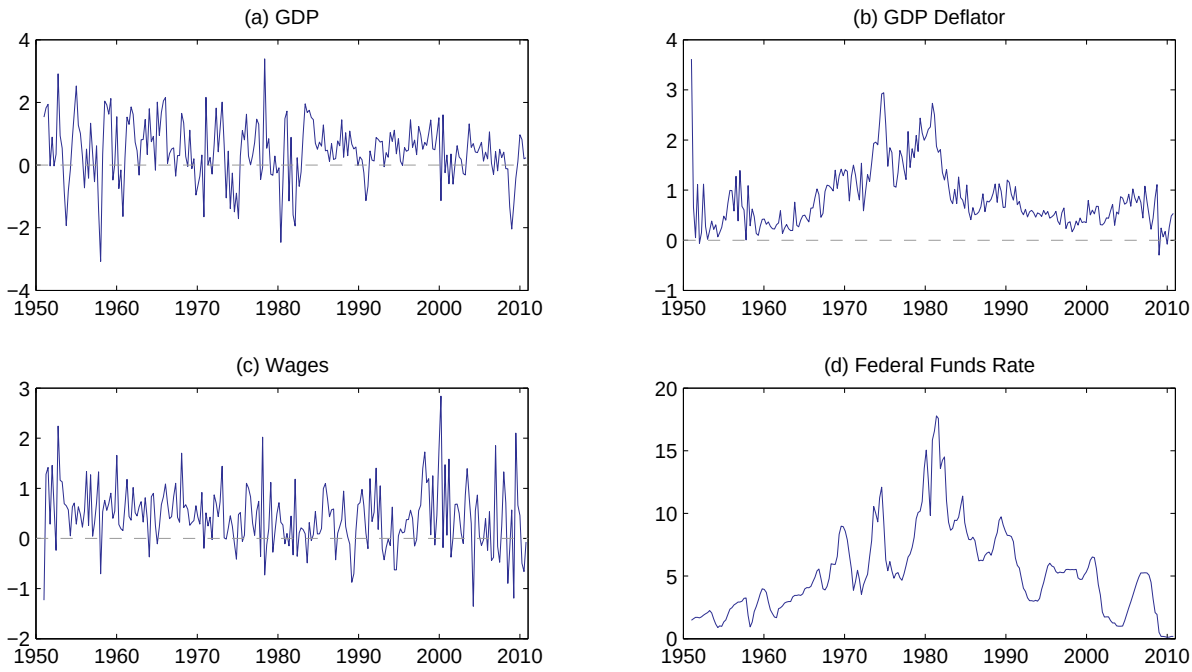

Figure 9: Data from 1951:1 to 2010:3 for four of the time series used in the VAR application: (a) first differences of log real GDP; (b) first differences of log GDP deflator; (c) first differences of log average real wage; and (d) federal funds rate.

of most of these series was notably lower in the period from about 1985:1 ($t = 137$) to about 2007:4 ($t = 228$) known to macroeconomists as "the great moderation." From inspection of similar graphs for all elements of y_t, we set the elements of the diagonal matrix D/ν in (27) to $(1, 9, 1, 1, 1, 1, 1)$.

5.2 Performance

All of the inference for the VAR model is based on $2^{16} = 65,536$ particles in $J = 2^4 = 16$ groups of $N = 2^{12} = 4,096$ particles each, with $D_1 = 0.5$ and $D_2 = 0.2$. Table 10 compares compute time and algorithm performance in computing log marginal likelihood and log score for some alternative choices of R. The log score is the numerical approximation of $\log p\left(y_{61:339} \mid y_{1:60}\right)$; observation $t = 61$ corresponds to 1966:1. For all values of R, the number of cycles is large compared to the sample size ($T = 239$), indicating rapid deterioration in effective sample size as observations are introduced in the C phase. Furthermore, $ESS/(NJ)$ often drops below the threshold $D_2 = 0.2$, especially in the early part of the sample, triggering additional Metropolis steps in the subsequent M phase. In general effective sample size is less adversely affected later in the sample and in quarters (like most of those in the great moderation) in which the time series exhibit less volatility. Figure 10 shows $ESS/(NJ)$ at each observation date for the case $R = 55$. That run of the algorithm involved $L = 137$ cycles (of which the

Table 10: Sensitivity to number of Metropolis steps R in phase M (VAR model)

Metropolis steps R	Compute time (seconds)	Cycles L	Log Marginal Likelihood	Numerical Standard Error	Log Score	Numerical Standard Error
5	1645	127	-1556.37	1.20	-1059.79	0.78
8	2779	134	-1537.94	1.08	-1058.19	0.52
13	4546	135	-1535.21	0.61	-1058.69	0.38
21	7387	136	-1531.37	0.32	-1059.05	0.20
34	12,047	136	-1528.12	0.36	-1058.68	0.24
55	19,402	137	-1527.17	0.29	-1059.18	0.15
89	31,344	137	-1526.93	0.32	-1059.16	0.22
$J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$						

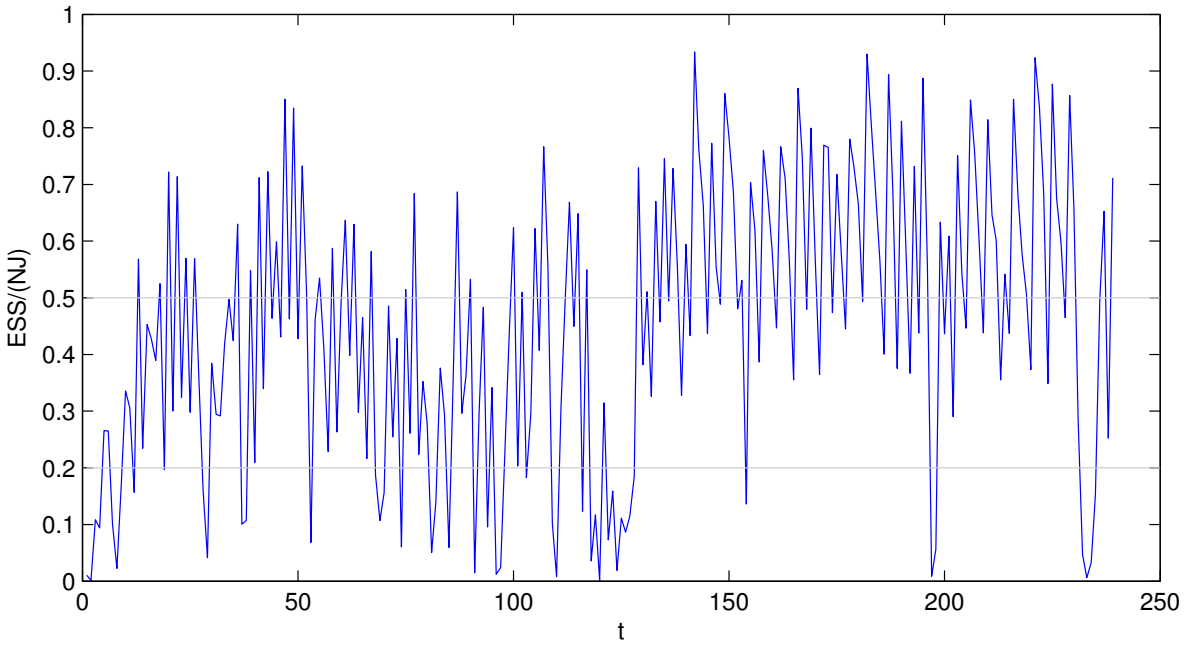

Figure 10: Effective sample size in the C phase of the algorithm in the VAR application. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$.

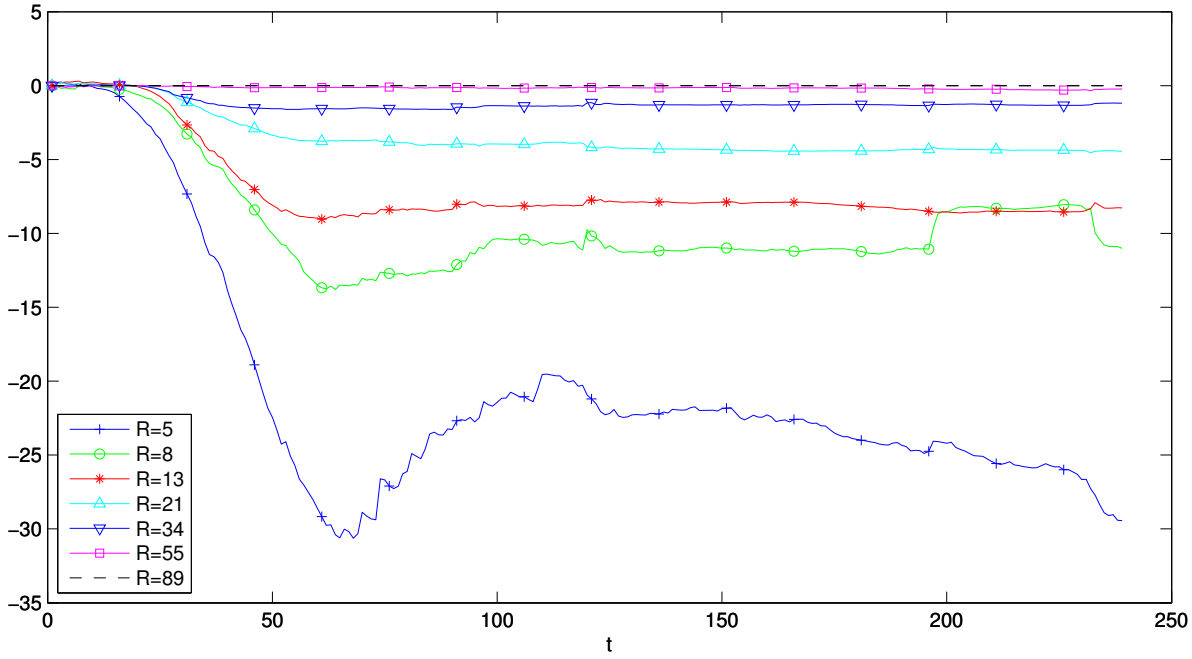

Figure 11: Numerical approximations of $\log p\,(y_{1:t})$ for various values of R, relative to $R = 89$. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$.

threshold $ESS/(NJ) < D_2$ was hit 46 times) and a total of 12,595 Metropolis iterations.

Log score and log marginal likelihood approximations do not achieve the accuracy here that they did in the EGARCH or MNP models. In view of the large number of parameters involved in this model, this is not surprising. For log score the numerical approximations reported in Table 10 are mutually consistent for all values of R, as indicated by the associated numerical standard errors. Numerical standard errors of log marginal likelihoods approximations are half again to twice as large as those of log scores. The approximations themselves increase monotonically with R, and the discrepancies for small values of R are dramatic both absolutely and in comparison with *NSE*.

Figures 11–13 provide more perspective on the evident unreliability in the log marginal likelihood approximations. Figure 11 compares the approximations of $\log p\,(y_{1t})$ for each value of R with those for $R = 89$. Most of the discrepancy arises early in the sample, though for $R = 5$ and $R = 8$ there are important changes later in the sample as well. For the choice $R = 55$, Figure 12 exhibits the approximations of $\log p\,(y_t \mid y_{1:t-1})$ and Figure 13 shows the corresponding *NSE*'s. The approximations and the *NSE*'s show a strong inverse relationship. Small values of $\log p\,(y_t \mid y_{1:t-1})$, with attendant high *NSE*'s, occur early in the sample and again during quarters when there were significant macroeconomic shocks: $t = 120$ (peak inflation), $t = 197$ (burst of the dot-com bubble), and $t = 232$–234 (depth of the global financial crisis).

This behavior is driven by the fact that the Monte Carlo sampling distribution of the function of interest $g\,(\theta) = p\,(y_t \mid y_{1:t-1}, \theta)$ is positively skewed when only a few of the

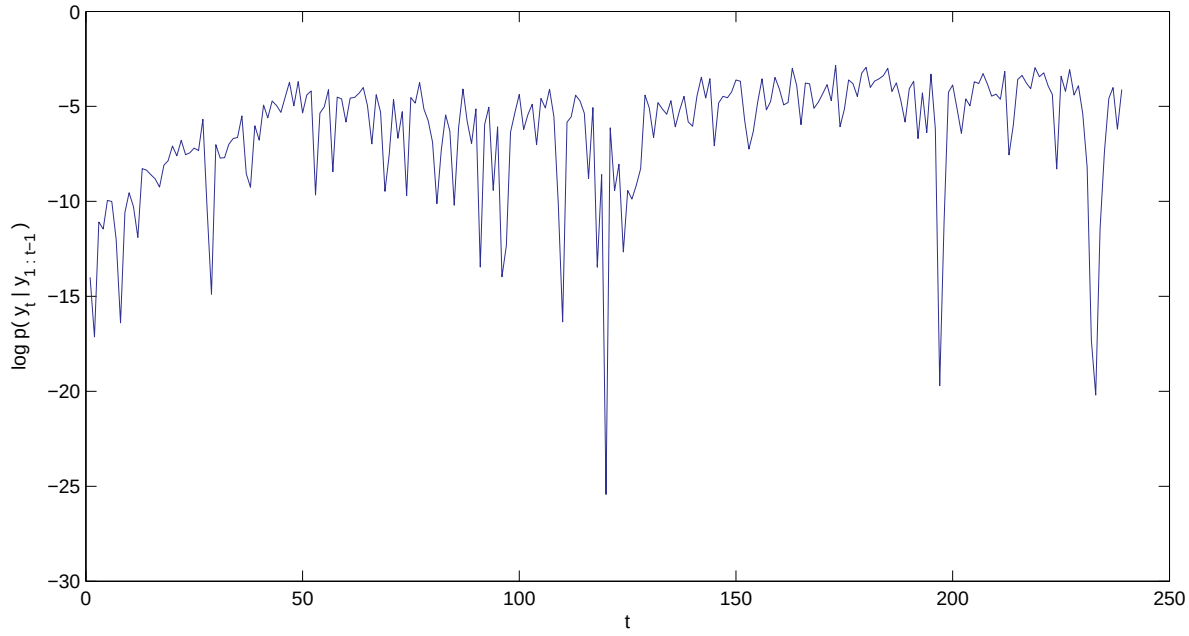

Figure 12: Log predictive likelihoods $\log p\left(y_t \mid y_{1:t-1}\right)$. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$.

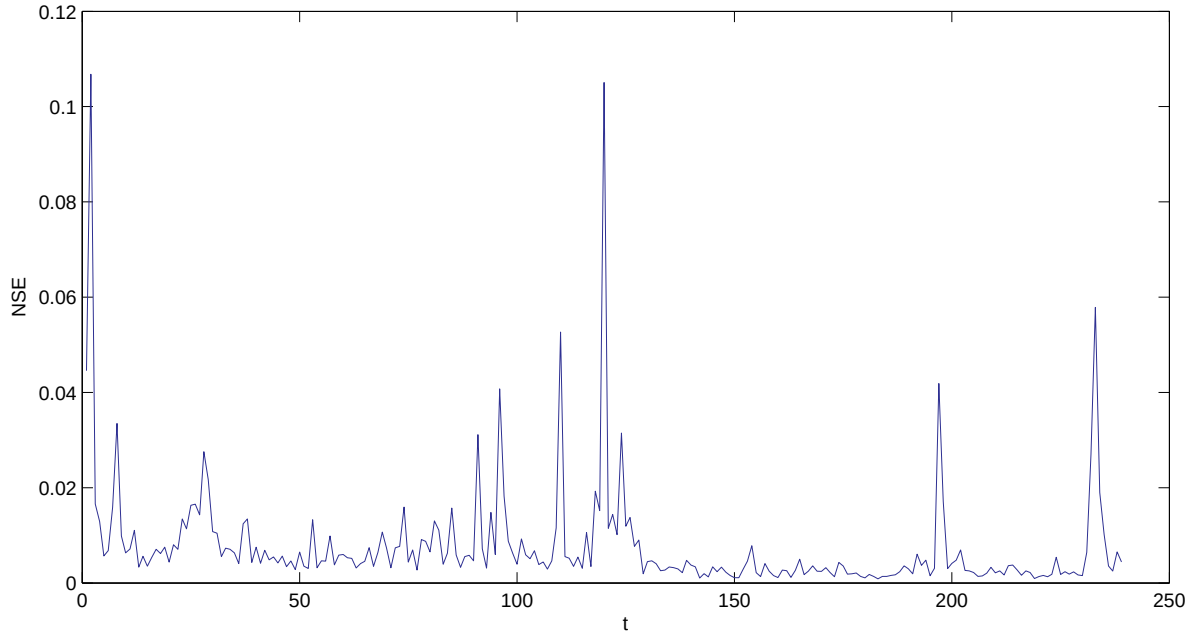

Figure 13: Numerical standard error of log predictive likelihoods $\log p\left(y_t \mid y_{1:t-1}\right)$. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$.

points in the Monte Carlo sample are drawn from that part of Θ accounting for most of the mass of the kernel $g(\theta)$. It is the same phenomenon noted for the EGARCH model in assessing tail probabilities of asset returns in Section 3.3. The problem is mitigated by increasing R because this also increases effective sample size. The problem is greater when sample size is smaller because of greater entropy in the posterior distribution, an effect exacerbated by high-dimensional parameter spaces like the one in this model. It also emerges for outlying observations y_t, which (by definition) are most plausible conditional on parameter values θ that are not characteristic of the posterior distribution and therefore represented by fewer particles. It is perhaps worth emphasizing that in these circumstances Monte Carlo moment approximations of probability are still unbiased, but they are more likely to be too low than to be too high.

The prescription for applied work is straightforward. A trial run of the algorithm with a large value of R and a small value of T, together with a spreadsheet indicating approximations of $\log p(y_{1:t})$ and their numerical standard error as a function of r and t, will clearly indicate both the extent of the problem and choices of R that will reduce the problem to a manageable level. Inspection of these results will also indicate situations in which increasing the number of particles JN is necessary or preferable to increasing R. Including these procedures in generic software for Bayesian inference should be straightforward.

5.3 Some specific aspects of the application

The sequence of log predictive likelihoods $\{\log p(y_t \mid y_{1:t-1})\}$ embodies the record of the model in predicting the events that actually transpired. It is central in both conventional Bayesian model averaging and in model pooling to optimize log predictive score (Geweke and Amisano, 2011). The numerical approximation of these log scores at each observation date t is shown in Figure 12. Since the predictive distribution in this model is unimodal, higher values of $\log p(y_t \mid y_{1:t-1})$ indicate realizations y_t that are closer to the mode, whereas realizations y_t in the tails of the distribution lead to low values of $\log p(y_t \mid y_{1:t-1})$. The most remarkable of these are associated with the events previously noted with respect to Figure 9: peak inflation, and the associated response in the Federal funds rate, in 1980:4 ($t = 120$); the end of the dot-com boom in 2001:1 ($t = 197$); and the global financial crisis 2008:4–2009:2 ($t = 232$–234). In such periods it is also the case that $p(\theta \mid y_{1:t-1})$ provides a poor source distribution for sampling $p(y_t \mid y_{1:t-1}, \theta)$, and this is evident in the numerical standard errors for $\log p(y_t \mid y_{1:t-1})$ provided in Figure 13. Conversely accuracy is highest in tranquil periods. For reasons discussed in the previous section approximation error is also high when sample size is small.

The implications for the accuracy of log marginal likelihood through the sample are not immediate from Figures 12 and 13, because approximation errors in successive evaluations of $\log p(y_t \mid y_{1:t-1})$ are not independent. Figure 14 shows all of these intermediate log marginal likelihoods, together with ± 10 numerical standard error bands. Consistent with the observations and analysis of Section 5.2, most of the error accumulates early in the sample. Much more important is controlling for the possibility that approximation

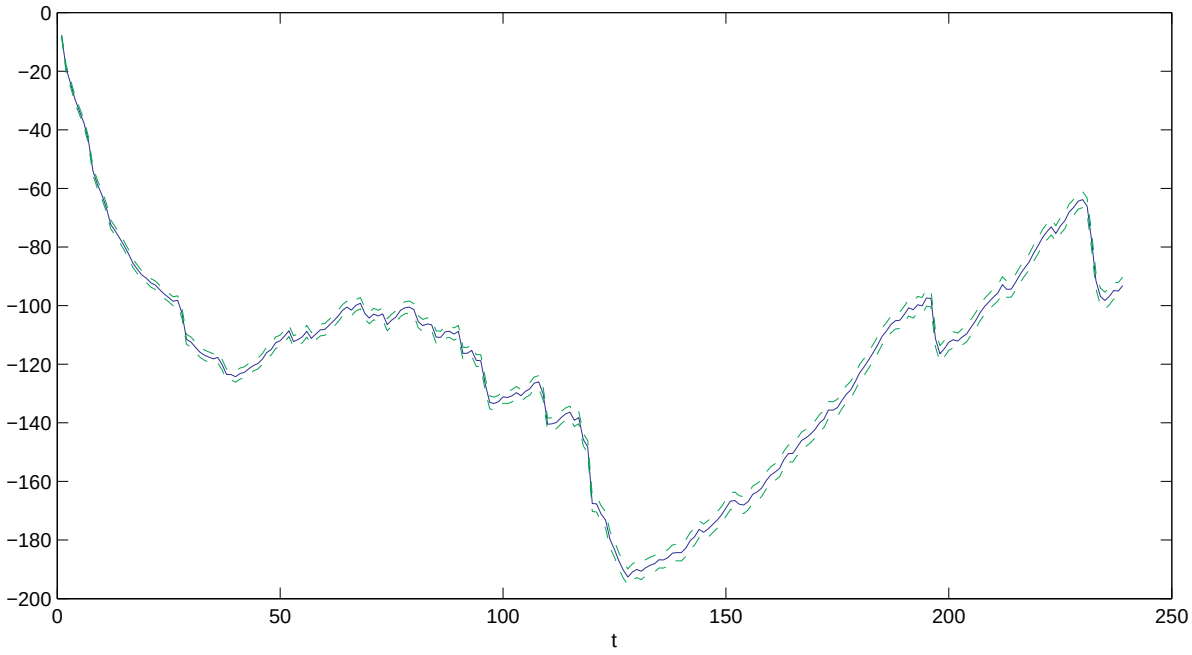

Figure 14: Numerical approximations of $\log p\left(y_{1:t}\right) + 6t$ (solid line) and bands of ± 10 numerical standard errors (dashed lines). $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$

errors for $\log p\left(y_t \mid y_{1:t-1}\right)$ may be systematically negative early in the sample for small R, as discussed in the previous section.

The many individual coefficients in a vector autoregression—the elements of the matrices A_s and the vector c in (26)—are not directly of interest in most applications of the model. Instead, the dynamic behavior implied by the model is often summarized in terms of the response of the observable time series vector y_t to changes in the vector of shocks to the system, ε_t. Following standard practice, we rewrite (26) in the form

$$y_t = c + \sum_{s=1}^{S} A_s y_{t-s} + P\eta_t, \tag{31}$$

where P is the lower triangular Cholesky factor of $\Sigma = PP'$ and $\eta_t \overset{iid}{\sim} N\left(0, I_K\right)$. For each $j = 1, \ldots, K$, let the shock $\eta_0^{(j)}$ be the K-dimensional vector with a one in the jth component and zeros elsewhere; let $f_r^{(j)}$ $(r = -S, \ldots, -1)$ be K-dimensional zero vectors; let $f_0^{(j)} = P\eta_0^{(j)}$; and let $f_r^{(j)} = \sum_{s=1}^{S} A_s f_{r-s}^{(j)}$ $(r = 1, 2, \ldots)$. The vector $f_r^{(j)}$ can be interpreted as the change in the conditional expectation of the vector y_{t+r} due to the shock $\eta_0^{(j)}$ at time t.

This creates a $K \times K$ array F_r of impulse response functions, $F_r = \left[\begin{array}{ccc} f_r^{(1)} & \cdots & f_r^{(K)} \end{array}\right] = \left[f_r^{(i,j)}\right]$. Using the Anderson (1984, Section 7.2) construction of the Wishart prior dis-

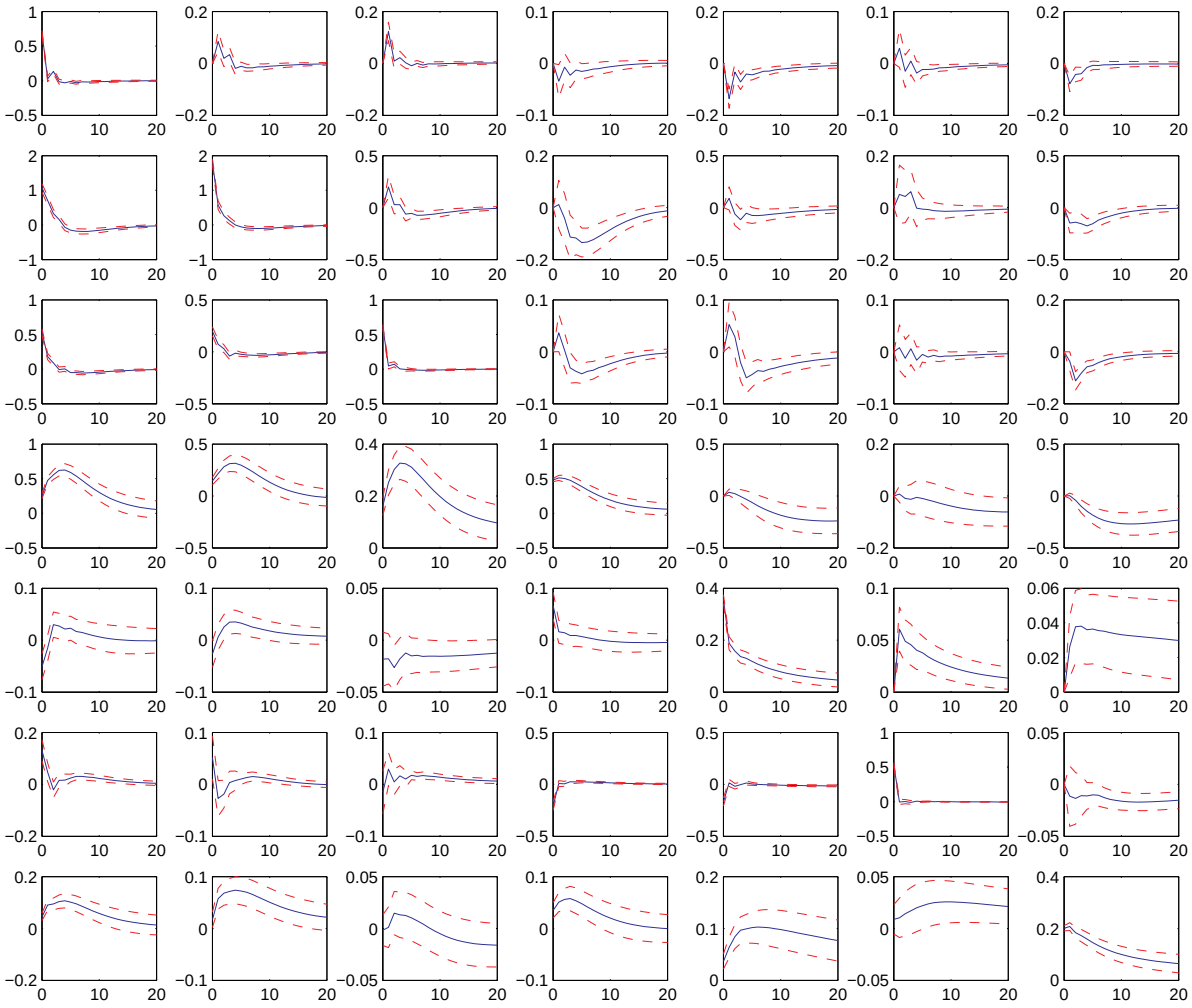

Figure 15: Matrix of impulse response functions. Solid line is posterior mean, dotted lines indicate ± one posterior standard deviation. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$, $R = 55$.

tribution of Σ in (27) it is straightforward to show that the elements of F_r have finite prior means and variances. Since the likelihood function is bounded this condition also guarantees the existence of these moments in the posterior distribution.

In general impulse response functions are sensitive to the ordering of variables in y_t. In the example worked here the ordering is the one given at the end of Section 5: consumption, investment, output, weekly hours worked, inflation, the real wage and the federal funds rate. Figure 15 shows the posterior means of these functions for $r = 0, \ldots, 20$ together with posterior standard deviation bands. Bands for ± 10 numerical standard errors are indistinguishable from the posterior means on the scale of the figure and are not shown. We present these results here because they are characteristic of the highly nonlinear functions of interest in applications of this model.

Table 11: Posterior moments in the VAR model

Metropolis steps R	Compute time (seconds)	Functions of interest $\frac{1}{2}\log\sigma_{22}$ Mean	*NSE*	*RNE*	$\frac{1}{2}\log\sigma_{55}$ Mean	*NSE*	*RNE*
5	1645	0.7436	0.0017	0.0097	-0.9855	0.0022	0.0078
8	2779	0.7444	0.0009	0.0341	-0.9879	0.0011	0.0304
13	4546	0.7438	0.0006	0.0966	-0.9892	0.0005	0.1703
21	7387	0.7441	0.0002	0.6766	-0.9883	0.0003	0.3333
34	12,047	0.7441	0.0002	0.7308	-0.9883	0.0002	0.6081
55	19,402	0.7440	0.0001	1.3608	-0.9883	0.0003	0.4763
89	31,344	0.7442	0.0002	0.9563	-0.9886	0.0001	1.7344

Mean is posterior mean; *NSE* is numerical standard error; *RNE* is relative numerical efficiency. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$.

Table 12: Posterior moments in the VAR model

Metropolis steps R	Compute time (seconds)	Functions of interest $f_4^{(2,2)}$ Mean	*NSE*	*RNE*	$f_4^{(5,5)}$ Mean	*NSE*	*RNE*
5	1645	-0.0258	0.0026	0.0160	0.1313	0.0007	0.0208
8	2779	-0.0098	0.0020	0.0277	0.1308	0.0005	0.0364
13	4546	-0.0033	0.0012	0.0802	0.1307	0.0002	0.1663
21	7387	-0.0036	0.0006	0.3652	0.1303	0.0001	0.5363
34	12,047	-0.0027	0.0006	0.3372	0.1304	0.0001	0.4196
55	19,402	-0.0016	0.0004	0.8635	0.1303	0.0001	1.4848
89	31,344	-0.0024	0.0004	0.6257	0.1305	0.0001	1.1166

Mean is posterior mean; *NSE* is numerical standard error; *RNE* is relative numerical efficiency. $J = 16$, $N = 4096$, $D_1 = 0.5$, $D_2 = 0.2$.

Tables 11 and 12 provide more analytical detail on the success of the algorithm in providing reliable numerical approximations to posterior moments of this kind. The results shown pertain to four moments. The first two (Table 11) indicate the scale of shocks $\left(\frac{1}{2}\log\sigma_{ii}\right)$ in the original parameterization (26) of the model. The last two (Table 12) are the values of impulse response functions of variables to their own shocks at a one-year ($r = 4$) horizon. These functions were chosen from among the many that we examined and indicate the range of numerical accuracy and reliability of numerical standard errors for posterior moments typically of interest.

In interpreting these tables it is important to bear in mind, as discussed in Sections 2.3 and 3.2, that reported *NSE* is a rough approximation to the true standard error of approximation due to the fact that it is based on comparisons across $J = 16$ groups of particles. Nevertheless, in the case of the moments $\frac{1}{2}\log\sigma_{ii}$ there are no indications of any problems. Discrepancies in the approximation of posterior mean, across different choices of R, are all within the range that might be anticipated given *NSE*. Relative numerical efficiency is consistent with decreasing dependence among the particles as R increases, and with independence for $R = 89$. Over the range of R displayed in Table 11 the return in accuracy to the additional compute time for increasing R is greater than it would have been pursuing the alternative of increasing the number of particles.

In the case of the impulse response functions, Table 12, there is evidence of some unreliability in numerical standard errors for $R = 5$ and $R = 8$. However the problems are not of the order of magnitude noted in Section 5.2 for log marginal likelihood. As emphasized in Section 5.2, those evaluations reflect the reliability of numerical standard error in all samples, back to $t = 1$, whereas the results here pertain entirely to the full sample $T = 239$. Decreasing dependence among the particles with increasing R is again evident, as is the conclusion that increases in R over the range shown are preferable to the alternative of increasing the number of particles.

6 Conclusion

This paper constructs an algorithm for Bayesian inference that is generic, requiring no special adaptation for a wide class of models and applications; is robust to irregular posterior distributions; is much faster than existing approaches to posterior simulation; is very accurate, because of the number of simulations feasible with massively parallel desktop computing; provides reliable assessment of the degree of accuracy attained; and provides marginal likelihoods and model specification diagnostics as by-products. It does this, in the main, by combining "off the shelf" technologies of several kinds: hardware devices that make possible massively parallel desktop computing at reasonable cost; software that makes it convenient to write computationally efficient code for these devices; sequential Monte Carlo algorithms that combine particle filtering and Markov chain Monte Carlo; and the comparatively simple asymptotics of independent chains for assessing the accuracy of approximation. There are many alternative ways these technologies could have been combined, and the one presented here is the out-

come of much experimentation with the theory and wide-ranging applications. These experiments demonstrated to us the importance of adaptive sequential Monte Carlo in any algorithm that would be both generic and practical; others have come to the same conclusion, e.g. Chopin et al. (2011). Limit theory for single chains of adaptive sequential Monte Carlo is difficult, but the ability to execute sequential Monte Carlo on large and independent groups of particles provides an alternative approach that immediately extends the theory to much more flexible adaptations. That is the sole strictly methodological contribution of this paper.

We believe that this approach, and improvements on this approach, have far-reaching implications for applied statistics. With the algorithm developed in this paper, the fundamental integration problems of Bayesian statistics are easier to solve than are the fundamental optimization problems of non-Bayesian statistics. For a wide class of models, bridging the gap between formulating a new model and having reliable code that meets the practical requirements of the practicing statistician becomes a routine matter much less demanding of long development time and specialized talent. The main step is to produce code that simulates from the prior distribution, evaluates the prior density, and evaluates the data density conditional on parameter values. The density function evaluation code must be written for device cores; existing and widely available software makes this a task that can be delegated to many graduate students and scientific programmers. We suggest augmenting this software with code that simulates from the distribution of observable data conditional on parameters, typically a simple task. This allows the investigator to study prior predictive distributions that in turn can provide comprehensive understanding of the entire model, including the prior distribution (Box, 1980; Lancaster, 2004, Section 2.4.2; Geweke, 2005, Section 8.3.1).

For the applied Bayesian statistician the essential barriers to entry for this technology are modest: one or more graphics processing units, at several hundred dollars per unit; and the aforementioned code, which can be written efficiently using the CUDA extension of the C programming language as well as a growing number of higher-level scientific programming languages. For some, the most significant barrier will be the requirement of a proper prior distribution that is central to the algorithm. Sequential Monte Carlo algorithms replicate Bayesian learning beginning from a proper prior distribution. This leads to robustness for the same reason that Bayesian learning has been successful in so many applications, even when it has had to proceed in disguise, as documented in McGrayne (2011). The ease of using prior predictive distributions in this setting, and the large practical returns to massively parallel sequential Monte Carlo for Bayesian inference, should ease this transition for those who have not yet made it.

The algorithm provided in this paper is more than a proof of concept: the examples in Sections 3, 4 and 5 show that it can be applied directly to models at the level of sophistication found in state-of-the-art substantive research. At the same time further study and experimentation in various dimensions can undoubtedly improve it. One such dimension is the degree of flexibility in the algorithm. Examples here include adapting the number of Metropolis steps in the M phase to performance, and using more general proposal distributions in both the C and M phases, both on our research

agenda. There are very significant categories, such as hierarchical models and state space models, to which the algorithm presented in this paper can be modified generically and to substantial practical advantage, and these modifications are also on the research agenda. An important strategic issue in this research is identifying particular categories of models and generic approaches that provide orders of magnitude improvement that will have practical benefits for many applications. We hope that the results presented here will encourage these lines of research.

References

Amdahl G (1967). Validity of the single processor approach to achieving large-scale computing capabilities. AFIPS Conference Proceedings 30: 483–485.

Anderson TW (1984). An Introduction to Multivariate Statistical Analysis. New York: Wiley.

Andrieu C, Doucet A, Holenstein A (2010). Particle Markov chain Monte Carlo. Journal of the Royal Statistical Society, Series B 72: 1–33.

Baker JE (1985). Adaptive selection methods for genetic algorithms. In: Grefenstette J (ed.), Proceedings of the International Conference on Genetic Algorithms and Their Applications, 101–111. Malwah NJ: Erlbaum.

Baker JE (1987). Reducing bias and inefficiency in the selection algorithm. In: Grefenstette J (ed.) Genetic Algorithms and Their Applications, 14–21. New York: Wiley.

Berkowitz, J (2001). Testing density forecasts with applications to risk management. Journal of Business and Economic Statistics 19: 465–474.

Börsch-Supan A, Hajivassiliou V (1993). Smooth unbiased multivariate probability simulators for maximum likelihood estimation of limited dependent variable models. Journal of Econometrics 58: 347–368.

Box GEP (1980). Sampling and Bayes' inference in scientific modeling. Journal of the Royal Statistical Society, Series A 143: 383–430.

Carpenter J, Clifford P, Fearnhead P (1999). Improved particle filter for nonlinear problems. IEEE Proceedings – Radar Sonar and Navigation 146: 2–7.

Celeux G, Hurn M, Robert CP (2000). Computational and inferential difficulties with mixture posterior distributions. Journal of the American Statistical Association 95: 957–970.

Chen CF (1985). On asymptotic normality of limiting density functions with Bayesian implications. Journal of the Royal Statistical Society, Series B 47: 540–546.

Chopin N (2002). A sequential particle filter method for static models. Biometrika 89: 539–551.

Chopin N (2004). Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. Annals of Statistics 32: 2385–2411.

Chopin N, Jacob P (2010). Free energy sequential Monte Carlo, application to mixture modelling. In: Bernardo JM, Bayarri MJ, Berger JO, Dawid AP, Heckerman D, Smith AFM, West M (eds.), Bayesian Statistics 9. Oxford: Oxford University Press.

Chopin N, Jacob PI, Papaspiliopoulis O (2011). SMC^2: A sequential Monte Carlo algorithm with particle Markov chain Monte Carlo updates. Working paper. http://arxiv.org/PS_cache/arxiv/pdf/1101/1101.1528v2.pdf

Del Moral P, Doucet A, Jasra A (2006). Sequential Monte Carlo samplers. Journal of the Royal Statistical Society, Series B 68: 411–436.

Del Moral P, Doucet A, Jasra A (2011). On adaptive resampling strategies for sequential Monte Carlo methods. Bernoulli, forthcoming.

Diebold FX, Gunther TA, Tay AS. (1998). Evaluating density forecasts with applications to financial risk management. International Economic Review 39: 863–883.

Doan T, Litterman RB, Sims CA (1984). Forecasting and conditional projection using realistic prior distributions. Econometric Reviews 3: 1–100.

Douc R, Moulines E (2008). Limit theorems for weighted samples with applications to sequential Monte Carlo methods. The Annals of Statistics 36: 2344–2376.

Doucet A, Godsil S, Andrieu C (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing 10: 197–208.

Duan JC, Fulop A (2011). Marginalized sequential Monte Carlo samplers. Working paper. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1837772

Durham G, Geweke J (2011). Improving asset price prediction when all models are false. Working paper. http://www.censoc.uts.edu.au/pdfs/geweke_papers/gp_working_5b.pdf

Fearnhead P (2002). Markov chain Monte Carlo, sufficient statistics, and particle filters. Journal of Computational and Graphical Statistics 11: 848–862.

Flegal JM, Jones GL (2010). Batch means and spectral variance estimates in Markov chain Monte Carlo. Annals of Statistics 38: 1034–1070.

Flury T, Shephard N (2008). Bayesian inference based only on simulated likelihood: Particle filter analysis of dynamic economic models. Forthcoming, Econometric Theory. http://economics.ouls.ox.ac.uk/15222/

Fruhwirth-Schnatter S (2001). Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. Journal of the American Statistical Association 96: 194–209.

Fulop A, Li J (2011). Robust and efficient learning: A marginalized resample-move approach. Working paper. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1724203

Gelfand AE, Smith AFM (1990). Sampling-based Approaches to Calculating Marginal Densities. Journal of the American Statistical Association 85: 398–409.

Geweke J (1989). Bayesian inference in econometric models using Monte Carlo integration. Econometrica 57: 1317–1340.

Geweke J (2005). Contemporary Bayesian Econometrics and Statistics. Englewood Cliffs NJ: Wiley.

Geweke J (2007). Interpretation and inference in mixture models: simple MCMC works. Computational Statistics and Data Analysis 51: 3529–3550.

Geweke J, Amisano G (2010). Comparing and evaluating Bayesian predictive distributions of asset returns. International Journal of Forecasting 26: 216–230.

Geweke J, Amisano G (2011). Optimal prediction pools. Journal of Econometrics 164: 130–141.

Geweke J, Keane MP (2001). Computationally intensive methods for integration in econometrics. In: Heckman JJ, Leamer EE (eds.), Handbook of Econometrics volume 5. Amsterdam: North-Holland, 3463–3568.

Geweke J, Keane MP, Runkle D (1993). Alternative computational approaches to inference in the multinomial probit model. Review of Economics and Statistics 76: 609–632.

Geweke J, Keane MP, Runkle D (1997). Statistical inference in the multinomial mjultiperiod probit model. Journal of Econometrics 80: 125–165.

Geweke J, Whiteman C (2006). Bayesian forecasting. In: Elliott G, Granger CWJ, Timmermann A (eds.), Handbook of Economic Forecasting. Amsterdam: Elsevier. Chapter 1, 3–80.

Gilks WR, Berzuini C (2001). Following a moving target – Monte Carlo inference for dynamic Bayesian models. Journal of the Royal Statistical Society, Series B 63: 127–146.

Gordon NG, Salmond DG, Smith AFM (1993). A novel approach to non-linear and non-Gaussian Bayesian state estimation. IEEE Proceedings F: Radar and Signal Processing 140: 107–113.

Hajivassiliou V, McFadden D, Ruud P (1996). Simulation of multivariate normal rectangle probabilities and their derivatives: Theoretical and computational results. Journal of Econometrics 72: 85–134.

Hendeby G, Karlsson R, Gustafsson F (2010). Particle filtering: The Need for Speed. EURASIP Journal on Advances in Signal Processing doi:10.1155/2010/181403

Innovative Computing Laboratory (ICL), University of Tennessee (2011). Matrix Algebra for GPU and Multicore Architectures. http://icl.cs.utk.edu/magma/

Jasra A, Doucet A, Stephens DA, Holmes CC (2007a). Interacting sequential Monte Carlo samplers for transdimensional simulation. Computational Statistics and Data Analysis 52: 1765–1791.

Jasra A. Stephens DA, Holmes CC (2007b). On population-based simulation for static inference. Statistics and Computing 17: 263–279.

Kloek T, van Dijk HK (1978). Bayesian estimates of equation system parameters: An application of integration by Monte Carlo. Econometrica 46:1–19.

Kong A, Liu JS, Wong WH (1994). Sequential imputations and Bayesian missing data problems. Journal of the American Statistical Association 89:278–288.

Koopman SJ, Shephard N, Creal D (2009). Testing the assumptions behind importance sampling. Journal of Econometrics 149: 2–11.

Künsch HR (2005). Recursive Monte Carlo filters: Algorithms and theoretical analysis. The Annals of Statistics 33: 1983–2021.

Lancaster T (2004). An Introduction to Modern Bayesian Econometrics. Malden MA: Blackwell Publishing.

Lee L, Yau C, Giles MB, Doucet A, Homes CC (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo Methods. Journal of Computational and Graphical Statistics 19: 769–789.

Lindley DV, Smith AFM (1972). Bayes estimates for the linear model. Journal of the Royal Statistical Society, Series B 55: 837–847.

Litterman RB (1986). Forecasting with Bayesian vector autoregressions: Five years of experience. Journal of Business and Economic Statistics 4: 25–38.

Liu JS, Chen R (1995). Blind deconvolution via sequential imputations. Journal of the American Statistical Association 90: 567–576.

Liu JS, Chen R (1998). Sequential Monte Carlo methods for dynamic systems. Journal of the American Statistical Association 93: 1032–1044.

Liu J, West M (2001). Combined parameters and state estimation in simulation-based filtering. In: Doucet A, de Freitas N, Gordon, N (eds.), Sequential Monte Carlo methods in practice. New York: Springer.

Matlab (2011). Matlab Parallel Computing Toolbox. http://www.mathworks.com /products/parallel-computing/description5.html

McGrayne SHB (2011). The Theory that Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunded Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy. New Haven: Yale University Press.

Metropolis N, Rosenbluth MN, Rosenbluth MN, Teller AH, Teller E (1953). Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21: 1087–1092.

Nelson DB (1991). Conditional heteroskedasticity in asset returns: A new approach. Econometrica 59: 347–370.

Nvidia (2011). Nvidia CUDA C Programming Guide Version 4.0. http://developer .download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C/ Programming_Guide.pdf

Primiceri GE (2005). Time varying structural autoregressions and monetary policy. Review of Economic Studies 72: 821–852.

Rosenblatt, M (1952). Remarks on a multivariate transformation. Annals of Mathematical Statistics 23: 470–472.

Rossi PE, Allenby GM, McCulloch R (2005). Bayesian Statistics and Marketing. Englewood Cliffs NJ: Wiley.

Rubin DB (1988). Using the SIR algorithm to simulate posterior distributions. In: Bernardo JM, DeGroot MH, Lindley DV, Smith AFM (eds.), Bayesian Statistics 3, 395-402. Oxford: Oxford University Press.

Sims CA (1980). Macroeconomics and reality. Econometrica 48: 1–48.

Shephard N, Pitt MK (1997). Likelihood analysis of non-Gaussian measurement time series. Biometrika 84: 653–667.

Smets F, Wouters R (2007). Shocks and frictions in US business cycles: A Bayesian DSGE approach. American Economic Review 97: 586–606.

Smith, JQ (1985). Diagnostic checks of non-standard time series models. Journal of Forecasting 4: 283–291.

Suchard MA, Wang Q, Chan C, Frelinger J, Cron A, West M (2010). Understanding GPU programming for statistical computation: Studies in Massively parallel massive mixtures. Journal of Computational Graphics and Statistics 19: 419–438.

Sweeting TJ, Adekola AO (1987). Asymptotic posterior normality for stochastic processes revisited. Journal of the Royal Statistical Society, Series B 49: 215–222.

Whitley D (1994). A genetic algorithm tutorial. Statistics and Computing 4:65–85.

Zellner A (1971). An Introduction to Bayesian Inference in Econometrics. New York: Wiley.