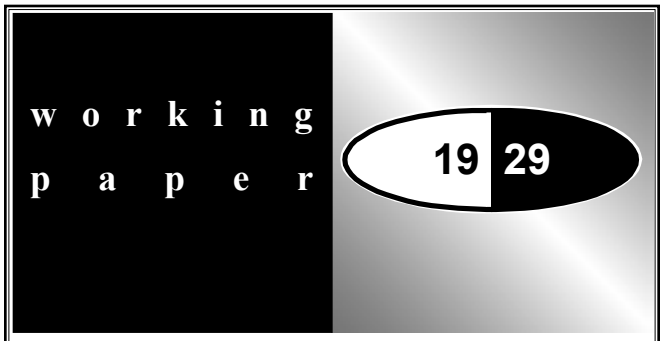

Sequential Bayesian Inference for Vector Autoregressions with Stochastic Volatility

Mark Bognanni and John Zito

FEDERAL RESERVE BANK OF CLEVELAND

ISSN: 2573-7953

Working Paper 19-29 December 2019

Sequential Bayesian Inference for Vector Autoregressions with Stochastic Volatility

Mark Bognanni and John Zito

We develop a sequential Monte Carlo (SMC) algorithm for Bayesian inference in vector autoregressions with stochastic volatility (VAR-SV). The algorithm builds particle approximations to the sequence of the model's posteriors, adapting the particles from one approximation to the next as the window of available data expands. The parallelizability of the algorithm's computations allows the adaptations to occur rapidly. Our particular algorithm exploits the ability to marginalize many parameters from the posterior analytically and embeds a known Markov chain Monte Carlo (MCMC) algorithm for the model as an effective mutation kernel for fighting particle degeneracy. We show that, relative to using MCMC alone, our algorithm increases the precision of inference while reducing computing time by an order of magnitude when estimating a medium-scale VAR-SV model.

JEL Codes: C11, C32, C51, E17.

Keywords: Vector autoregressions, stochastic volatility, sequential Monte Carlo, particle filter, Rao-Blackwellization.

Suggested citation: Bognanni, Mark, and John Zito. 2019. "Sequential Bayesian Inference for Vector Autoregressions with Stochastic Volatility." Federal Reserve Bank of Cleveland, Working Paper no. 19-29. https://doi.org/10.26509/frbc-wp-201929.

Mark Bognanni is at the Federal Reserve Bank of Cleveland (mark.j.bognanni@gmail.com). John Zito is at Rice University.

1. Introduction

Over the last 40 years the vector autoregression (VAR) has become a benchmark tool in empirical macroeconomics. Among the various extensions to the VAR, stochastic volatility (SV) has perhaps been the most robustly successful at improving forecasting and model fit. The demonstrated preference of standard macroeconomic time-series for including some form of SV in the VAR has been evident at least since Primiceri (2005) and Sims and Zha (2006).[1] In a particularly thorough recent example, Clark and Ravazzolo (2015) show that a wide variety of stochastic volatility specifications robustly improve VAR forecasting performance.[2] However, while the gains from including stochastic volatility in the VAR are now well documented, so too are the challenges of VAR-SV model estimation. Most practitioners analyze the model from a Bayesian perspective, but, unlike with a constant-parameter VAR, the Bayesian posterior distribution of the VAR-SV cannot be fully characterized analytically. Simulation-based computational methods are then the key tool for posterior inference.

This paper's primary contribution is to develop a parallelizable algorithm for sequential Bayesian estimation of VAR-SV models. By "sequential" we mean that when new data are incorporated into the model estimates, an approximation to the new posterior adapts the approximation to the old posterior. Importantly, our proposed approach remains fully Bayesian in the sense that inference is based on the model's full likelihood function and the algorithm can estimate posterior moments arbitrarily accurately in the limit of certain algorithm settings. The upshot of our estimation algorithm is that, given an approximation of the posterior at time t, the approximation of the posterior at time $t + 1$ can be obtained an order of magnitude faster than under the extant approach in the literature. We demonstrate the effectiveness of our estimation approach by applying it to a seven-variable "medium scale" VAR-SV, in which our sequential algorithm yields more precise estimates than when using the extant Markov chain Monte Carlo (MCMC) algorithm alone, while also dramatically decreasing computation time.

Our estimation method is fundamentally a sequential Monte Carlo (SMC)

[1] In the context of regime-switching models like Sims and Zha (2006), also see Sims, Waggoner, and Zha (2008) and Bognanni and Herbst (2017).

[2] Also see Clark (2011) and Carriero, Clark, and Marcellino (2016) for additional results on the forecasting gains from adding stochastic volatility to a VAR, as well as Chan and Eisenstat (2018) for similar conclusions about model fit.

algorithm. Sequential Monte Carlo algorithms are particle-based methods that approximate a sequence of densities of interest with discrete approximations based on weighted samples. In our setting, the relevant densities to approximate are the sequence of VAR-SV posteriors as the available time-series of data expands. An important practical aspect of SMC, and one that we heavily exploit, is that the most computationally intensive parts of the algorithm can be executed in parallel.

Our SMC implementation leverages the known structure of the VAR-SV posterior in two important ways in order to tackle the notoriously difficult problem of simultaneous sequential inference for both static parameters (the linear VAR coefficients) and dynamic parameters (latent volatility states). First, when updating the particles from one stage to the next, we use the fact that one can analytically marginalize the static parameters, at which point we are effectively executing the algorithm in a parameter space of reduced dimension while still accounting for the posterior of the static parameters.[3] Second, when rejuvenating particle diversity, a stage of SMC called "mutation," we use a known MCMC algorithm specifically tailored to the VAR-SV. The MCMC algorithm is of a relatively efficient variety known as a Gibbs sampler, which makes our algorithm considerably more effective at rejuvenating meaningful particle diversity compared to more naive alternatives. The potential utility of such approaches has long been discussed within the SMC literature, in which SMC and MCMC are sometimes referred to as "complementary."[4] In practice, however, SMC algorithms used in the literature are usually implemented with only relatively naive mutation kernels. Our paper shows, in a practical and empirically relevant example, the usefulness of folding a relatively more effective MCMC kernel into SMC when such a kernel is available.

To contextualize our contribution, it is helpful to understand the standard method of Bayesian inference for the VAR-SV. To date, fully Bayesian inference in the VAR-SV is conducted by means of simulating random samples from a Markov chain Monte Carlo (MCMC) algorithm. The key strength of the MCMC algorithm is that it "works" in the formal sense that, asymptotically in the number of iterations of the Markov chain, posterior moments of interest can be estimated

[3] One might just as well say that we "Rao-Blackwellize" the static parameters.

[4] For example, see the discussion in the introductory section of Del Moral, Doucet, and Jasra (2006).

arbitrarily accurately. Indeed this fact motivates our inclusion of the MCMC algorithm into our approach as the mutation kernel. However, MCMC also has two key weaknesses that our SMC algorithm addresses. The first weakness is that MCMC samples must be generated serially. This is a simple consequence of the fact that the distribution of the i^{th} random sample from a Markov chain conditions on the $(i-1)^{th}$ random sample. Hence, the MCMC algorithm cannot be cleanly parallelized, and the only way to increase its estimation accuracy is to simply let it run longer. In contrast, key steps of our SMC algorithm can be executed in parallel. This means the SMC algorithm's accuracy can be improved by making use of additional CPUs *while keeping the runtime fixed*.

The second weakness is that any change to the information set, such as the arrival of a new data point, alters the joint posterior distribution of all model parameters, but the MCMC algorithm has no notion of an incremental update to existing posterior samples. Rather the MCMC algorithm must simply be rerun from scratch using the new data set.[5] In a production environment, the model parameters would ideally always be estimated from the most recent, and most complete, information set to yield the best possible forecasts and analysis. Hence, it would be a substantial nuisance that a satisfactory run of the MCMC algorithm for the seven-variable model takes three hours. In contrast, the key benefit of our algorithm is that it can rapidly update estimates from a previous period.

Lastly, this paper also contributes by demonstrating the practical viability of using publicly available resources to implement high performance computing tasks. This is important because our algorithm performs best when using computing hardware that few researchers have physically available in their offices. However, our implementation uses only resources, including the computing hardware, that are readily accessible to every researcher with an internet connection. To be more specific, the key features of the computational environment we use

[5]This annoyance has been observed since, and served as a motivator for, the early work on sequential algorithms for Bayesian inference. For the moment letting H_t denote a posterior sample of a model's parameters based on information through time t, Berzuini et al. (1997) write, "When computing the new sample H_{t+1}, it seems sensible to try to use information contained in the available sample H_t. Under conventional MCMC sampling, this is not possible; with each new data item, the available sample of parameter values must be discarded, and a new sample must be created by restarting the MCMC from scratch on the entire model. This waste of information causes responses to new data to become slow. In particular, it hampers application of the method in real-time contexts."

are threefold: 1) the computer hardware was remotely provisioned from a public cloud service, namely, Amazon Web Services; 2) our programs are written in the `Julia` language, which is open source and distributed under the MIT License; and 3) the parallelism is implemented using only "high level" `Julia` commands.

Algorithm in Context of the Literature. With regard to the estimation algorithm, our paper clearly builds off of the vast literatures on SMC and particle filters. One can find excellent overviews of SMC methods in Doucet, de Freitas, and Gordon (2001) and Doucet and Johansen (2011). With discussions more targeted to the interests of economists, Creal (2012) and Herbst and Schorfheide (2015) also provide thorough overviews. Our algorithm is also closely related to the sequentially adaptive Bayesian learning (SABL) algorithm, which is also based on SMC, used in Durham and Geweke (2014) and Durham et al. (2019).

Somewhat more specifically, our interest in sequential estimation of both static and dynamic unobservables is related to the work of Storvik (2002), Fearnhead (2002), and Djurić and Miguez (2002), all of which are also aimed at circumventing the poor performance of a more naive SMC approach of simply augmenting the state vector with the unknown static parameters and use a particle filter to estimate the augmented state. Such an approach is known to degenerate rapidly.[6] Our approach differs from these by integrating Rao-Blackwellization into a key stage of the algorithm.

From here the rest of the paper proceeds as follows. In Section 2 we introduce the VAR-SV model and describe some of its key analytical properties. In Section 3 we describe our sequential Monte Carlo. In Section 4 we apply our estimation method to a seven-variable VAR-SV and rigorously document the algorithm's performance relative to using MCMC alone. In Section 5 we describe some additional details of our computational environment. In Section 6 we conclude.

[6] Another attempt to address this is from Kitagawa (1998) and Liu and West (2001), who introduced artificial dynamics for the parameters. While lessening the degeneracy problem, this approach changes the structure of the model and requires careful application-specific tuning to work well. The recent SMC^2 approach of Chopin, Jacob, and Papaspiliopoulos (2013) uses nested SMC algorithms to jointly estimate parameters and states. Unlike the aforementioned "state-augmentation" approaches, SMC^2 does not suffer as badly from degeneracy, and is asymptotically valid. For more on particle-based approaches to joint state and parameter inference, see Kantas et al. (2015).

2. Bayesian VAR-SV Model

The VAR-SV model stipulates that an $n \times 1$ vector of time-series data $\mathbf{y}_t$ evolves according to

$$\mathbf{y}_t' = \sum_{\ell=1}^{p} \mathbf{y}_{t-\ell}' \mathbf{B}_{(\ell)} + \mathbf{c} + \mathbf{u}_t' \,, \quad \mathbf{u}_t \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma}_t) \,, \quad \text{for } 1 \leq t \leq T, \tag{1}$$

where each $\mathbf{B}_{(\ell)}$ is an $n \times n$ matrix, $\mathbf{c}$ is a length n vector, $\mathbf{u}_t$ is a length n vector, and $\mathbf{\Sigma}_t$ is an $n \times n$ time-varying symmetric, positive-definite covariance matrix of the model's mean-zero forecast errors. Equation (1) can be written more compactly by defining $\mathbf{B} = [\mathbf{B}_{(1)}', \ldots, \mathbf{B}_{(p)}', \mathbf{c}']'$ and $\mathbf{x}_t = [\mathbf{y}_{t-1}', \ldots, \mathbf{y}_{t-p}', 1]'$, and writing

$$\mathbf{y}_t' = \mathbf{x}_t' \mathbf{B} + \mathbf{u}_t' \quad , \quad \mathbf{u}_t \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma}_t) \quad , \quad \text{for } 1 \leq t \leq T \tag{2}$$

where $\mathbf{x}_t$ is $m \times 1$ for $m = np + 1$. Regardless of the details of how $\mathbf{\Sigma}_t$ evolves, the conditional density for the observation $\mathbf{y}_t$ takes the form

$$p(\mathbf{y}_t | \mathbf{B}, \mathbf{x}_t, \mathbf{\Sigma}_t) = \mathrm{N}(\mathbf{y}_t | \mathbf{B}' \mathbf{x}_t, \mathbf{\Sigma}_t) \tag{3}$$

and the density of the full sequence $\mathbf{y}_{1:T}$, given $\mathbf{B}$ and the sequence $\mathbf{\Sigma}_{0:T}$, is $p(\mathbf{y}_{1:T} | \mathbf{B}, \mathbf{\Sigma}_{1:T}) = \prod_{t=1}^{T} p(\mathbf{y}_t | \mathbf{B}, \mathbf{x}_t, \mathbf{\Sigma}_t)$. We define $\mathbf{b} = \mathrm{vec}(\mathbf{B})$ and refer to $\mathbf{b}$ and $\mathbf{B}$ interchangeably when including them in a conditioning set of information.

In this paper we consider models for which $\mathbf{\Sigma}_t$ is parameterized as in Primiceri (2005), that is

$$\mathbf{\Sigma}_t = \mathbf{A}_t^{-1} \mathbf{\Lambda}_t \mathbf{A}_t^{-1'} \tag{4}$$

with

$$\mathbf{A}_t = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ a_{2,1,t} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n,1,t} & \cdots & a_{n,n-1,t} & 1 \end{bmatrix}, \; \mathbf{\Lambda}_t = \begin{bmatrix} \exp(v_{1,t}) & 0 & \cdots & 0 \\ 0 & \exp(v_{2,t}) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \exp(v_{n,t}) \end{bmatrix}. \tag{5}$$

It will be useful to have notation to refer to all of the free elements of $\mathbf{A}_t$ and $\mathbf{\Lambda}_t$. To that end, let $\mathbf{a}_{i,t}$ be the vector containing the free elements of the i-th row of $\mathbf{A}_t$ so,

for example, $\mathbf{a}_{1,t}$ is empty and in general $\mathbf{a}_{i,t} = [a_{i,1,t}, \ldots, a_{i,i-1,t}]'$. We then let $\mathbf{a}_t$ be the vector that concatenates all of the $\mathbf{a}_{i,t}$ vectors so that $\mathbf{a}_t \equiv [\mathbf{a}'_{2,t}, \mathbf{a}'_{3,t}, \ldots, \mathbf{a}'_{n,t}]'$. To collect all of the free parameters that pin down $\mathbf{\Sigma}_t$, we define the length $n_s \equiv n(n+1)/2$ vector $\mathbf{s}_t \equiv [\mathbf{v}'_t, \mathbf{a}'_t]'$ where $\mathbf{v}_t \equiv [v_{1,t}, \ldots, v_{n,t}]'$. Lastly, we let $f(\mathbf{s}_t)$ be the function that maps $\mathbf{s}_t$ to $\mathbf{\Sigma}_t$ via the transformations in equations (4) and (5).

Law of Motion for Latent States. The stochastic volatility piece of the model is completed by specifying a law of motion for $\mathbf{s}_t$. We allow each element of $\mathbf{s}_t$ to change over time according to an AR(1) process,

$$s_{i,t} = \beta_{i,0} + \beta_{i,1} s_{i,t-1} + \eta_{i,t}, \quad \eta_{i,t} \sim N\left(0, \sigma_i^2\right) \quad \text{for } 1 \leq t \leq T \tag{6}$$

where $s_{i,t}$ denotes the i-th element of $\mathbf{s}_t$. To collect the unobservables pertaining to the i-th state transition equation we define the vectors $\boldsymbol{\beta}_i \equiv \left[\beta_{i,0}, \beta_{i,1}\right]'$ and $\boldsymbol{\theta}_i \equiv \left[\boldsymbol{\beta}'_i, \sigma_i^2\right]'$, and finally the full set of static parameters governing all the state transitions is denoted $\boldsymbol{\theta} = \left\{\boldsymbol{\theta}_i\right\}_{i=1}^{n_s}$. The normally distributed innovations $\eta_{i,t}$ are assumed to be independently distributed across both i and t. The joint density for the sequence of all latent states, given initial conditions, then takes the form

$$p(\mathbf{s}_{1:T} | \boldsymbol{\theta}, \mathbf{s}_0) = \prod_{i=1}^{n_s} \prod_{t=1}^{T} p(s_{i,t} | \boldsymbol{\theta}_i, s_{i,t-1}) \tag{7}$$

Priors. To complete the definition of the VAR-SV as a Bayesian model requires prior distributions for the initial conditions of the latent states $\mathbf{s}_0$ and for the two sets of static parameters, $\mathbf{b}$ and $\boldsymbol{\theta}$. We assume that the joint prior $p(\mathbf{b}, \mathbf{s}_0, \boldsymbol{\theta})$ can be factored as

$$p(\mathbf{b}, \mathbf{s}_0, \boldsymbol{\theta}) = p(\mathbf{b}) \prod_{i=1}^{n_s} p(s_{i,0}) p(\boldsymbol{\theta}_i), \tag{8}$$

where $\mathbf{b} \sim \mathrm{N}\left(\overline{\mathbf{b}}_0, \mathbf{V}_0\right)$ and $s_{i,0} \sim \mathrm{N}\left(\overline{s}_{i,0}, \varsigma_{i,0}^2\right)$ for each i. The prior for each $\boldsymbol{\theta}_i$ takes the natural conjugate form for the AR(1) laws of motion specified in equation (6) but with the parameter space truncated to enforce nonexplosiveness of each $s_{i,t}$

process.[7] This is to say that, for $i = 1, \ldots, n_s$,

$$(9) \quad p(\boldsymbol{\beta}_i, \sigma_i^2) = p(\boldsymbol{\beta}_i | \sigma_i^2)\, p(\sigma_i^2) \propto \mathrm{N}\big(\boldsymbol{\beta}_i | \overline{\boldsymbol{\beta}}_i, \sigma_i^2 \Sigma_{i,\beta}\big)\, \mathrm{IG}\big(\sigma_i^2 | a_i, b_i\big) \cdot \mathbf{1}\big\{|\beta_{i,1}| \leq 1\big\}$$

where IG denotes the inverse gamma distribution and $\mathbf{1}\{\cdot\}$ denotes the indicator function.[8] The prior hyperparameters $\big(\overline{\mathbf{b}}_0, \mathbf{V}_0\big)$, $\big\{\overline{s}_{i,0}, \varsigma_{i,0}^2\big\}_{i=1}^{n_s}$, and $\big\{\overline{\beta}_i, \Sigma_{i,\beta}, a_i, b_i\big\}_{i=1}^{n_s}$ are objects specified by the researcher. Appendix B describes the way we set the prior hyperparameters for the application in Section 4.

2.1 Tractable Features of the Posterior and Predictive Distributions

The joint posterior of the VAR-SV's unobservables takes the form

$$(10) \qquad p(\mathbf{s}_{0:T}, \mathbf{b}, \boldsymbol{\theta} | \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} | \mathbf{b}, \mathbf{s}_{0:T})\, p(\mathbf{s}_{0:T} | \boldsymbol{\theta})\, p(\mathbf{b}) \prod_{i=1}^{n_s} p(s_{i,0}) p(\boldsymbol{\theta}_i)}{p(\mathbf{y}_{1:T})}.$$

It is the density in equation (10) that the researcher must characterize in order to proceed with Bayesian inference. Unfortunately, arbitrary moments of interest from the posterior distribution are not available analytically. However, a few particular analytical properties of the distribution are available in closed form and will be usefully incorporated into our estimation method. At a high level, one might summarize the analytically tractable aspects of the model as,"if, at time t, we knew the latent states $\mathbf{s}_{0:t}$ in addition to $\mathbf{y}_{1:t}$, then we would know everything." Two manifestations of that fact are particularly useful for us: first, the ability to simulate iid values from the predictive distribution of $\mathbf{s}_{t+1}$ given the history $\mathbf{s}_{0:t}$; second, the ability to evaluate the model's predictive density for $\mathbf{y}_{t+1}$, but marginal of $\mathbf{b}$, given $\mathbf{y}_{1:t}$ and values for $\mathbf{s}_{0:t+1}$.

Toward the first property, note that a valid way to generate simulations from each of the distributions $p(s_{i,t+1} | s_{i,0:t})$ is to simulate from the joint distribution $p(s_{i,t+1}, \boldsymbol{\theta}_i | s_{i,0:t})$ and then simply discard the draws of $\boldsymbol{\theta}_i$. To that end, note further that $p(s_{i,t+1}, \boldsymbol{\theta}_i | s_{i,0:t})$ can be factored into a marginal and a conditional as

$$(11) \qquad p(s_{i,t+1}, \boldsymbol{\theta}_i | s_{i,0:t}) = p(s_{i,t+1} | \boldsymbol{\theta}_i, s_{i,t}) p(\boldsymbol{\theta}_i | s_{i,0:t}).$$

[7] This restriction is standard in the literature. Even when researchers do not explicitly say so within the text, the restriction is typically imposed in their computer code.

[8] There are various parameterizations of the inverse gamma distribution used in the literature. Our parameters (a, b) pertain to an inverse gamma density of the form $p(x|a, b) \propto x^{-a-1} \exp(-b/x)$.

In principle, simulations from the predictive distribution of $s_{i,t+1}|s_{i,0:t}$ can then be produced by using the right-hand side of equation (11), that is by first simulating from the marginal for $\boldsymbol{\theta}_i$ and then conditioning on its value to simulate $s_{i,t+1}$.[9] Each piece required for that method is indeed tractable. The posterior distribution $p(\boldsymbol{\theta}_i|s_{i,0:t})$ takes the same functional form as equation (9) and thus draws from $p(\boldsymbol{\theta}|\mathbf{s}_{0:t})$ can be readily simulated.[10] Lastly, the sample from $p(s_{i,t+1}|\boldsymbol{\theta}_i, s_{i,t})$ is then produced by simply simulating forward equation (6).

Toward the second property, note that the posterior of $\mathbf{b}$ conditional on $\boldsymbol{\Sigma}_{1:T}$ is known in closed form as

$$p(\mathbf{b}|\mathbf{y}_{1:T}, \mathbf{s}_{0:T}) = \frac{p(\mathbf{y}_{1:T}|\mathbf{b}, \boldsymbol{\Sigma}_{1:T})\, p(\mathbf{b})}{p(\mathbf{y}_{1:T}|\boldsymbol{\Sigma}_{1:T})} = \mathrm{N}\big(\mathbf{b} \,|\, \overline{\mathbf{b}}_T, \mathbf{V}_T\big) \tag{12}$$

where the parameters of the posterior normal distribution are given by

$$\mathbf{V}_T = \left(\mathbf{V}_0^{-1} + \sum_{t=1}^{T} \left(\boldsymbol{\Sigma}_t^{-1} \otimes \mathbf{x}_t\mathbf{x}_t'\right)\right)^{-1} \tag{13}$$

$$\overline{\mathbf{b}}_T = \mathbf{V}_T\left(\mathbf{V}_0^{-1}\overline{\mathbf{b}}_0 + \sum_{t=1}^{T} \mathrm{vec}\left(\mathbf{x}_t\mathbf{y}_t'\boldsymbol{\Sigma}_t^{-1}\right)\right) \tag{14}$$

and $\boldsymbol{\Sigma}_t = f(\mathbf{s}_t)$ for each t. This known functional form plays an important role in the MCMC algorithm embedded within our SMC algorithm. It also leads to the result that when also conditioning on $\mathbf{s}_{t+1}$, the predictive density for $\mathbf{y}_{t+1}$ takes the form

$$p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \boldsymbol{\Sigma}_{1:t+1}) = \int_{\mathbf{b}} p(\mathbf{y}_{t+1}|\mathbf{y}_{1:t}, \mathbf{b}, \boldsymbol{\Sigma}_{1:t+1}) p(\mathbf{b}|\mathbf{y}_{1:t}, \boldsymbol{\Sigma}_{1:t}) d\mathbf{b} \tag{15}$$

$$= \mathrm{N}\big(\mathbf{y}_{t+1}|\overline{\mathbf{y}}_{t+1}, \boldsymbol{\Gamma}_{t+1}\big) \tag{16}$$

where

$$\overline{\mathbf{y}}_{t+1} = \overline{\mathbf{B}}_t'\mathbf{x}_{t+1} \qquad \text{and} \qquad \boldsymbol{\Gamma}_{t+1} = \boldsymbol{\Sigma}_{t+1} + \mathbf{X}_{t+1}'\mathbf{V}_t\mathbf{X}_{t+1} \tag{17}$$

[9] The two-step procedure is required because of the truncation on the space of $\boldsymbol{\theta}_i$. In the absence of the truncation, the predictive distribution of each $s_{i,t+1}$ is known in closed form as a Student's T-distribution.

[10] If nothing else, the truncation can then be implemented by a simple accept-reject algorithm, though more efficient approaches are available and we describe the one we use in the Appendix.

and $\mathbf{X}_{t+1} = \mathbf{I}_n \otimes \mathbf{x}_{t+1}$. Hence, knowledge of the latent states is also sufficient to fully characterize the model's predictive distribution analytically.

3. A Sequential Algorithm for Fully Bayesian Inference

One might inutitively expect that the posteriors for successive time periods would be, in some sense, similar. After all, the only change in the information set is typically a single vector of observations. The idea of sequential inference is to leverage the similarity of the posteriors in proximate periods and adapt the approximation of the old posterior into an approximation of the new posterior. The SMC algorithm we develop in this section builds this idea into an algorithm for which the larger SMC literature provides formal statements about the algorithm's ability to systematically estimate features of the posterior. We next describe our algorithm, as we implement it, and then follow with a discussion. Appendix C contains additional details about some of the algorithm's technical properties.

3.1 SMC for the VAR-SV

At a general level, SMC algorithms recursively construct discrete approximations to the distribution of the random variables of a sequence of target densities. In our setting, the sequence of target distributions consists of the sequence of joint posteriors for the VAR-SV's unobservables. Two key properties of SMC make it particularly attractive for our purposes. First, the discrete approximations constructed by SMC can be used for provably valid Bayesian inference about moments of interest, and the inference can be made arbitrarily accurate by increasing the granularity of the approximation. Second, we can build the approximation sequentially, adapting the approximation from one stage to the next rather than starting the inference from scratch at each stage.

SMC consists of an initialization stage followed by, for each target density, an iteration over three phases: correction, selection, and mutation. We denote the t-th target density as π_t so $\pi_t = p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$ for $t = 1, 2, \ldots, T$. The i-th point in the discrete approximation to π_t consists of a weight, $0 \leq W^i \leq 1$, and a vector of values for the unobservables, $\mathbf{s}^i_{0:t}$. We refer to the tuple $\left(\mathbf{s}^i_{0:t}, W^i\right)$ as a particle and to the collection of N particles $\left\{\mathbf{s}^i_{0:t}, W^i\right\}_{i=1}^N$ as a swarm, where $\sum_{i=1}^N W^i = 1$.

Algorithm 1 summarizes the structure of an SMC algorithm and we subsequently describe each stage in detail.

Algorithm 1 - SMC for VAR-SV

Initialize $\left\{\mathbf{s}_0^i, W_0^i\right\}_{i=1}^N$ *via* $\mathbf{s}_0^i \sim p(\mathbf{s}_0)$ *and* $W_0^i = \frac{1}{N}$ *for* $i = 1, \ldots, N$

for $t = 1, \ldots, T$

1. *Correction (Algorithm 2)*
   $\{\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i\}_{i=1}^N \leftarrow \text{correction}\left(\{\mathbf{s}_{0:t-1}^i, W_{t-1}^i\}_{i=1}^N, \mathbf{y}_{0:t}\right)$
2. *Selection (Algorithm 3)*
   $\{\hat{\mathbf{s}}_{0:t}^i, W_t^i\}_{i=1}^N \leftarrow \text{selection}\left(\{\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i\}_{i=1}^N\right)$
3. *Mutation (Algorithm 4)*
   $\{\mathbf{s}_{0:t}^i, W_t^i\}_{i=1}^N \leftarrow \text{mutation}\left(\{\hat{\mathbf{s}}_{0:t}^i, W_t^i\}_{i=1}^N, \mathbf{y}_{0:t}\right)$

end

Correction. The "Correction" phase updates the particles to correct for the discrepancy between the old target kernel k_{t-1} and the new target kernel k_t. In full generality, this is done by importance sampling $\pi_t(\mathbf{s}_{0:t}) = p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$. Importance sampling entails simulating values for $\mathbf{s}_{0:t}$ from a proposal distribution $g(\tilde{\mathbf{s}}_{0:t}^i|\mathbf{y}_{1:t})$ and reweighting the draws appropriately to account for the discrepancy between the proposal and target densities. In particular, defining a particle's *incremental* weight $\widetilde{w}_t^i$ as

$$\widetilde{w}_t^i = \frac{k_t(\mathbf{s}_{0:t}^i)}{k_{t-1}(\mathbf{s}_{0:t-1}^i)} \frac{1}{g(\mathbf{s}_{0:t}^i|\mathbf{y}_{0:t})}. \tag{18}$$

and constructing each particle's unnormalized and normalized weights, respectively, as

$$w_t^i = \widetilde{w}_t^i W_{t-1}^i \tag{19}$$

$$\tilde{W}_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}. \tag{20}$$

yields a particle approximation $\left\{\tilde{\mathbf{s}}^i_{0:t},\ \tilde{W}^i_t\right\}_{i=1}^N$ for which laws of large numbers and central limit theorems for moments of π_t obtain in the number of particles.

In our setting, the expression for the incremental weights in equation (18) can be simplified considerably. Defining the target kernel k_t as

$$k_t(\mathbf{s}_{0:t}) = p(\mathbf{s}_{0:t})p(\mathbf{y}_{1:t}|\mathbf{s}_{0:t}), \tag{21}$$

the ratio of target kernels required for the computation of the incremental weights $\widetilde{w}_t$ according to equation (18) becomes[11]

$$\frac{k_t(\mathbf{s}_{0:t})}{k_{t-1}(\mathbf{s}_{0:t-1})} = p(\mathbf{s}_t|\mathbf{s}_{0:t-1})p(\mathbf{y}_t|\mathbf{y}_{0:t-1}, \mathbf{s}_{0:t}). \tag{22}$$

Lastly, we need to specify a proposal distribution $g(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$. In practice, upon entering stage t, researchers usually propose $\mathbf{s}_{0:t}$ by keeping each particle's $\mathbf{s}^i_{0:t-1}$ fixed at their values from the end of stage $t-1$ and proposing a new value for only $\tilde{\mathbf{s}}^i_t$ from a source density $g_t(\tilde{\mathbf{s}}^i_t|\mathbf{s}^i_{0:t-1}, \mathbf{y}_{1:t})$.[12] It is common in particle filter applications to use the law of motion for the dynamic parameters as proposal distribution, in which case $g_t = p(\tilde{\mathbf{s}}^i_t|\mathbf{s}^i_{0:t-1})$ and the algorithm is known as the bootstrap particle filter. We follow that choice here. With the bootstrap choice of $g_t(\cdot)$, and using the expression in equation (22), the expression for the incremental weight in equation (18) reduces to simply

$$\widetilde{w}^i_t = p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{s}^i_{0:t-1}, \tilde{\mathbf{s}}^i_t). \tag{23}$$

The recipe for implementing the correction phase is then given by Algorithm 2.

Algorithm 2 - Correction

(parallel)for $i = 1, \ldots, N$

---

[11] The subsequent expression uses the fact that $p(\mathbf{y}_{1:t-1}|\mathbf{\Sigma}_{0:t-1}) = p(\mathbf{y}_{1:t-1}|\mathbf{\Sigma}_{0:t})$, hence

$$\frac{p(\mathbf{y}_{0:t}|\mathbf{\Sigma}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{\Sigma}_{0:t-1})} = \frac{p(\mathbf{y}_{0:t}|\mathbf{\Sigma}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{\Sigma}_{0:t})} = p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{\Sigma}_{0:t}).$$

[12] In this case one could say that g is still source density for the whole sequence but with a pointmass for $\tilde{\mathbf{s}}^i_{0:t-1}$ at the values $\mathbf{s}^i_{0:t-1}$.

1. *Sample* $\tilde{\mathbf{s}}_t^i \sim p(\tilde{\mathbf{s}}_t^i | \mathbf{s}_{0:t-1}^i)$.

2. *Compute incremental weight* $\widetilde{w}_t^i$ *via equation* (23).

3. *Compute unnormalized weight* w_t^i *via equation* (19).

end

- *Compute normalized weights* $\{\tilde{W}_t^i\}_{i=1}^N$ *via equation* (20).
- *The particle system is now given by* $\left\{\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i\right\}_{i=1}^N$

To recap the key ingredients required of our Correction recipe, we need to know how to simulate from the proposal distribution $p(\mathbf{s}_t|\mathbf{s}_{0:t-1})$ and how to evaluate pointwise $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{s}_{1:t})$, with both densities *marginal* of all static parameters. For the VAR-SV model of interest here, both tasks are feasible using the expressions in Section 2.1.

Selection. The selection phase consists of either resampling the particles or doing nothing. When resampling, we use multinomial resampling because SMC's theoretical properties are best understood with this choice.[13] This amounts to sampling iid, with replacement, from the set of trajectories $\{\tilde{\mathbf{s}}_{0:t}^i\}_{i=1}^N$, with the i^{th} trajectory being sampled with probability $\tilde{W}_t^i$. In Algorithm 3 we denote such a sample as $\hat{\mathbf{s}}_t^i \sim \{\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i\}_{i=1}^N$. We resample at stage t if the diversity of particles with meaningful weights falls below a particular threshold.[14] In particular, we follow a standard choice in the literature and resample if a summary statistic of particle degeneracy in the swarm, the "effective sample size" (ESS), falls below the threshold $N/2$. The recipe for implementing the selection phase is then given by Algorithm 3.

---

[13] In particular, multinomial resampling at the selection phase has facilitated the proof of central limit theorems for the particle approximation. In practice, alternative resampling strategies can give better performance. See Douc and Cappe (2005) and Murray, Lee, and Jacob (2016) for alternative resampling algorithms.

[14] This type of resampling rule is called "adaptive" because it depends on the current state of the particle approximation. Importantly, Del Moral, Doucet, and Jasra (2012) prove that, with adaptive resampling schemes such as this one, posterior moments computed from the particle system still converge to their true values asymptotically in N.

Algorithm 3 - Selection

- *Compute* $ESS = \left(\sum_{i=1}^{N} (\tilde{W}_t^i)^2 \right)^{-1}$.

if $(ESS/N) < 0.5$

for $i = 1, \ldots, N$

Sample $\hat{\mathbf{s}}_t^i \sim \{\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i\}_{i=1}^N$

end

Set $W_t^i = 1/N$ *for each* i.

else

Set $(\hat{\mathbf{s}}_{0:t}^i, W_t^i) = (\tilde{\mathbf{s}}_{0:t}^i, \tilde{W}_t^i)$ *for each* i.

end

- *The particle system is now given by* $\left\{\hat{\mathbf{s}}_{0:t}^i, W_t^i\right\}_{i=1}^N$.

Mutation. At a high level, the Mutation phase consists of "jittering" each particle's values for $\mathbf{s}_{0:t}^i$. This step is critical for combating the well-known issue of particle degeneracy, particularly when many identical copies of the same particle were created by resampling in the Selection phase.[15] Critically for the computational efficiency of the algorithm, the mutation steps occur on a per particle basis, where each particle is mutated independently of the others. Hence, the computationally intensive Mutation phase can be executed in parallel across particles.

We mutate each particle by iteratively sampling, n_{mut} times, its values of $\mathbf{s}_{0:t}$ from a known MCMC algorithm for the VAR-SV. We denote the MCMC kernel as K_t. The MCMC algorithm is, in most respects, not an innovation to this paper so we leave the details to Appendix E.[16] A key thing to note about our implementation of this stage is that, as an MCMC algorithm for the full

[15] Mutation steps are also sometimes known as "move" steps.

[16] The MCMC algorithm is essentially Algorithm 3 of Del Negro and Primiceri (2015), but with a time-invariant $\mathbf{b}$.

model, K_t generates samples of $(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta})$, while we only carry the sample of $\mathbf{s}_{0:t}$ into period $t+1$ as part of the particle. For our purposes, the samples of $(\mathbf{b}, \boldsymbol{\theta})$ are merely a byproduct of using an algorithm that operates in the full parameter space to efficiently move the values $\mathbf{s}_{0:t}$ around the target posterior. Hence, the samples of $(\mathbf{b}, \boldsymbol{\theta})$ are simply discarded after completing the mutation phase. Importantly, since our mutation kernel is a Gibbs sampler, it is extremely efficient at rejuvenating the diversity of $\mathbf{s}_{0:t}$ values. The recipe for implementing the mutation phase is given by Algorithm 4.

Algorithm 4 - Mutation

- *Let K_t be a Markov transition kernel with invariant distribution $p(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$.*

(parallel)for $i = 1, \ldots, N$

Set $\mathbf{s}_{0:t}^{i,(0)} = \hat{\mathbf{s}}_{0:t}^{i}$ *and sample* $(\mathbf{b}^{(0)}, \boldsymbol{\theta}^{(0)}) \sim p(\mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t}, \mathbf{s}_{0:t}^{i,(0)})$

for $j = 1, \ldots, n_{mut}$

Sample $\left(\mathbf{s}_{0:t}^{i,(j)}, \mathbf{b}^{(j)}, \boldsymbol{\theta}^{(j)}\right) \sim K_t\left(\mathbf{s}_{0:t}^{i,(j)}, \mathbf{b}^{(j)}, \boldsymbol{\theta}^{(j)}|\mathbf{s}_{0:t}^{i,(j-1)}, \mathbf{b}^{(j-1)}, \boldsymbol{\theta}^{(j-1)}\right)$.

end

Set $\mathbf{s}_{0:t}^{i} = \mathbf{s}_{0:t}^{i,(n_{mut})}$

end

- *The particle system is now given by* $\{\mathbf{s}_{0:t}^{i}, W_t^i\}_{i=1}^{N}$

3.2 Discussion

The algorithm we have presented leaves two aspects to be chosen by the researcher: the number of particles (N) and the number of mutation steps per phase (n_{mut}). In both respects, the most basic fact to be aware of is that "more is better." Importantly, Gilks and Berzuini (2001) and Chopin (2004) show that a central limit theorem continues to obtain for estimating moments of interest when using mutation steps in the SMC algorithm. In Appendix C we describe formally the sense in which our algorithm is equivalent to an SMC algorithm for the full parameter space.

Lastly, we note that in typical macroeconomic applications, a substantial set of observations are already available in the first estimation period of interest. In other words, updates based on one observation at a time will be necessary in the future, but the batch of observations $\mathbf{y}_{0:t}$ is available today. A reasonable approach to putting the SMC updates into practice would then be to generate a sample from the posterior up to the most recent available data using the MCMC algorithm, and subsequently using SMC to update the draws as subsequent observations become available The MCMC draws would be used as an equally weighted sample of π_t to initialize the SMC algorithm, from which Algorithm 1 would then proceed as usual when $\mathbf{y}_{t+1}$ becomes available.[17]

4. Application: Estimating a Seven-variable VAR-SV

In this section we demonstrate the effectiveness of our proposed estimation algorithm. Ideally, we would be able to compare our algorithm's estimation output to iid posterior draws. Unfortunately, this is not possible for the VAR-SV (else we would not have needed to write this paper at all). We can, however, compare our algorithm to the extant MCMC estimation method. When correctly specified, both SMC and MCMC yield samples of draws from which posterior moments can be consistently estimated. Hence, we compare SMC to MCMC in terms of their reliability, across repeated runs, at estimating features of the posterior. Lending further credence to the exercise, the correctness of the MCMC algorithm can be readily verified, and we have done so, using the "getting it right" test of Geweke (2004).[18]

As a setting for the comparison we estimate a VAR-SV with $n = 7$ variables, the same variables used in the "medium scale" system in Giannone, Lenza, and Primiceri (2015).[19] At a conceptual level, the variables in the VAR are real output (GDP), prices (P), consumption (C), investment (I), hours worked (H), wages (W), and interest rates (R). We give the details on the exact data series used for each object in Appendix A. The time-series consists of quarterly observations running from 1954:Q3 to 2019:Q1 and the VAR has $p = 4$ lags. This seven-variable

[17] Koop and Potter (2007) also suggest this type of hybrid approach in the context of estimating their "change point" models.

[18] See Appendix F.

[19] They are also the same variables, at least conceptually if not the precise choice of data series, used to inform the Smets and Wouters (2007) model.

system, with a year's worth of lags of each variable, is rich enough to capture many of the key macroeconomic dynamics of interest to policymakers while remaining parsimonious enough to not be too unwieldy.[20]

4.1 The Estimation Algorithms to Be Compared

We begin by describing the algorithms we compare. Standard practice in the literature is for researchers to work with a sample of 10,000 draws to approximate the posterior. Hence, we take a posterior approximation based on 10,000 values of the unobservables as the focal point for our comparisons.[21]

For MCMC, we consider two different ways to obtain the 10,000 draws.

1. **MCMC-short**. A "quick and dirty" MCMC chain of 10,000 iterations and preserving all of them.

2. **MCMC-long**. An MCMC chain of 105,000 iterations, burning the first 5,000 and thinning the remaining draws to 10,000 by keeping the sample from only every 10th iteration.

The MCMC-short algorithm is simply the fastest way to get 10,000 draws from the MCMC chain. We will see that, in terms of computing time, this is a highly relevant comparison for SMC's speed at producing a posterior update. The MCMC-long algorithm closely comports with what researchers do in practice when they wish to obtain an accurate posterior approximation.

For SMC we focus on algorithms with $n_{mut} = 5$ and $n_{mut} = 25$ while keeping the granularity of the approximation at N =10,000. We will sometimes refer to the two algorithms with the shorthand SMC-5 and SMC-25. We also compute the runtime for an SMC algorithm with no mutation steps at all, which, when compared to the other two specifications, facilitates the calculation of how much

[20] It is known in the literature that, for the stochastic volatility model described in Section 2, the order of the variables in $\mathbf{y}_t$ matters for inference. Relatedly, one of this paper's authors has research questioning the suitability for structural inference of models with this form of stochastic volatility; see Bognanni (2018). Nonetheless, models of this form have a strong track record in macroeconomic forecasting, as shown in the papers cited in our introduction, so we expect their estimation to be of interest for the foreseeable future. We use the variables in exactly the order in which they are listed above; in effect, this choice becomes part of the model. This is simply the order in which the variables appear, from left to right, in the Giannone, Lenza, and Primiceri (2015) replication files (though not exactly the order given in their Table 1).

[21] For the MCMC algorithms, each draw is of course equally weighted, while the SMC draws will potentially have different weights as determined by the algorithm described in Section 3.

TABLE I

COMPUTING TIME IN MINUTES TO ESTIMATE 7-VARIABLE VAR-SV POSTERIORS

Estimation Method	$p(\mathbf{s}_{0:t}\|\mathbf{y}_{1:t})\|p(\mathbf{s}_{0:t-1}\|\mathbf{y}_{1:t-1})$ $t = 100$	$t = T = 217$	$p(\mathbf{s}_{0:T}\|\mathbf{y}_{1:T})$	$\{p(\mathbf{s}_{0:t}\|\mathbf{y}_{1:t})\}_{t=1}^{T}$
SMC – $n_{mut} = 0$	0.2	0.3	41.9	41.9
SMC – $n_{mut} = 5$	1.0	2.2	247.5	247.5
SMC – $n_{mut} = 25$	4.5	9.5	1067.9	1067.9
MCMC – short	6.9	20.0	20.0	1838.1
MCMC – long	66.9	175.8	175.8	17606.1

Notes. SMC times are based on using 96 virtual CPUs. The $p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})|p(\mathbf{s}_{0:t-1}|\mathbf{y}_{1:t-1})$ columns give computing time to estimate the time t posterior given the estimates of the time $t-1$ posterior. The $p(\mathbf{s}_{0:T}|\mathbf{y}_{1:T})$ column gives the computing time to estimate the posterior for the full trajectory of $\mathbf{s}_{0:T}$ using all of the data but starting from scratch. The $\{p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})\}_{t=1}^{T}$ column gives the cumulative computing time required to estimate all of the posteriors for $\mathbf{s}_{0:t}$ with an expanding window of observations.

time the algorithm spends in the mutation phase.[22] The reasons for our choices for n_{mut} will be apparent after some discussion of the runtime results.

4.2 Estimation Algorithm Computation Times

In Table I we tabulate the runtime required for each algorithm to complete three different tasks of posterior inference. The first two columns of runtimes give the time required to approximate a particular posterior, $p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$, *conditional* on having available an approximation to the previous period's posterior, for which we use our swarm notation $p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$. The next column tabulates the runtime required to approximate the full posterior $p(\mathbf{s}_{0:T}|\mathbf{y}_{1:T})$, all the way to $t = T$, from scratch. The last column gives the runtime required to approximate all of the posteriors, starting with $t = 1$ and running through the end of the sample.

Table I makes clear a number of key points about the way the algorithms work. First, focusing on the two columns of runtimes for incremental updates, the SMC algorithms generate a new posterior approximation in an order of magnitude less time than the MCMC-long algorithm and still only half as long as the more crude MCMC-short algorithm. This is our single most important point in this section, but the runtimes in the last two columns give some additional clarity about how

[22] With $n_{mut} = 0$ the algorithm is equivalent to the "sequential importance sampling-resampling" (SISR) algorithm of Gordon, Salmond, and Smith (1993).

the algorithms differ.

Turning then to the third column of runtimes, one can see why the SMC incremental updates tabulated in the second column were so much faster than the MCMC algorithms. Namely, the MCMC algorithms have no notion of an incremental update at all. Rather, since the MCMC algorithms have to estimate the full target posterior from scratch, the MCMC "incremental" times for the T posterior are inherently identical to their full sample runtimes. However, for a single full sample estimation one can see that even the MCMC-long is considerably faster than the SMC algorithms. Hence, it is worth emphasizing that we do not claim to have a better method for estimating a single posterior.

The last column of runtimes makes clear that, although the SMC algorithm is slower to generate full sample estimates when starting from scratch, in doing so it is producing an approximation to all $t = 1, \ldots, T$ intervening posteriors. This point is made clear by the fact that the SMC algorithm runtimes to the last two columns are identical. This is an important point because it also summarizes the total computation time required over the life of the algorithm if it had been run "online" in a production setting from $t = 1$ onward. One can see that running the MCMC-long algorithm would have been more time consuming, by an order of magnitude, than SMC.

Lastly, we point out that for all of the algorithms considered here, increasing t increases the computing time. This is true for the MCMC algorithms (and thus the mutation phase of the SMC algorithms), since each iteration has to wash over an expanding sequence of $\mathbf{s}_{0:t}$. Also, and perhaps less obvious, it is true (though a minor consideration) for incremental updates for SMC even without any mutation steps. This is because the marginalization of $(\mathbf{b}, \boldsymbol{\theta})$ requires computing statistics that depend on the full sequences $\mathbf{y}_{1:t}$ and $\mathbf{s}_{0:t}$. Hence, as these sequences expand, the computation of those statistics takes longer. In this sense our algorithm is "sequential" but not "online" in the sense often meant by statisticians in which updates take a fixed amount of computation time.

4.3 High-Level Comparison of SMC and MCMC Estimates

We now document how effectively each algorithm approximates key posterior features. Before turning to the formal metrics, we begin with a high-level demonstration of "what's going on." Namely, when sized large enough, both al-

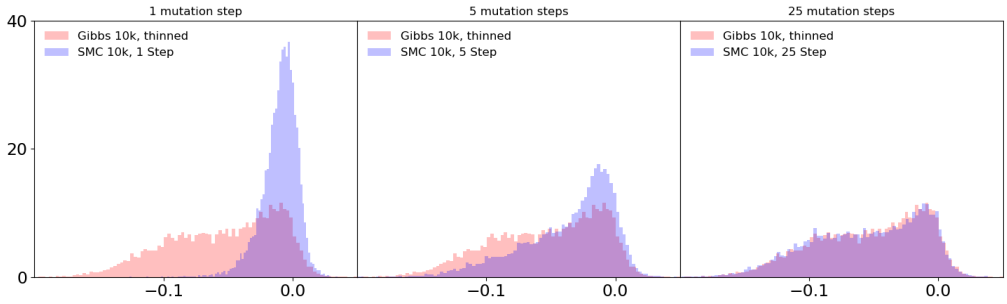

FIGURE 1.—$p(a_{5,4,t=180}|\mathbf{y}_{1:T})$ Estimates

gorithms yield identical posteriors. Of course the space of potential comparisons of posterior features is enormous. Each $\mathbf{s}_t$ has 28 free elements, and with 217 quarters of data (plus initial conditions) the full trajectory of $\mathbf{s}_{0:T}$ contains 6,104 distinct unobservables, any moment or cross-moment of which could form the basis of a posterior comparison. Here we focus on the features of the posterior of $a_{5,4,0:T}$ simply because it yields particularly clean visual comparisons among algorithms. In the next section we document the broader posterior features more rigorously and more thoroughly.

The three panels of Figure 1 show histograms of this posterior as estimated by SMC with 10,000 particles and 1, 5, and 25 mutation steps (columns). For comparison, each panel also shows a histogram of the estimated posterior from the MCMC-long algorithm. When the two histograms appear essentially on top of each other, the two algorithms have generated essentially the same posterior approximations.

Looking across the panels, from left to right, it is visually apparent that as we increase n_{mut} the SMC swarm characterizes the model's posterior increasingly accurately. The appendix contains figures with a full battery of plots for comparisons across algorithm specifications and for the means and 68 percent credible sets for the full trajectories of the elements of $\mathbf{s}_{0:T}$. The key takeaway is simply that as we increase the number of mutation steps, the SMC posterior estimates become indistinguishable from the MCMC posterior estimates. This gives us a sense of what we should expect to be true when examining the more formal estimation metrics in the next section.

While Figure 1 shows a comparison of posteriors for a single t, one could also examine a visual comparison for the full trajectory $p(a_{5,4,0:T}|\mathbf{y}_{1:T})$, which

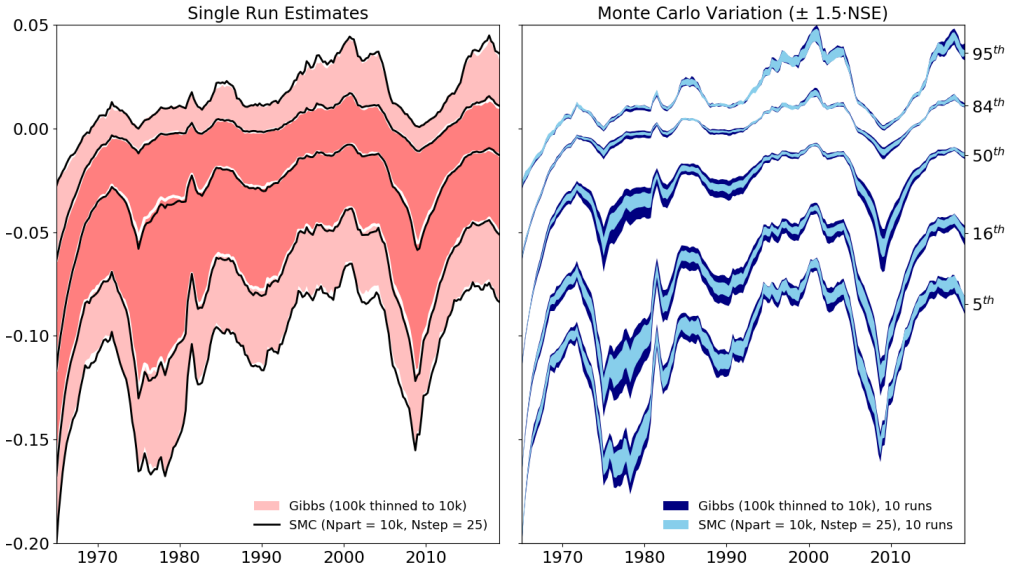

FIGURE 2.—SMC vs. MCMC: Quantile Estimates of $p(a_{5,4,t}|\mathbf{y}_{1:T})$

is what we present in the left panel of Figure 2 (titled "Single Run Estimates"). The shaded regions in the figure demarcate, at each t, the MCMC estimate of $p(a_{5,4,t}|\mathbf{y}_{1:T})$ divided into its 5^{th}, 16^{th}, 50^{th}, 84^{th}, and 95^{th} quantiles. The black lines are estimates from the SMC-25 algorithm of the same quantiles. In fact, the shaded regions are divided by thin white bands but, to the extent that the white bands are not visible and the black lines appear to outline the shaded regions, the MCMC and SMC quantile estimates coincide across all t. We will not belabor the point further as these visual comparisons merely suggest that SMC is capable of generating the "same" results as MCMC. The more rigorous comparison is foreshadowed by the right panel of Figure 2, to which we next turn.

4.4 Formal Comparisons of Estimation Accuracy

There will of course be Monte Carlo variation in the estimates generated across runs of any of the estimation algorithms. Having seen that SMC can give estimation results comparable to MCMC, we next turn to comparisons of the Monte Carlo variation of estimates across repeated runs. The goal is to assess SMC's reliability at systematically producing accurate posterior characterizations.

We begin with the posterior features described at the end of the previous section, but now turning to the right panel of Figure 2. The right panel shows the Monte Carlo variation in the estimates of each quantile across 10 runs of each

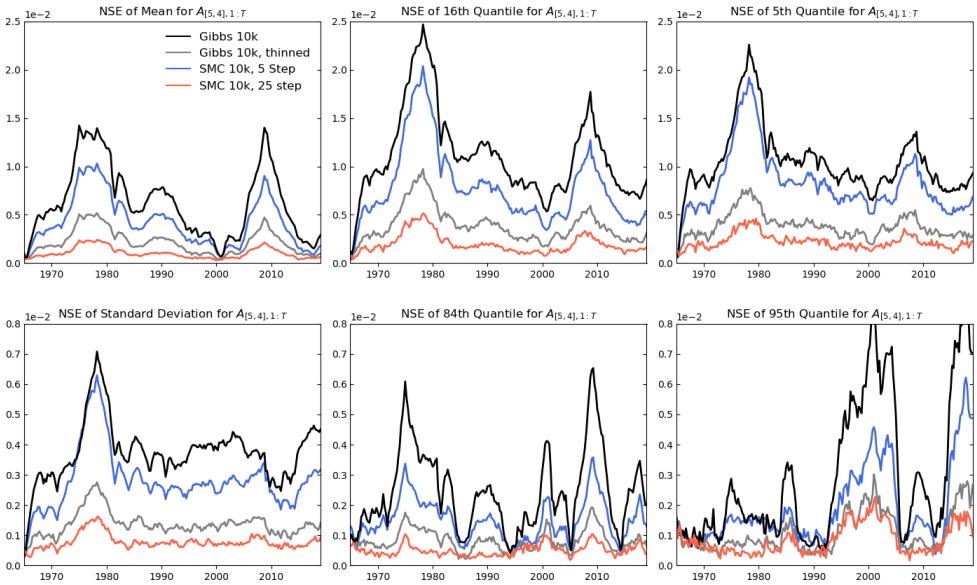

FIGURE 3.—Numerical Standard Errors of Posterior Moments of $a_{5,4,1:T}$

algorithm. Each band is centered at the mean of the estimates and extends to ± 1.5 of the standard deviation of the estimates across runs (the numerical standard error or "NSE"). Hence, provided that the bands are centered in approximately the same place, the algorithm with the narrower bands is the more accurate one. In the example in 2 one can see that the SMC-25 bands live strictly inside of the MCMC bands for virtually all quantiles and all t, and hence provides systematically more accurate approximation to the posterior.

Another way to see the result is to focus directly on the NSEs. For example, Figure 3 shows the NSE of the means, standard deviations, and four different quantiles of the approximation to the distribution $p(a_{5,4,t}|\mathbf{y}_{1:T})$ at each $t = 1, \ldots, T$. Each line in a panel shows the NSE from estimation with a different algorithm. Smaller numbers indicate increased estimation accuracy, in the sense of lower variation across runs of the estimation algorithm. The results are unambiguous. For every object of interest, and every t, the SMC algorithm with 25 mutation steps dominates the other estimation approaches, which is shown by the fact that its line traces out the smallest values in each panel and at each t.

The important question is then the extent to which this is a more general phenomenon for features of the model's posterior. It turns out that the key relationships among the algorithms that became visible with the spotlight on

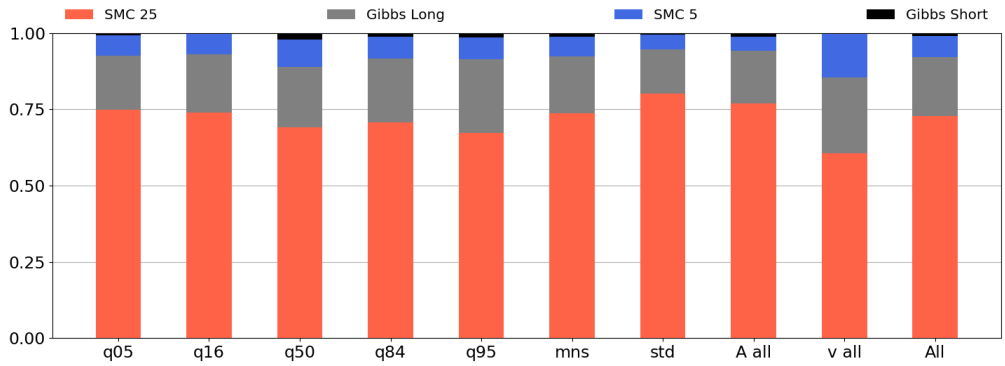

FIGURE 4.—Share of Smallest NSE Posterior Feature by Algorithm.

$p(a_{5,4,0:T}|\mathbf{y}_{1:T})$ in Figures 2 and 3 hold very generally across all the posterior for all elements of $\mathbf{s}_{0:T}$. Figure 4 summarizes numerous comparisons of the NSEs from estimating various posterior quantities. Each bar indicates the share of moments for which each algorithm has the smallest NSE. The label on the horizontal axis indicates which subset of moments we are restricting attention to for the comparison, while the rightmost bar gives the aggregated result across all moments we computed. The results are again unambiguous, with the SMC-25 algorithm having the smallest NSE for all subsets of moments we compare.

Lest one fear that the NSEs are somehow misleading as a result of an estimation algorithm systematically missing a given posterior feature in the same way, the appendix contains a full battery of figures (Figures 6–12) comparing the mean estimates of each feature from MCMC-long and SMC-25. Those figures show that the two estimates are always on top of each other. The appendix also contains figures showing the NSEs of each moment, at each t for all elements of $\mathbf{s}_{0:T}$ (Figures 13–19).

5. Computational Environment

The SMC results in the previous section make use of computing resources that few, if any, researchers will have physically present in their offices. For this reason, and as an additional contribution, we demonstrate that the algorithms could be readily implemented with publicly provisioned computing resources. In fact we had little choice; we do not have 96 vCPU machines in our offices either. To this end, we carried out all of our computations using the "public cloud," namely, using resources available through Amazon Web Services (AWS).

From AWS we could readily obtain virtual machines equipped with as many as 96 virtual CPUs. The fact that such a large degree of parallelization is now readily available on even a single machine instance is important for preserving the simplicity of our implementation, as it meant that we did not need to combine multiple virtual machine instances into a cluster.

Of course, since we provisioned our computing resources from the public cloud, we had to pay market prices for those resources. At the time of writing, virtual machine instances with 96 virtual CPUs were readily available as "spot" instances for $0.95 per hour. Hence, the posterior update at T for SMC-25, taking 9.5 minutes (see Table I), cost only $0.15. The "on demand" price was $4.61 per hour, at which rate the 9.5-minute update would still cost only $0.73.

Another consideration closely related to computing on the public cloud is the choice of software in which to write our computer code. We wanted the implementation to be entirely in a programming language that was open source and free to use, including its parallel functionality, so that we had no special licenses to procure in order to execute our programs on the AWS machines.[23] To this end, we wrote our computer code in `Julia`.[24] Essentially, all of `Julia`'s functionality, including the ability to parallelize to an arbitrary number of cores, is freely available and free to distribute and deploy. Though relatively new, `Julia` has garnered substantial attention as a programming language for scientific computing. For quantitative applications in economics, `Julia`'s attractive balance of performance and ease of use has been highlighted by Aruoba and Fernández-Villaverde (2015) and by its ongoing inclusion in the QuantEcon organization's projects.[25] The machines on which we executed the programs run the `Ubuntu 18.04 LTS` operating system, which is also open source and free to use.

Lastly, we handle the entirety of the parallelization with high-level commands

[23] For some closed-source, proprietary programming languages we would need to procure special licenses to install and use the requisite software on cloud resources. Some such programs also require additional proprietary packages to use high-level commands for parallelism, and these packages can come at a significant additional cost. In the recent past, some such software programs have also restricted the degree of parallelism allowed even after purchasing the requisite licenses for parallel functionality.

[24] See Bezanson et al. (2017) for an exposition of the key concepts guiding the `Julia` programming language's original, and ongoing, development.

[25] See Aruoba and Fernández-Villaverde (2018) for updated computational results. QuantEcon was founded, and remains co-chaired, by Thomas J. Sargent and John Stachurski. See https://quantecon.org/ for more information on the QuantEcon organization.

in `Julia`, i.e., we make no direct recourse to passing instructions at a lower level.[26] In particular, the aspects of the code that are specific to a parallel implementation consist entirely of only the following three functions or decorators defined by the `Distributed` package included in the standard `Julia` distribution: 1) `addprocs`, to make all of the machine's processors available to `Julia` for computation (known in `Julia` as "workers"); 2) `@everywhere`, to instantiate the key model objects on all the workers; and 3) `pmap`, to distribute the workload of parallelizable tasks across the available workers.

6. Conclusion

We developed a sequential Monte Carlo (SMC) algorithm for sequential Bayesian inference in vector autoregressions with stochastic volatility. The algorithm builds particle approximations to the sequence of posteriors under an expanding window of data, adapting the particles from one approximation to the next. Our SMC algorithm embeds the known MCMC algorithm for the model as an effective mutation kernel for fighting particle degeneracy. The algorithm is highly parallelizable, allowing for rapid updates from one posterior to the next. We applied the algorithm to a seven-variable vector autoregression and demonstrated that the SMC algorithm yields inference as precise as running the MCMC algorithm from scratch, but does so in only a fraction of the time.

Lastly, although MCMC and SMC methods tend to be pitted against each other in the literature, this paper demonstrates in a practical setting how the two methods can be complementary, namely, by using a known and efficient MCMC algorithm for a particular model as the mutation kernel within the SMC algorithm. In doing so we are largely able to have "the best of both worlds" in a time-series application.

[26] Fernández-Villaverde and Valencia (2018) also highlight the simplicity of parallel computing in `Julia` using the same functions that we highlight here.

A. Data

The specific data series used in the VAR, and their transformations, are given in Table A-1.

TABLE A-1
DATA FOR VAR-SV

Name	Abbreviation	FRED Mnemonic	Transformation
Real GDP	GDP	GDPC1	$400 \cdot \log$
Prices	P	GDPCTPI	$400 \cdot \log$
Consumption	C	PCECC96	$400 \cdot \log$
Investment	I	GPDIC1	$400 \cdot \log$
Hours worked	H	HOANBS	$400 \cdot \log$
Wages	W	COMPRNFB	$400 \cdot \log$
Interest rates	R	FEDFUNDS	none

Notes. All data are quarterly. Variables are ordered in $\mathbf{y}_t$ as they are listed above. Training sample: 1954:Q3 – 1964:Q4. Estimation sample: 1965:Q1 – 2019:Q1. All variables except FEDFUNDS are seasonally adjusted and the adjustment is made by the data source.

B. Prior Hyperparameters in the Application

The prior specification described in Table A-2 references a few objects we describe here. We follow Primiceri (2005) and train the prior on a pre-sample of observations spanning 1954:Q3 to 1964:Q4. Using the 1954:Q3 observation for initialization, we fit a constant-coefficient VAR(1) with intercept to the $T_0 = 41$ remaining pre-sample observations. Letting $\hat{\mathbf{B}}_0$ and $\hat{\mathbf{\Sigma}}_0$ denote the MLE estimates of the VAR coefficients over the pre-sample, we compute the $T_0 \times n$ matrix of pre-sample residuals $\hat{\mathbf{U}}_0 = \mathbf{Y}_0 - \mathbf{X}_0\hat{\mathbf{B}}_0$ and thus $\hat{\mathbf{U}}_0$ column-wise stores the time series of T_0 estimated residuals, $\hat{\mathbf{u}}_i$, for each variable $i = 1, \dots, n$.

A few aspects of the prior are worth highlighting. The prior for vec($\mathbf{B}$) is centered at a random walk with Minnesota variances. The priors for the initial states are independent Gaussian and trained on the pre-sample. The priors on the the static parameters in each law of motion are independent across equations and

take the natural conjugate form

$$p(\sigma_j^2) = IG\left(\frac{a_{j,0}}{2}, \frac{b_{j,0}}{2}\right) \tag{24}$$

$$p(\boldsymbol{\beta}_j \mid \sigma_j^2) = \mathrm{N}\left(\overline{\boldsymbol{\beta}}_{j,0},\ \sigma_j^2 \cdot \boldsymbol{\Omega}_{j,0}^{-1}\right), \tag{25}$$

denoted $NIG(a_{j,0}/2,\ b_{j,0}/2,\ \overline{\boldsymbol{\beta}}_{j,0},\ \boldsymbol{\Omega}_{j,0}^{-1})$. The *marginal* prior distribution of $\boldsymbol{\beta}$ in each equation is a multivariate t-distribution with covariance matrix $\frac{b_{j,0}}{a_{j,0}-2}\boldsymbol{\Omega}_{j,0}^{-1}$, so we set $\boldsymbol{\Omega}_{i,0}^{-1}$ in order to directly set the marginal prior variance of $\boldsymbol{\beta}$.

TABLE A-2
PRIOR FOR VAR-SV

Object	Distribution	Details
vec($\mathbf{B}$)	$N\left(\mathrm{vec}\left(\overline{\mathbf{B}}_0\right),\ \mathrm{diag}\left[\mathrm{sd}(b_{i,j,\ell})^2\right]\right)$	$\overline{\mathbf{B}}_0 = \begin{bmatrix}\mathbf{I}_n & \mathbf{0}'_{m-n,n}\end{bmatrix}'$
		$\mathrm{sd}(b_{i,j,\ell}) = \begin{cases}\theta_4 s_i & \ell = 0 \\ \frac{\theta_1}{\ell^{\theta_3}} & i = j \\ \frac{\theta_1\theta_2}{\ell^{\theta_3}}\frac{s_i}{s_j} & \text{else}\end{cases}$
		$\theta_1 = 0.1;\ \theta_2 = 1;\ \theta_3 = 1;\ \theta_4 = 100$
		s_i: OLS residual std from full sample AR(1) for variable i
$\mathbf{v}_0$	$N\left(\overline{\mathbf{v}}_0,\ \mathbf{I}_n\right)$	$\overline{\mathbf{v}}_0 = \log\mathrm{diag}(\hat{\boldsymbol{\Sigma}}_0)$
$\mathbf{a}_{i,0}$	$N\left(\overline{\mathbf{a}}_{i,0},\ 4\cdot\mathrm{diag}(\mathbf{q}_i)\right)$	$\overline{\mathbf{a}}_{i,0} = \left[\hat{\alpha}_k^{(OLS)}\right]$
		$\mathbf{q}_i = \left[\mathrm{var}\left(\hat{\alpha}_k^{(OLS)}\right)\right]$
		from regression $\hat{\mathbf{u}}_j = \sum_{k=1}^{j-1}\alpha_k\hat{\mathbf{u}}_k + \boldsymbol{\varepsilon}$
$\boldsymbol{\beta}_j,\ \sigma_j^2$	$NIG\left(\frac{a_{j,0}}{2},\ \frac{b_{j,0}}{2},\ \overline{\boldsymbol{\beta}}_{j,0},\ \mathrm{diag}(\omega_{j,1},\ \omega_{j,2})\right)$	$a_{j,0} = \begin{cases}8 & j = 1,\ ...,\ n \\ 3 & else\end{cases}$
		$b_{j,0} = d_{j,0}\cdot\begin{cases}0.03 & j = 1,\ ...,\ n \\ 0.1^2 & else\end{cases}$
		$\overline{\boldsymbol{\beta}}_{j,0} = \begin{bmatrix}0.9 & 0.0\end{bmatrix}'$
		$\omega_{j,1} = \frac{a_{j,0}-2}{b_{j,0}}\cdot\begin{cases}0.1^2 & j = 1,...,n \\ 0.01^2 & else\end{cases}$
		$\omega_{j,2} = \frac{a_{j,0}-2}{b_{j,0}}\cdot\begin{cases}0.1^2 & j = 1,...,n \\ 0.001^2 & else\end{cases}$

Notes. $\mathbf{s}_t = [\mathbf{v}_t', \mathbf{a}_t']'$ with generic element $s_{j,t}$, $j = 1, \ldots, n_s$. $b_{i,j,\ell}$ is the element of $\mathbf{B}$ corresponding to the $(i,\ j)^{th}$ element of $\mathbf{B}'_{(\ell)}$ from equation (1).

C. Reformulation as SMC in the Space of All Unobservables

While Section 3 describes the algorithm as we implement it in practice, in this section we describe how it should be formally interpreted as an SMC algorithm to which results such as CLTs and SLLNs in the SMC literature apply. Namely, it might appear that there is an inconsistency between the target density in Correction and the invariant distribution of the Mutation kernel. Recall that the target density, as defined during the Correction phase, is the marginal posterior $p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$, while the mutation kernel iterates in the large parameter space of $(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta})$. This matters because SMC's theoretical results with mutation steps require these two distributions to coincide, in the sense that K_t should be a Markov transition kernel with the target density as its invariant distribution. As a Gibbs sampler, K_t is Markov in the full set of unobservables $(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta})$ and possesses the unique invariant distribution $p(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$. However, because K_t is a *multi-step* Gibbs sampler, it is *not* Markov in the subset of unobservables $\mathbf{s}_{0:t}$. Nor does K_t have the marginal posterior $p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})$ as its unique invariant distribution. See Robert (2007) and Robert and Casella (2004) for further discussion on these properties of multi-step Gibbs samplers. We next show that our estimation algorithm implements a valid SMC algorithm in the full parameter space.

At a high level, let the stage t target density be the joint distribution $\tilde{\pi}_t = p(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$. Now consider factoring the joint posterior density as follows:

$$\tilde{\pi}_t = p(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t}) = p(\mathbf{s}_{0:t}|\mathbf{y}_{1:t}) p(\mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t}, \mathbf{s}_{0:t}) \tag{26}$$

and thus a kernel of the target density is

$$\tilde{k}_t = p(\mathbf{s}_{0:t}) p(\mathbf{y}_{1:t}|\mathbf{s}_{0:t}) p(\mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t}, \mathbf{s}_{0:t}). \tag{27}$$

One could then importance sample the target density by sampling from a density $\tilde{g}_t(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t})$ and giving the i^{th} draw the unnormalized weight

$$w_t^i = \frac{p(\mathbf{s}_{0:t}) p(\mathbf{y}_{1:t}|\mathbf{s}_{0:t}) p(\mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t}, \mathbf{s}_{0:t})}{\tilde{g}_t(\mathbf{s}_{0:t}, \mathbf{b}, \boldsymbol{\theta}|\mathbf{y}_{1:t})} \tag{28}$$

Without loss of generality, the proposal distribution can be factored as

$$\tilde{g}_t(\mathbf{s}_{0:t}, \mathbf{b}, \theta|\mathbf{y}_{1:t}) = \tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})\, \tilde{g}_{t,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t}). \tag{29}$$

Multiplying the top and bottom by k_{t-1} gives

$$\begin{aligned} w_t^i = & \frac{p(\mathbf{s}_{0:t-1})p(\mathbf{y}_{1:t-1}|\mathbf{s}_{0:t-1})p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})}{p(\mathbf{s}_{0:t-1})p(\mathbf{y}_{1:t-1}|\mathbf{s}_{0:t-1})p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})} \\ & \cdot \frac{p(\mathbf{s}_{0:t})p(\mathbf{y}_{1:t}|\mathbf{s}_{0:t})p(\mathbf{b}, \theta|\mathbf{y}_{1:t}, \mathbf{s}_{0:t})}{\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})\, \tilde{g}_{t,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t})}. \end{aligned} \tag{30}$$

Noting that

$$\frac{p(\mathbf{s}_{0:t})}{p(\mathbf{s}_{0:t-1})} = p(\mathbf{s}_t|\mathbf{s}_{0:t-1}) \qquad \text{and} \qquad \frac{p(\mathbf{y}_{1:t}|\mathbf{s}_{0:t})}{p(\mathbf{y}_{1:t-1}|\mathbf{s}_{0:t-1})} = p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t}), \tag{31}$$

equation (30) can be rearranged and written as

$$\begin{aligned} w_t^i = & \frac{p(\mathbf{s}_{0:t-1})p(\mathbf{y}_{1:t-1}|\mathbf{s}_{0:t-1})p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})}{p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})} \\ & \cdot \frac{p(\mathbf{s}_t|\mathbf{s}_{0:t-1})p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t})p(\mathbf{b}, \theta|\mathbf{y}_{1:t}, \mathbf{s}_{0:t})}{\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})\, \tilde{g}_{t,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t})}. \end{aligned} \tag{32}$$

Next, multiplying top and bottom by the proposal densities from $t-1$ gives

$$\begin{aligned} w_t^i = & \frac{p(\mathbf{s}_{0:t-1})p(\mathbf{y}_{1:t-1}|\mathbf{s}_{0:t-1})p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})}{\tilde{g}_{t-1,1}(\mathbf{s}_{0:t-1}|\mathbf{y}_{1:t-1})\, \tilde{g}_{t-1,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t-1}, \mathbf{y}_{1:t-1})} \\ & \cdot \frac{p(\mathbf{s}_t|\mathbf{s}_{0:t-1})p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t})p(\mathbf{b}, \theta|\mathbf{y}_{1:t}, \mathbf{s}_{0:t})\tilde{g}_{t-1,1}(\mathbf{s}_{0:t-1}|\mathbf{y}_{1:t-1})}{\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})\, \tilde{g}_{t,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t})} \\ & \cdot \frac{\tilde{g}_{t-1,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t-1}, \mathbf{y}_{1:t-1})}{p(\mathbf{b}, \theta|\mathbf{y}_{1:t-1}, \mathbf{s}_{0:t-1})} \end{aligned} \tag{33}$$

From equation (28), the first term is equivalent to w_{t-1}^i. If we also set

$$\tilde{g}_{t,2}(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t}) = p(\mathbf{b}, \theta|\mathbf{s}_{0:t}, \mathbf{y}_{1:t}), \tag{34}$$

and similarly for $t-1$, then all terms directly invoking $(\mathbf{b}, \theta)$ cancel and the

expression for the weights becomes

$$w_t^i = w_{t-1}^i \frac{p(\mathbf{s}_t|\mathbf{s}_{0:t-1})p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\mathbf{s}_{0:t})\tilde{g}_{t-1,1}(\mathbf{s}_{0:t-1}|\mathbf{y}_{1:t-1})}{\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})} \tag{35}$$

Lastly, choosing $\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{s}_{0:t})$, and similarly for $\tilde{g}_{t-1,1}$, we have that

$$\frac{\tilde{g}_{t-1,1}(\mathbf{s}_{0:t-1}|\mathbf{y}_{1:t-1})}{\tilde{g}_{t,1}(\mathbf{s}_{0:t}|\mathbf{y}_{1:t})} = \frac{p(\mathbf{s}_{0:t-1})}{p(\mathbf{s}_{0:t})} = \frac{1}{p(\mathbf{s}_t|\mathbf{s}_{0:t-1})} \tag{36}$$

and equation (35) becomes

$$w_t^i = w_{t-1}^i \, p(\mathbf{y}_t|\mathbf{y}_{1:t-1},\mathbf{s}_{0:t}). \tag{37}$$

Defining the incremental weight as in equation (23), we can see that the recursive expression for the weights is the same as that given in Section 3.

The weights we calculate in the algorithm described in the main text are then equivalent to the weights we would have calculated by "targeting" the joint posterior of all unobservables, and sampling the static parameters from their exact conditional posterior. Indeed, we do sample them in this way; we just do not bother to do it until initializing the Mutation phase, as can be seen in Algorithm 4. This is fine because, as can be seen in the preceding arguments, the sample of $(\mathbf{b}, \boldsymbol{\theta})$ does not affect the weights.

D. Additional Computational Considerations

Defining

$$\mathbf{H}_t \equiv \boldsymbol{\Sigma}_t^{-1} = \left(\mathbf{A}_t^{-1}\boldsymbol{\Lambda}_t\mathbf{A}_t^{-1\prime}\right)^{-1} = \mathbf{A}_t'\boldsymbol{\Lambda}_t^{-1}\mathbf{A}_t\,, \tag{38}$$

the posterior mean and covariance matrix in equations (13) and (14) can be written as

$$\mathbf{V}_T = \left(\mathbf{V}_0^{-1} + \sum_{t=1}^{T}\left(\mathbf{H}_t \otimes \mathbf{x}_t\mathbf{x}_t'\right)\right)^{-1} \tag{39}$$

$$\overline{\mathbf{b}}_T = \mathbf{V}_T\left(\mathbf{V}_0^{-1}\overline{\mathbf{b}}_0 + \sum_{t=1}^{T}\mathrm{vec}\left(\mathbf{x}_t\mathbf{y}_t'\mathbf{H}_t\right)\right) \tag{40}$$

Note that $\mathbf{H}_t$ can be constructed from $\mathbf{s}_t$ without needing to invert any matrices by populating $\mathbf{A}_t$ directly with the elements of $\mathbf{a}_t$ and using the fact that

$$\mathbf{\Lambda}_t^{-1} = \text{diag}\big(1./\exp(\mathbf{v}_t)\big)\,. \tag{41}$$

D.1 Posterior density of state transition parameters

In the absence of the truncation, the posterior density for $\boldsymbol{\theta}_i$ can be factored as

$$p(\boldsymbol{\beta}_i, \sigma_i^2 | s_{i,1:t}) = p(\sigma_i^2 | s_{i,1:t}) p(\boldsymbol{\beta}_i | s_{i,1:t}, \sigma_i^2) \tag{42}$$

where the density of $\boldsymbol{\beta}_i$ is multivariate normal,

$$p(\beta^{\{0\}}, \beta^{\{1\}} | s_{i,1:t}, \sigma^2) = N\left(\begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix}, \begin{bmatrix} Q_{0,0} & Q_{0,1} \\ Q_{1,0} & Q_{1,1} \end{bmatrix} \right) \tag{43}$$

The joint distribution in equation (43) can be equivalently expressed as a factorization into a marginal and a conditional as

$$p(\beta_i^{\{0\}} | s_{i,1:t}, \sigma_i^2) p(\beta_i^{\{1\}} | s_{i,1:t}, \sigma_i^2, \beta_i^{\{0\}}) \tag{44}$$

where $p(\beta^{\{0\}} | \mathbf{y}_{1:t}, \sigma^2) = N(\mu_0, Q_{0,0})$ and

$$p(\beta^{\{1\}} | \mathbf{y}_{1:t}, \sigma^2, \beta^{\{0\}}) = N\left(\mu_{1|0}, Q_{1|0}\right), \tag{45}$$

for

$$\mu_{1|0} = \mu_1 + Q_{1,0} Q_{0,0}^{-1} (\beta^{\{0\}} - \mu_0) \tag{46}$$

$$Q_{1|0} = Q_{1,1} - Q_{1,0} Q_{0,0}^{-1} Q_{0,1} \tag{47}$$

Accounting for the truncation yields density

$$\tilde{p}(\beta^{\{1\}} | \mathbf{y}_{1:t}, \sigma^2, \beta^{\{0\}}) = \frac{p(\beta^{\{1\}} | \mathbf{y}_{1:t}, \sigma^2, \beta^{\{0\}}) \cdot \mathbf{1}\left\{ |\beta^{\{1\}}| < 1 \right\}}{\mathbf{\Phi}(1 | \mu_{1|0}, Q_{1|0}) - \mathbf{\Phi}(-1 | \mu_{1|0}, Q_{1|0})} \tag{48}$$

where $\mathbf{\Phi}(x|a,b)$ denotes the cdf of a univariate normal distribution with mean a and variance b evaluated at the value x. Random samples from truncated normals

can be generated efficiently using the algorithms in Robert (1995).

E. MCMC Algorithm for Mutation

Equations (49)–(67) below summarize the key relationships referenced in the subsequent description of the MCMC algorithm.

$$\mathbf{y}_t' = \mathbf{x}_t'\mathbf{B} + \mathbf{u}_t' \qquad \mathbf{u}_t \sim N(\mathbf{0},\, \mathbf{\Sigma}_t) \tag{49}$$

$$\mathbf{\Sigma}_t = \mathbf{A}_t^{-1}\mathbf{\Lambda}_t(\mathbf{A}_t^{-1})' \tag{50}$$

$$\mathbf{A}_t = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \mathbf{a}_{2,t}' & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ \mathbf{a}_{n,t}' & & & 1 \end{bmatrix} \tag{51}$$

$$\mathbf{\Lambda}_t = \text{diag}[\exp(\mathbf{v}_t)] \tag{52}$$

$$\mathbf{b} = \text{vec}(\mathbf{B}) \tag{53}$$

$$\mathbf{a}_{i,t} = \mathbf{G}_i\mathbf{a}_{i,t-1} + \mathbf{g}_i + \boldsymbol{\xi}_{i,t} \qquad \boldsymbol{\xi}_{i,t} \sim N(\mathbf{0},\, \mathbf{\Omega}_i) \tag{54}$$

$$\mathbf{v}_t = \mathbf{G}_1\mathbf{v}_{t-1} + \mathbf{g}_1 + \boldsymbol{\eta}_t \qquad \boldsymbol{\eta}_t \sim N(\mathbf{0},\, \mathbf{\Omega}_1) \tag{55}$$

$$\mathbf{a}_{i,0} \sim N(\bar{\mathbf{a}}_{i,0},\, \mathbf{W}_{i,0}) \tag{56}$$

$$\mathbf{v}_0 \sim N(\bar{\mathbf{v}}_{0|0},\, \mathbf{P}_{0|0}) \tag{57}$$

$$\mathbf{b} \sim N(\bar{\mathbf{b}}_0,\, \mathbf{V}_0) \tag{58}$$

$$\begin{bmatrix} \mathbf{G}_i' & \mathbf{g}_i \end{bmatrix}', \mathbf{\Omega}_i \sim MNIW\left(\nu_0^{(i)},\, \mathbf{\Psi}_0^{(i)},\, \overline{\mathbf{\Xi}}_0^{(i)},\, \mathbf{\Gamma}_0^{(i)-1}\right) \tag{59}$$

$$\boldsymbol{\theta} = \{\mathbf{G}_i,\, \mathbf{g}_i,\, \mathbf{\Omega}_i\}_{i=1}^n \tag{60}$$

$$\bar{c} = 0.0001 \tag{61}$$

$$\kappa(x) = \sum_{k=1}^{10} p_k\, N(x \mid m_k,\, v_k^2) \tag{62}$$

$$\mathbf{u}_t = \mathbf{y}_t - \mathbf{B}'\mathbf{x}_t \tag{63}$$

$$\mathbf{e}_t = \mathbf{A}_t\mathbf{u}_t \tag{64}$$

$$\tilde{\mathbf{e}}_t = \ln[\mathbf{e}_t^2 + \bar{c}] \tag{65}$$

$$\tilde{\boldsymbol{\varepsilon}}_t = \tilde{\mathbf{e}}_t - \mathbf{v}_t \tag{66}$$

$$z_{i,t} \in \{1,\, ...,\, 10\} \tag{67}$$

Algorithm 5 - MCMC for VAR-SV

Block 1. $p(\mathbf{v}_{0:T} \mid \mathbf{y}_{1:T},\ \mathbf{b},\ \mathbf{a}_{1:T},\ \boldsymbol{\theta},\ z_{1:T})$

Let $q(\mathbf{v}_{0:T}) = \tilde{p}(\mathbf{v}_{0:T} \mid \tilde{\mathbf{e}}_{1:T},\ \boldsymbol{\theta},\ z_{1:T})$ *denote the posterior for the linear Gaussian state-space system:*

$$\begin{aligned} \tilde{\mathbf{e}}_t &= \mathbf{v}_t + \mathbf{m}_t(\mathbf{z}_t) + \boldsymbol{\varepsilon}_t & \boldsymbol{\varepsilon}_t &\sim N(\mathbf{0},\ \boldsymbol{\Phi}_t(\mathbf{z}_t)) \\ \mathbf{v}_t &= \mathbf{G}_1 \mathbf{v}_{t-1} + \mathbf{g}_1 + \boldsymbol{\eta}_t & \boldsymbol{\eta}_t &\sim N(\mathbf{0},\ \boldsymbol{\Omega}_1) \\ \mathbf{v}_0 &\sim N(\bar{\mathbf{v}}_{0|0},\ \mathbf{P}_{0|0}) \end{aligned}$$

where, for each $t = 1, \ldots, T$, $\tilde{\mathbf{e}}_t$ *is observable and computed from equations* (63)–(65) *and*

$$\mathbf{m}_t(\mathbf{z}_t) = \left[m_{z_{1,t}}, \ldots, m_{z_{n,t}}\right] \quad \textit{and} \quad \boldsymbol{\Phi}_t(\mathbf{z}_t) = diag\left(\left[\delta^2_{z_{1,t}}, \ldots, \delta^2_{z_{n,t}}\right]\right).$$

with $m_{z_{i,t}}$ *and* $\delta^2_{z_{i,t}}$ *determined by the integer value of* $z_{i,t}$ *and values given in Table A-3.*

Sample a proposal $\mathbf{v}^*_{0:T} \sim q(\mathbf{v}_{0:T})$ *by running the Kalman filter forward and then sampling from the simulation Kalman smoother.*

Compute the acceptance/rejection probabilities

$$r = \frac{p\left(\mathbf{v}^*_{0:T} \mid \cdot\right) q\left(\mathbf{v}^{(m-1)}_{0:T}\right)}{p\left(\mathbf{v}^{(m-1)}_{0:T} \mid \cdot\right) q\left(\mathbf{v}^*_{0:T}\right)} = \frac{\prod_{t=1}^{T}\left[N(\mathbf{e}_t \mid \mathbf{0},\ \boldsymbol{\Lambda}^*_t)\prod_{i=1}^{n}\kappa\left(\tilde{\varepsilon}^{(m-1)}_{i,t}\right)\right]}{\prod_{t=1}^{T}\left[N\left(\mathbf{e}_t \mid \mathbf{0},\ \boldsymbol{\Lambda}^{(m-1)}_t\right)\prod_{i=1}^{n}\kappa\left(\tilde{\varepsilon}^*_{i,t}\right)\right]}$$

$$\alpha = \min\{1,\ r\}$$

where each $\boldsymbol{\Lambda}^*_t = f(\mathbf{v}^*_t)$ *according to equation* (52) *and the* $\tilde{\varepsilon}^*_{i,t}$ *are computed as in equation* (66) *using the proposed* $\mathbf{v}^*_t$.

Set $\mathbf{v}_{0:T}^{(m)}$ *according to*

$$\mathbf{v}_{0:T}^{(m)} = \begin{cases} \mathbf{v}_{0:T}^{*} & \text{with probability } \alpha \\ \mathbf{v}_{0:T}^{(m-1)} & \text{with probability } 1-\alpha. \end{cases}$$

Block 2. $p\left(\mathbf{b}|\mathbf{y}_{1:T}, \mathbf{v}_{0:T}, \mathbf{a}_{0:T}, \theta, z_{1:T}\right)$

Note that

$$p\left(\mathbf{b}|\mathbf{y}_{1:T}, \mathbf{v}_{0:T}, \mathbf{a}_{0:T}, \theta, z_{1:T}\right) = p\left(\mathbf{b}|\mathbf{y}_{1:T}, \mathbf{v}_{0:T}, \mathbf{a}_{0:T}\right) = \mathrm{N}\left(\mathbf{b} \mid \bar{\mathbf{b}}_T, \mathbf{V}_T\right)$$

with the parameters $\left(\bar{\mathbf{b}}_T, \mathbf{V}_T\right)$ *given in equations* (13) *and* (14) *of the main text.*

Block 3. $p(\mathbf{a}_{0:T} \mid \mathbf{y}_{1:T}, \mathbf{v}_{1:T}, \mathbf{b}, \theta)$

For $i = 2, ..., n$ *use the Kalman filter and simulation smoother to sample* $\mathbf{a}_{i,0:T}$ *from the posterior of this linear Gaussian system:*

$$\begin{aligned} u_{i,t} &= -\begin{bmatrix} u_{1,t} & \cdots & u_{i-1,t} \end{bmatrix} \mathbf{a}_{i,t} + r_{i,t} && r_{i,t} \sim \mathrm{N}\left(0, \exp(v_{i,t})\right) \\ \mathbf{a}_{i,t} &= \mathbf{G}_i \mathbf{a}_{i,t-1} + \mathbf{g}_i + \boldsymbol{\xi}_{i,t} && \boldsymbol{\xi}_{i,t} \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Omega}_i) \\ \mathbf{a}_{i,0} &\sim \mathrm{N}(\bar{\mathbf{a}}_{i,0}, \mathbf{W}_{i,0}). \end{aligned}$$

where each $\mathbf{u}_t$ *is observable and computed via equation* (63).

Block 4. $\theta \sim p(\theta \mid \mathbf{v}_{0:T}, \mathbf{a}_{0:T})$

For $i = 1, \ldots, n_s$ *sample* θ_i *as described in Appendix D.1.*

Block 5. $p(z_{1:T} \mid \mathbf{y}_{1:T}, \mathbf{b}, \mathbf{a}_{0:T}, \theta)$

For each $i = 1, \ldots, n$ *and* $t = 1, \ldots, T$, *draw* $z_{i,t} \in \{1, \ldots, 10\}$ *according to the discrete distribution with probabilities*

$$Pr(z_{i,t} = k \mid \tilde{\varepsilon}_{i,t}) = \frac{p_k \, N(\tilde{\varepsilon}_{i,t} \mid m_k, \delta_k^2)}{\sum_{j=1}^{10} p_j \, N(\tilde{\varepsilon}_{i,t} \mid m_j, \delta_j^2)} \qquad k = 1, ..., 10$$

where $\tilde{\varepsilon}_{i,t}$ denotes the i-th element of $\tilde{\boldsymbol{\varepsilon}}_t$ as defined in equation (66) *and the parameters $\left\{p_k, m_k, \delta_k^2\right\}_{k=1}^n$ are given in Table A-3.*

k	p_k	m_k	δ_k^2
1	0.00609	1.92677	0.11265
2	0.04775	1.34744	0.17788
3	0.13057	0.73504	0.26768
4	0.20674	0.02266	0.40611
5	0.22715	-0.85173	0.62699
6	0.18842	-1.97278	0.98583
7	0.12047	-3.46788	1.57469
8	0.05591	-5.55246	2.54498
9	0.01575	-8.68384	4.16591
10	0.00115	-14.65000	7.33342

TABLE A-3

GAUSSIAN MIXTURE APPROXIMATION TO $\ln \chi^2(1)$ DISTRIBUTION, AS GIVEN IN OMORI ET AL. (2007).

The main element to call attention to is the introduction of a vector of indicators $z_{1:T}$ on mixture components used in sampling $\mathbf{v}_{0:T}$. The use of the mixture approximation introduces a potential wedge between the invariant distribution of the algorithm's simulations and the model's true posterior. The MCMC algorithm we use corrects for this wedge by using the mixture approximation as a device for generating good proposals rather than just accepting the mixture output "as is," yielding a sampler with the correct invariant distribution.

F. MCMC Algorithm "Getting it Right"

The MCMC algorithm plays a critical role in our SMC algorithm, so we used the "Getting it Right" algorithm of Geweke (2004) to formally verify that the algorithm (and our implementation of it) are correct. The test environment uses $T = 10$, $n = 3$, and $p = 4$. We take 10^5 draws from the "marginal-conditional (MC)" simulator and run 4×10^6 iterations of the "successive-conditional (SC)" simulator, which we thin to 10^5 draws.

We summarize the results of the tests in Table A-4, which shows the p-values associated with nine different test functions, and we provide additional visual

diagnostics in the form of Q-Q plots in Figure 5. The null hypothesis to which the p-values pertain is that the moment (test function) is equal in the distributions of the MC and SC draws. From the p-values given in the table it is apparent that this null hypothesis is not rejected for any of the test functions. The Q-Q plots show the quantiles from MC and SC simulators for each of the same test functions. Each dot in the plot pertains to a quantile, while the dashed red line is the "45-degree" line. If the two distributions are the same, then the dots should lie on the line. One can see in Figure 5 that all of the dots appear on the 45-degree line. With all p-values outside standard "significance" levels, and the dots in the Q-Q plots on top of the 45-degree line, we can be confident that our MCMC algorithm is implemented correctly.

TABLE A-4
"GETTING IT RIGHT" TEST OUTPUT FOR MCMC (ALGORITHM 5)

test function	p-value	test function	p-value	test function	p-value
$\mathbf{B}_{3,3}$	0.100	$\beta^2_{2,1}$	0.828	$v_{1,t=6}$	0.702
$\mathbf{B}^2_{3,3}$	0.823	$a_{3,2,t=7}$	0.507	$v^2_{1,t=6}$	0.209
$\beta_{2,1}$	0.834	$a^2_{3,2,t=7}$	0.191	$a_{3,2,t=7} \cdot v_{1,t=6}$	0.709

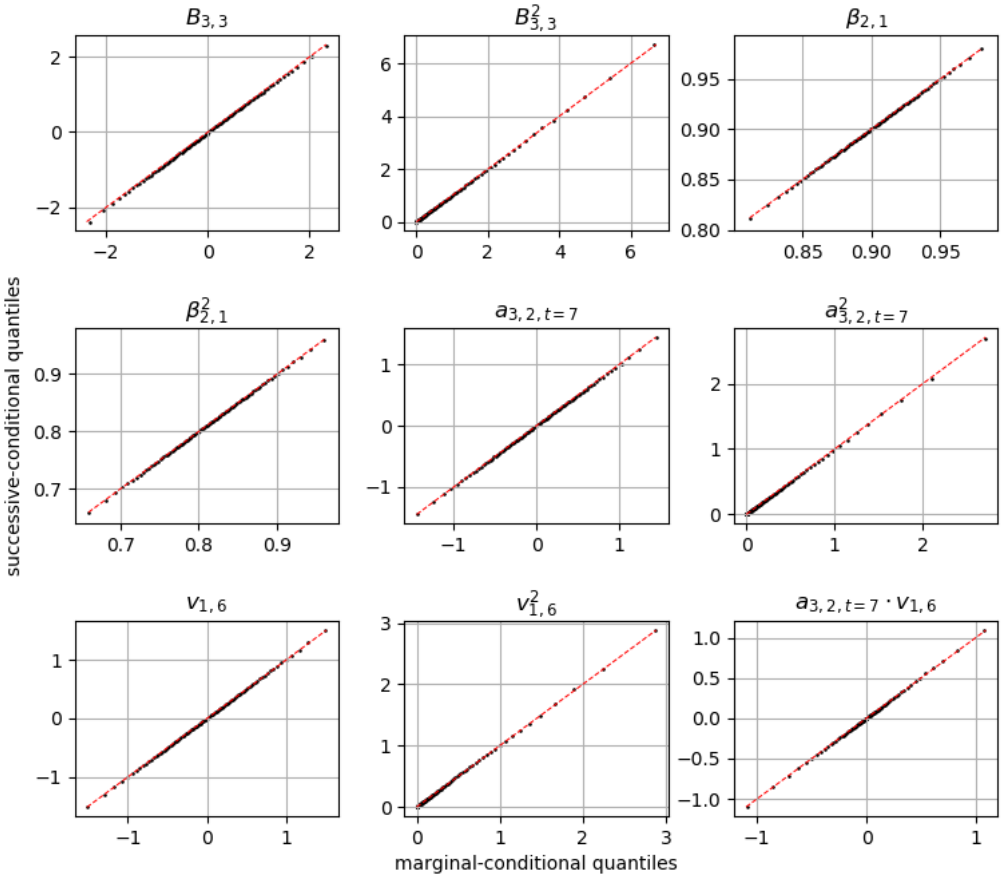

FIGURE 5.—Q-Q plots of test functions for "getting it right." Dots represent quantiles; red dashed line is the 45-degree line.

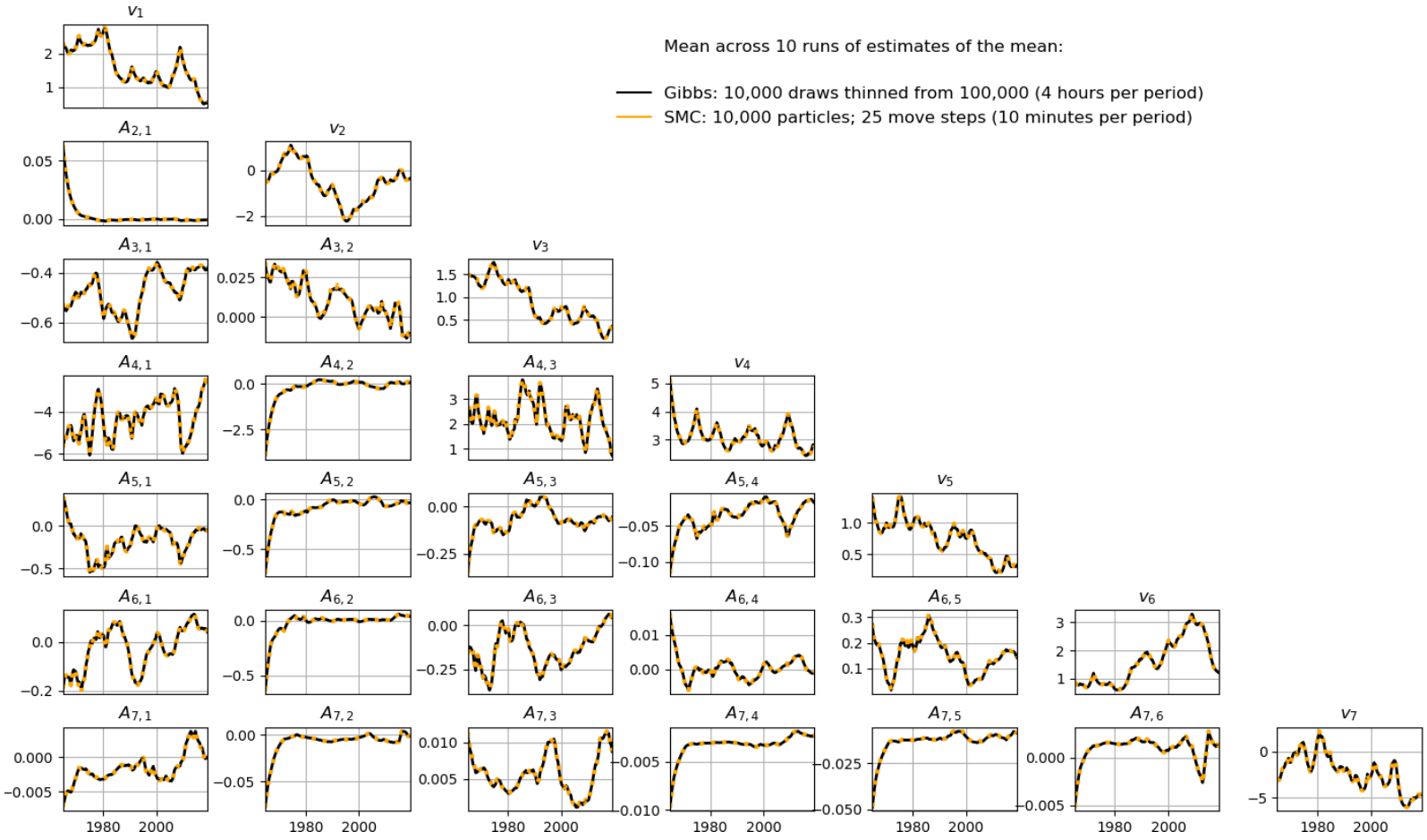

FIGURE 6.—Mean in each t: mean estimate across runs of MCMC and SMC

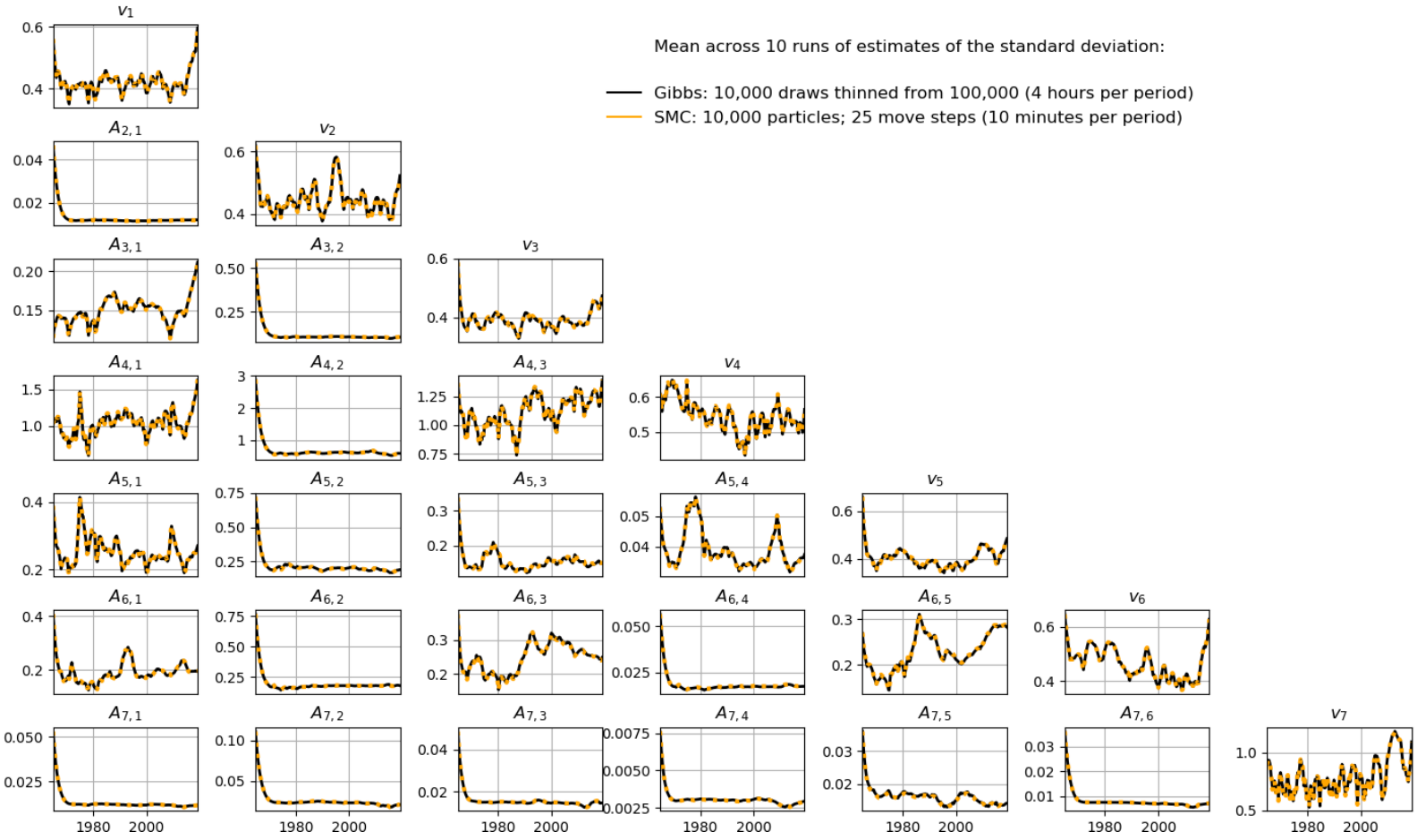

FIGURE 7.—Standard deviation in each t: mean estimate across runs of MCMC and SMC

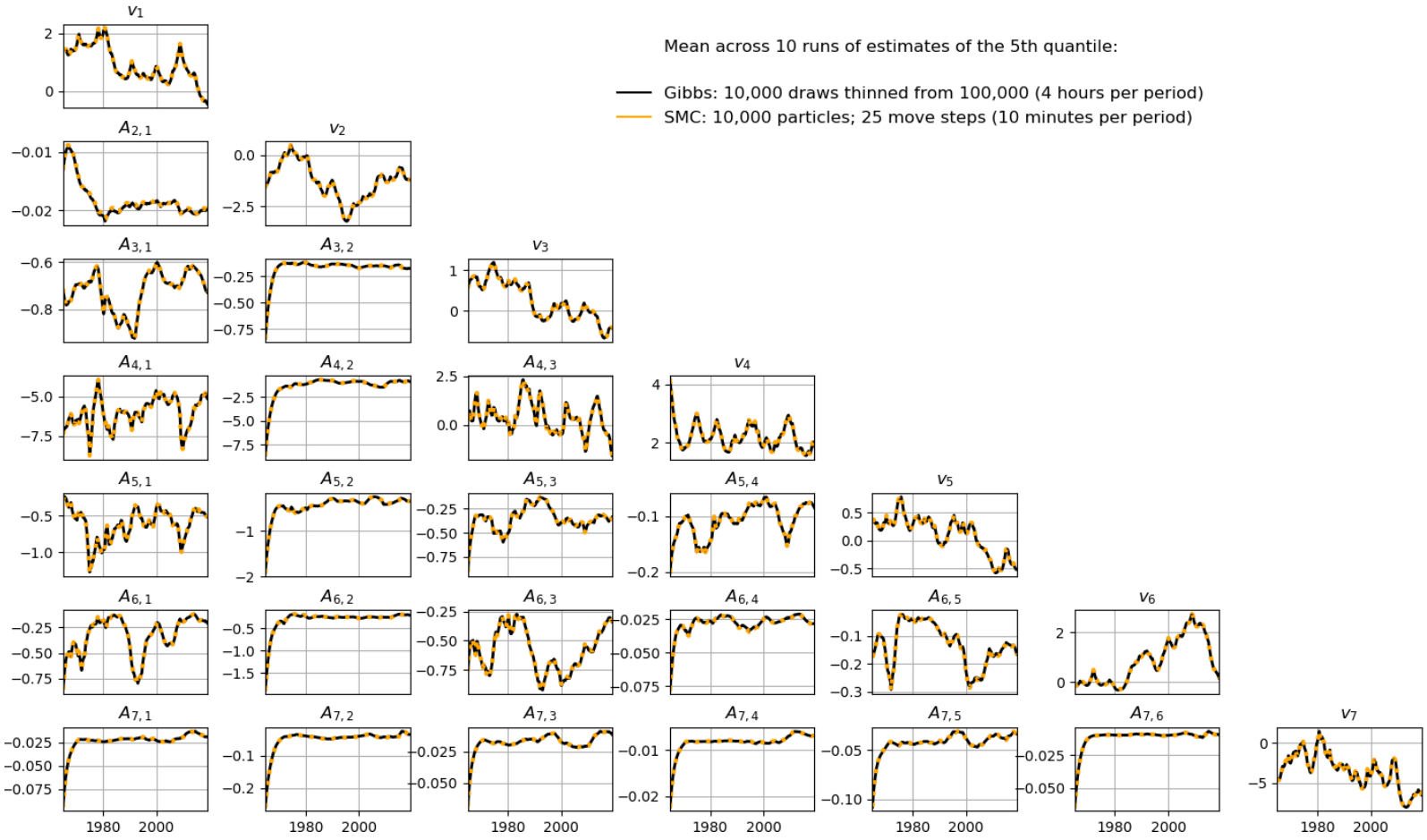

FIGURE 8.—5th quantile in each t: mean estimate across runs of MCMC and SMC

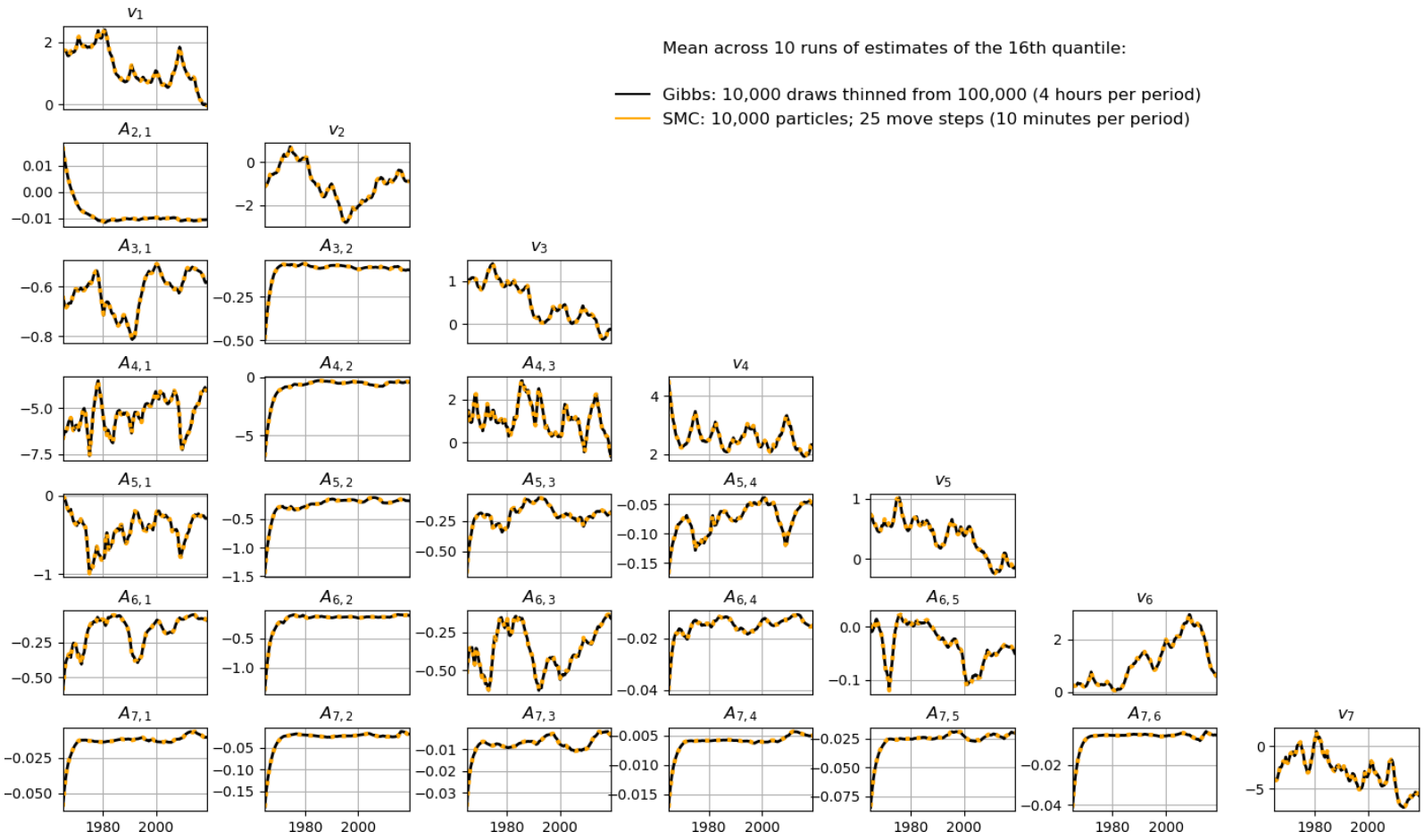

FIGURE 9.—16th quantile in each t: mean estimate across runs of MCMC and SMC

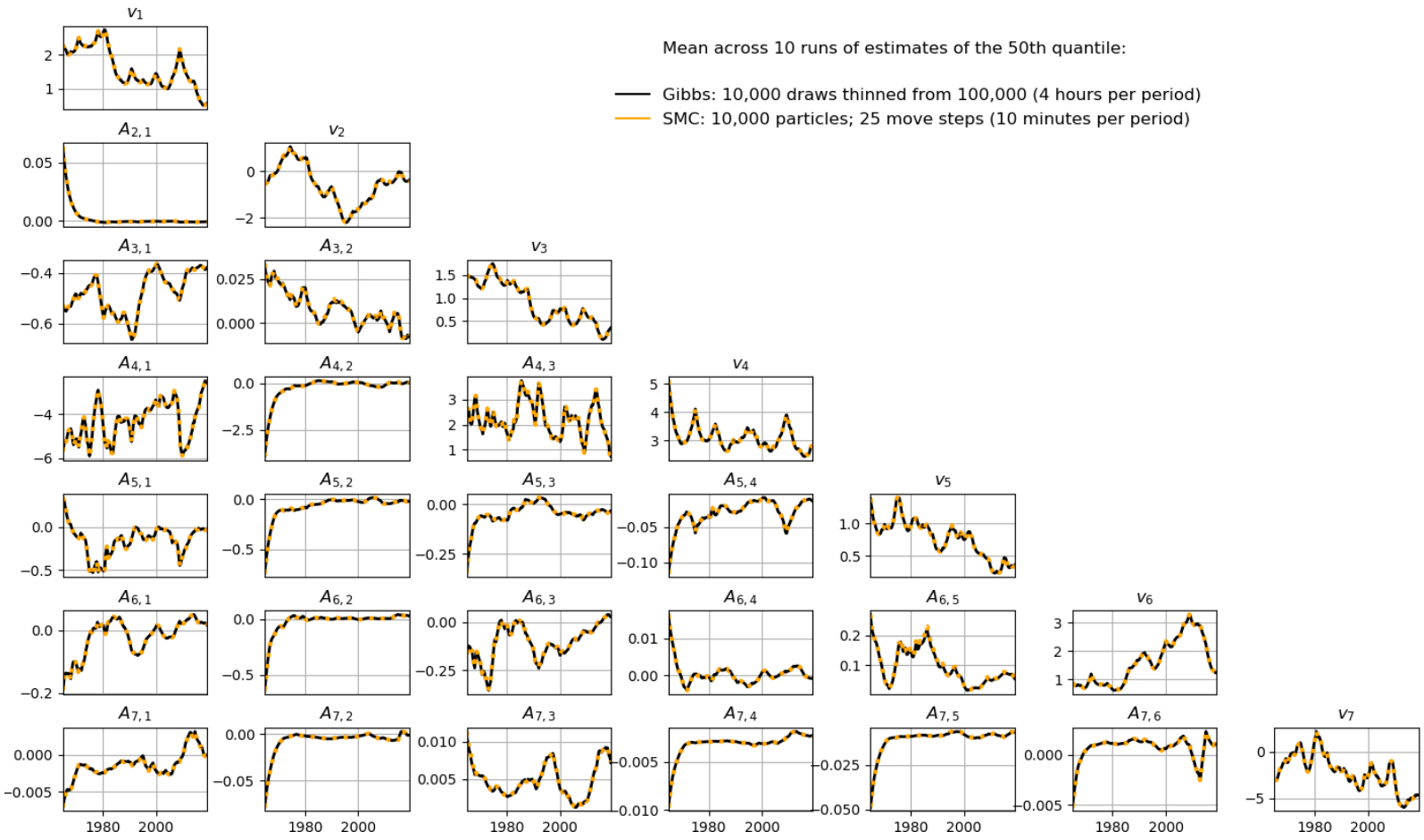

FIGURE 10.—50th quantile in each t: mean estimate across runs of MCMC and SMC

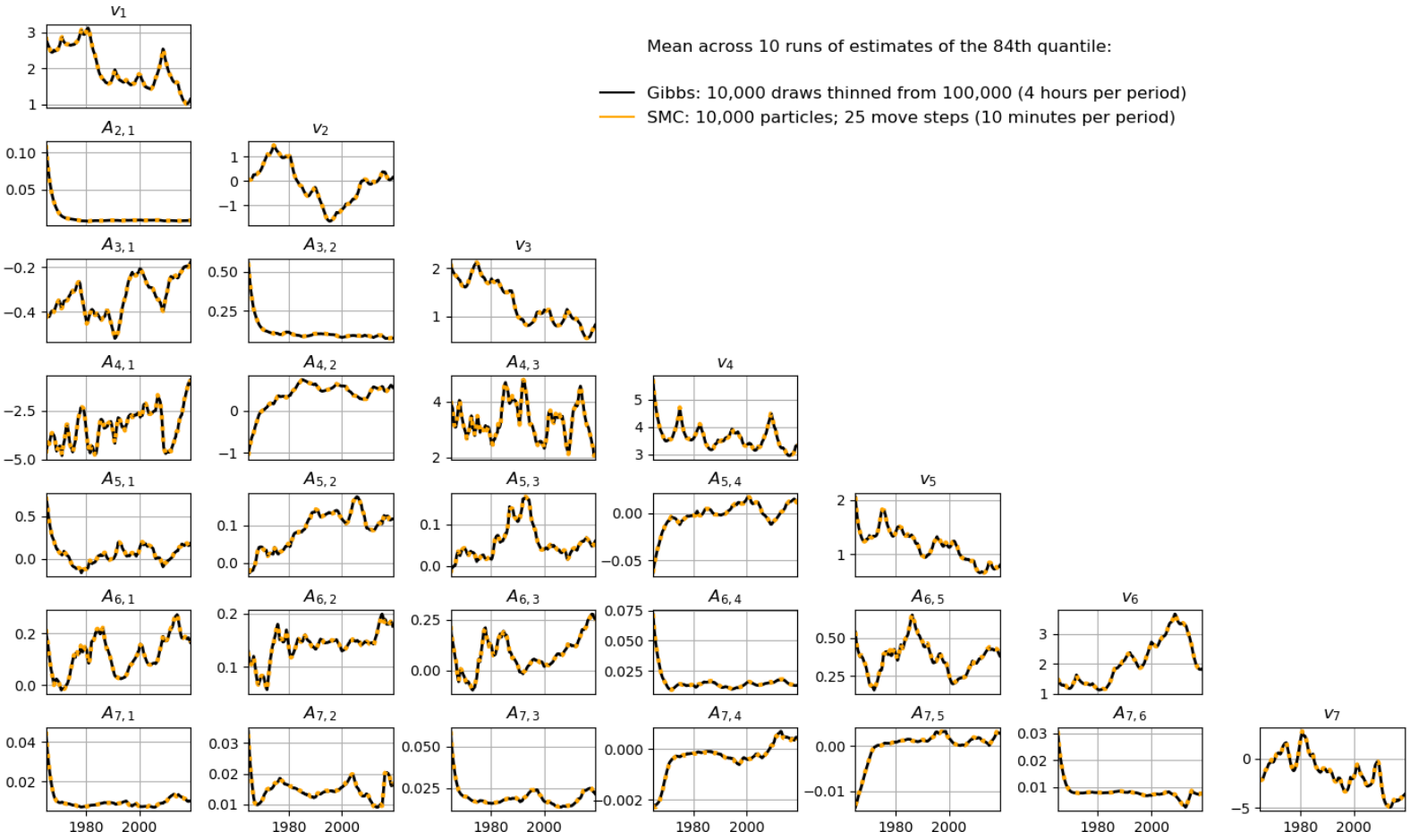

FIGURE 11.—84th quantile in each t: mean estimate across runs of MCMC and SMC

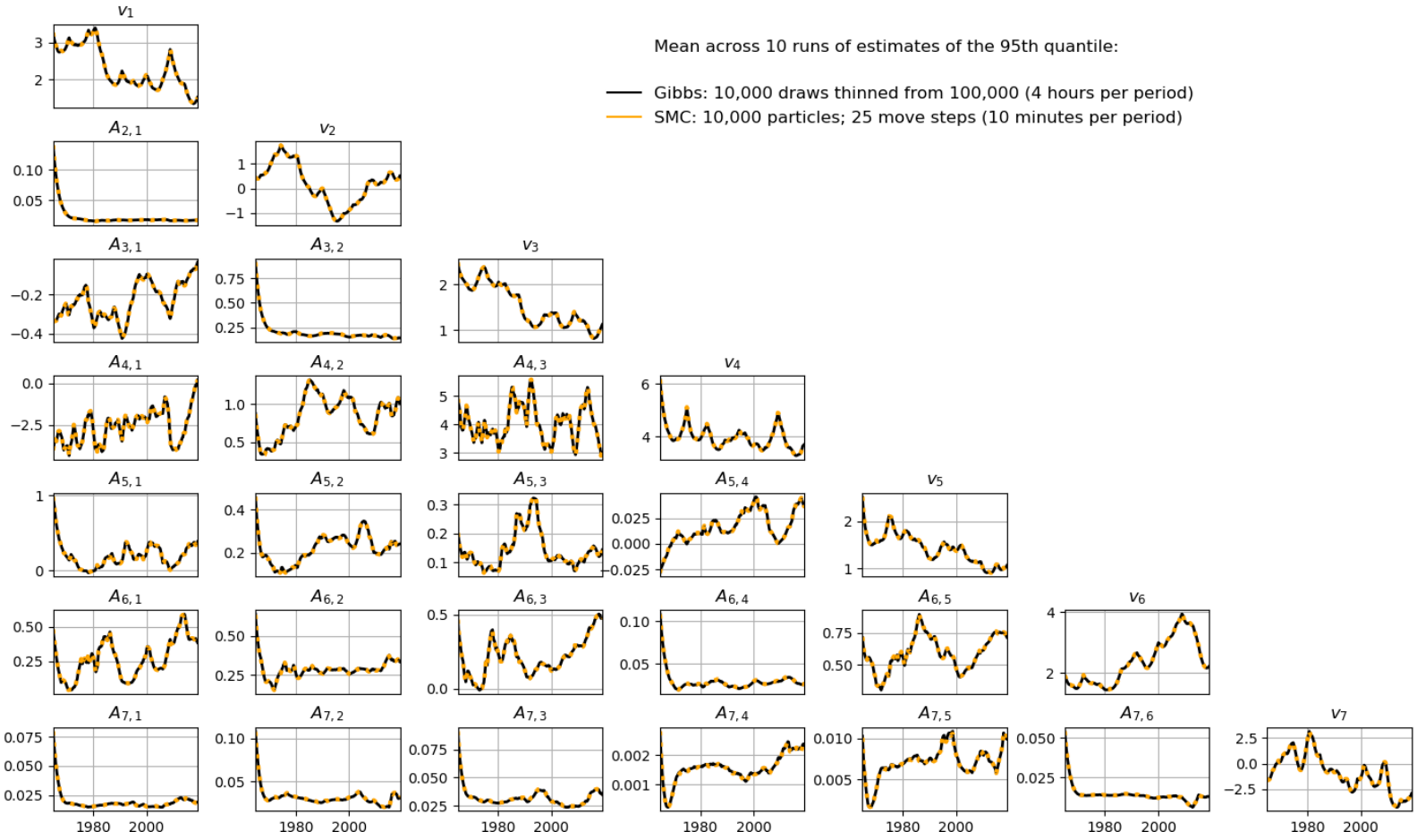

FIGURE 12.—95th quantile in each t: mean estimate across runs of MCMC and SMC

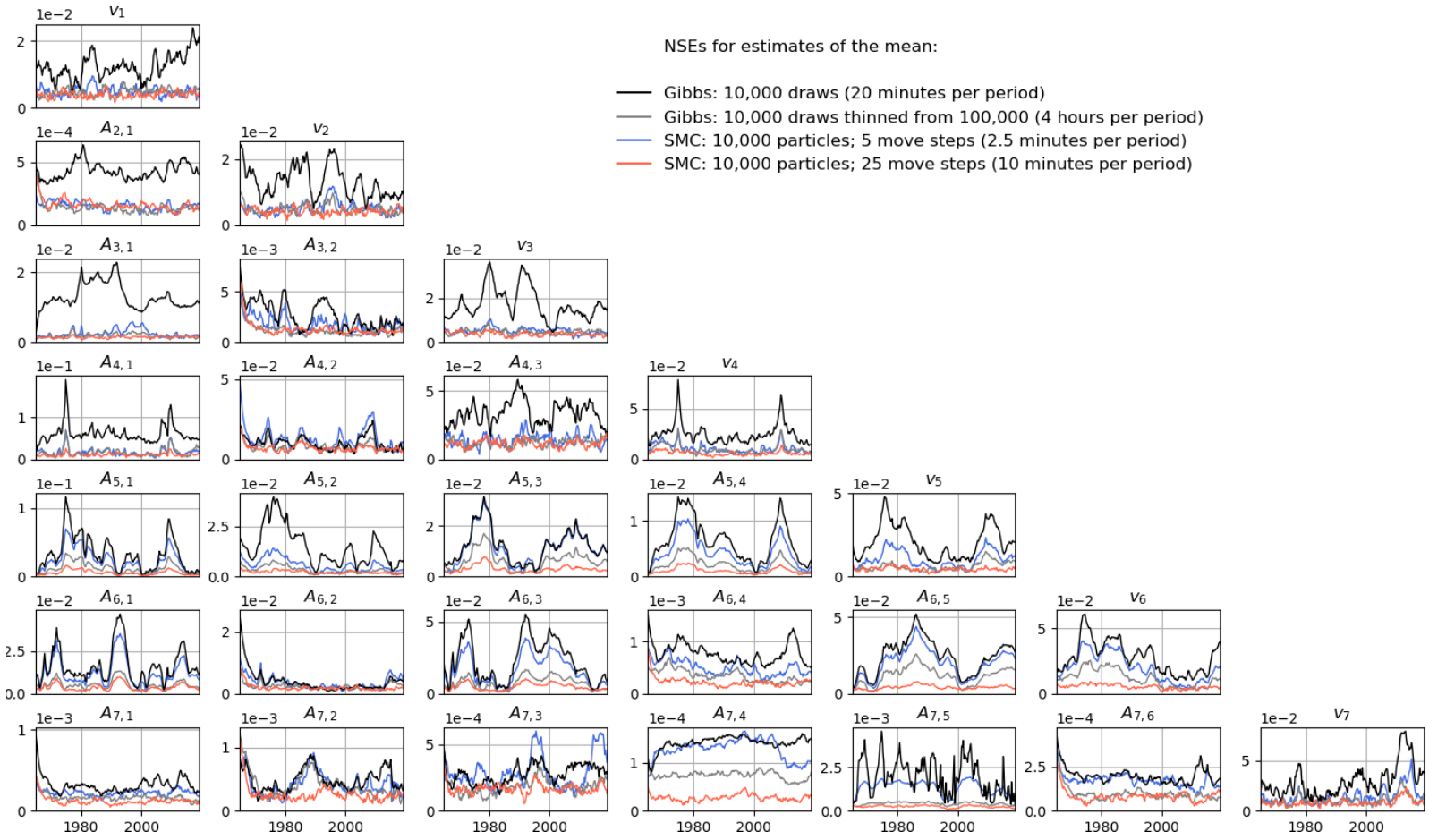

FIGURE 13.—NSEs of Posterior Means

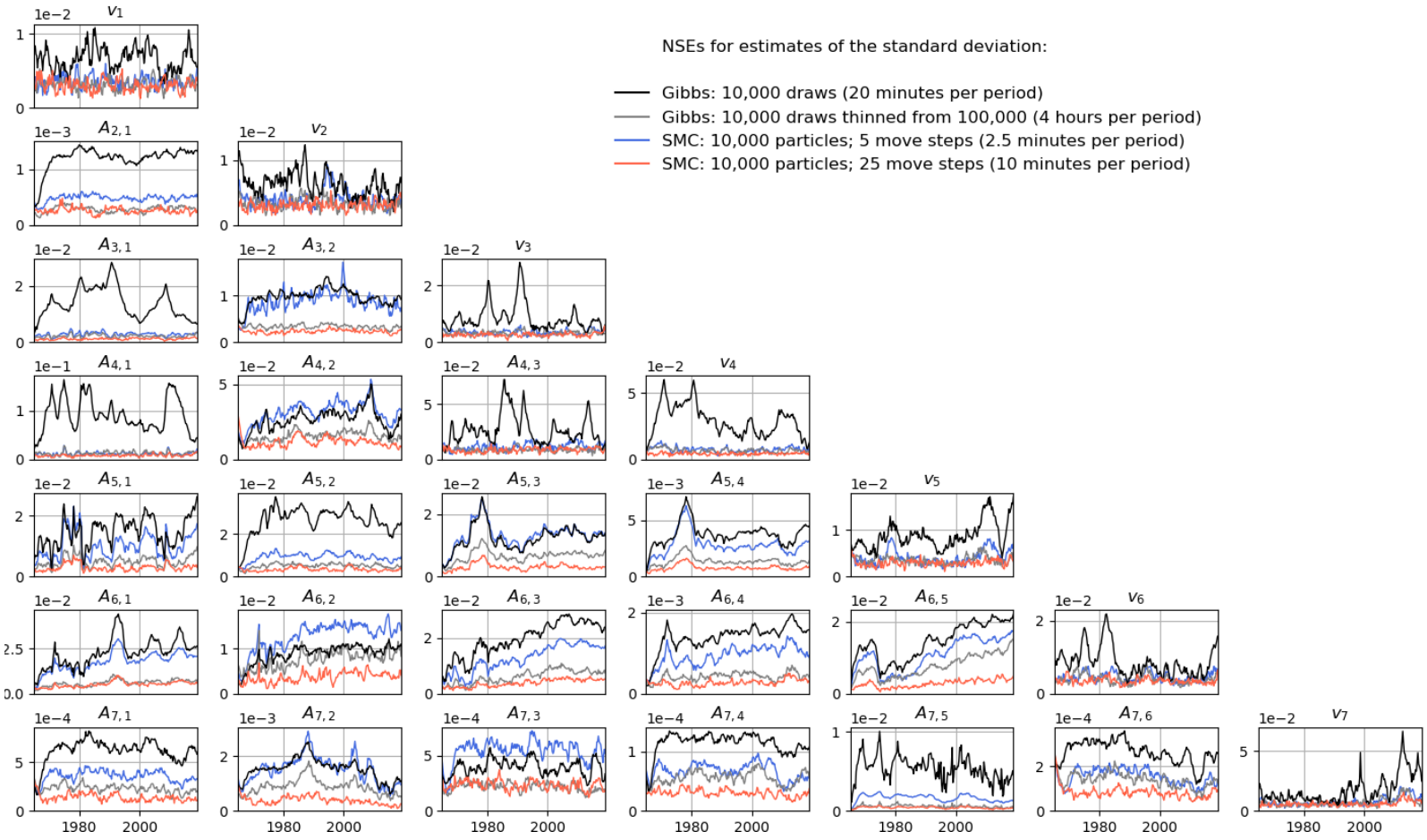

FIGURE 14.—NSEs of Posterior STDs

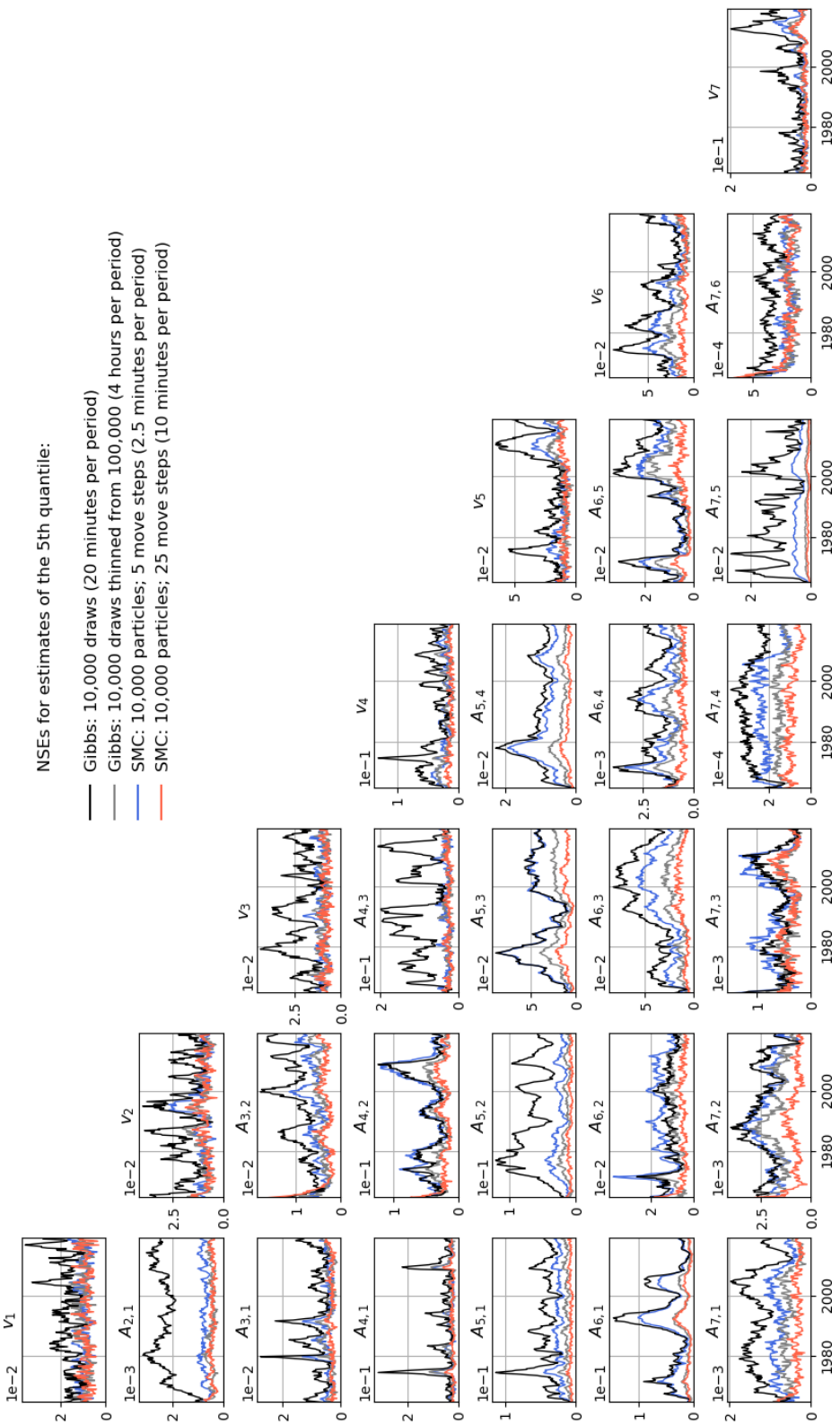

FIGURE 15.—NSEs of Posterior 5th Percentile

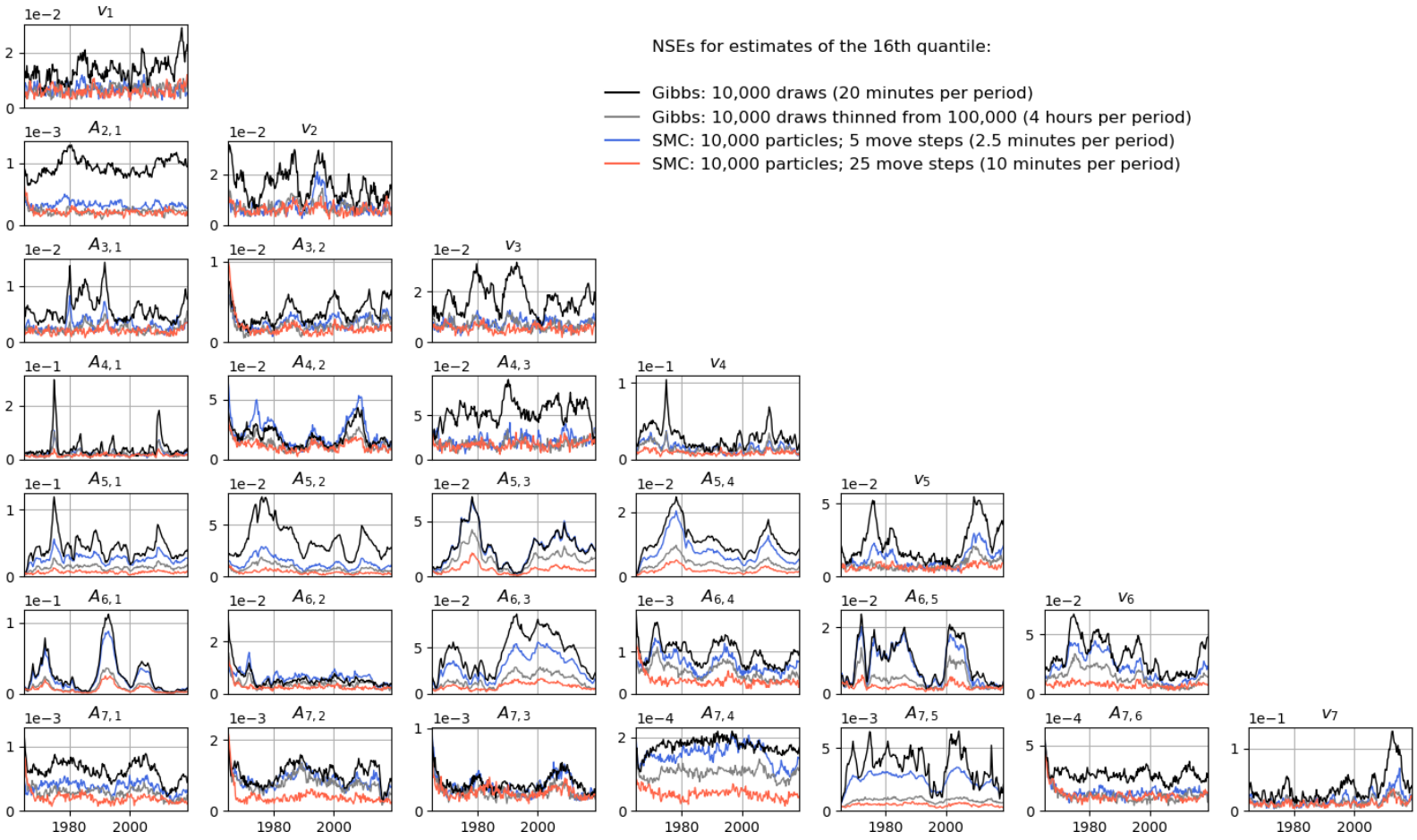

FIGURE 16.—NSEs of Posterior 16th Percentile

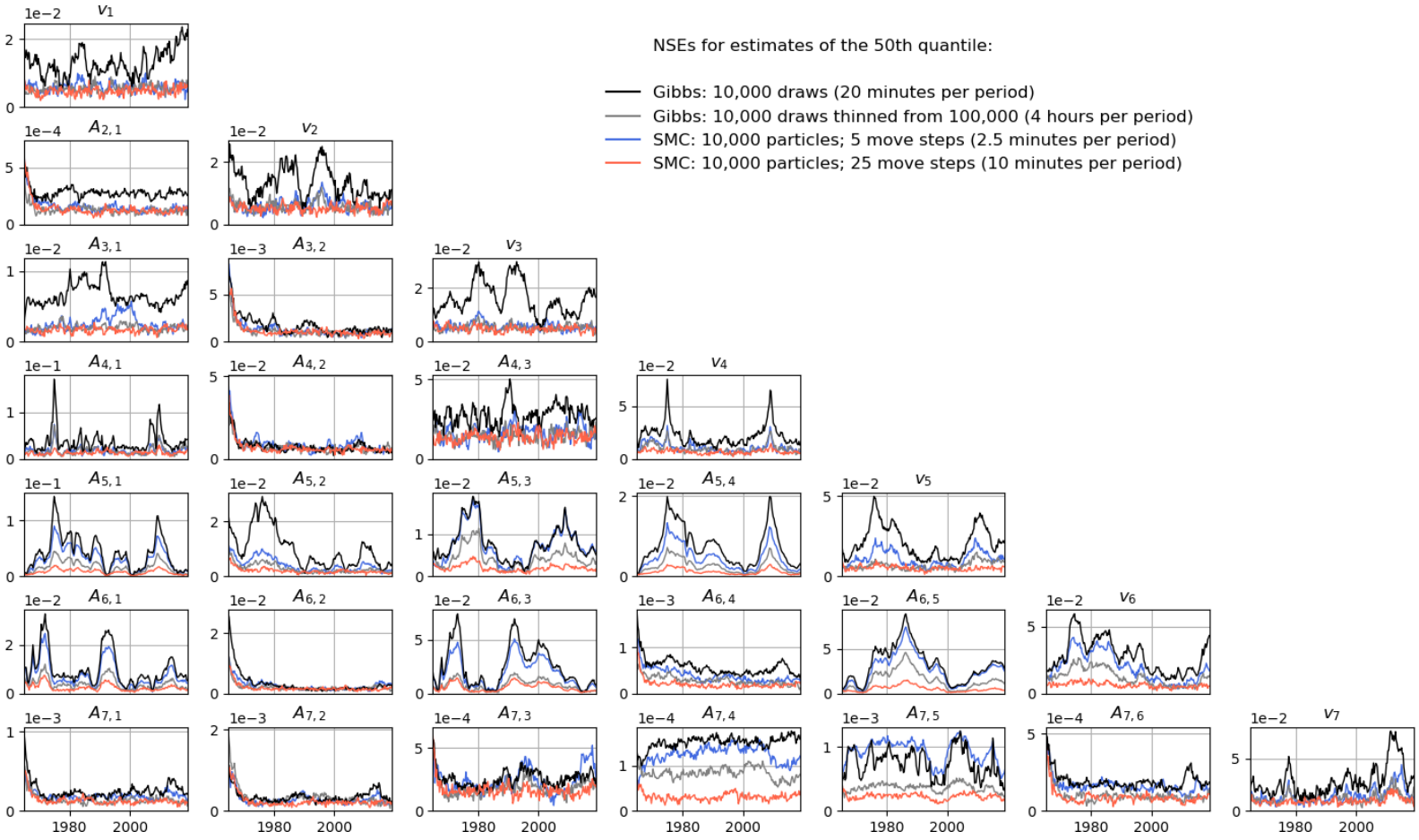

FIGURE 17.—NSEs of Posterior 50th Percentile

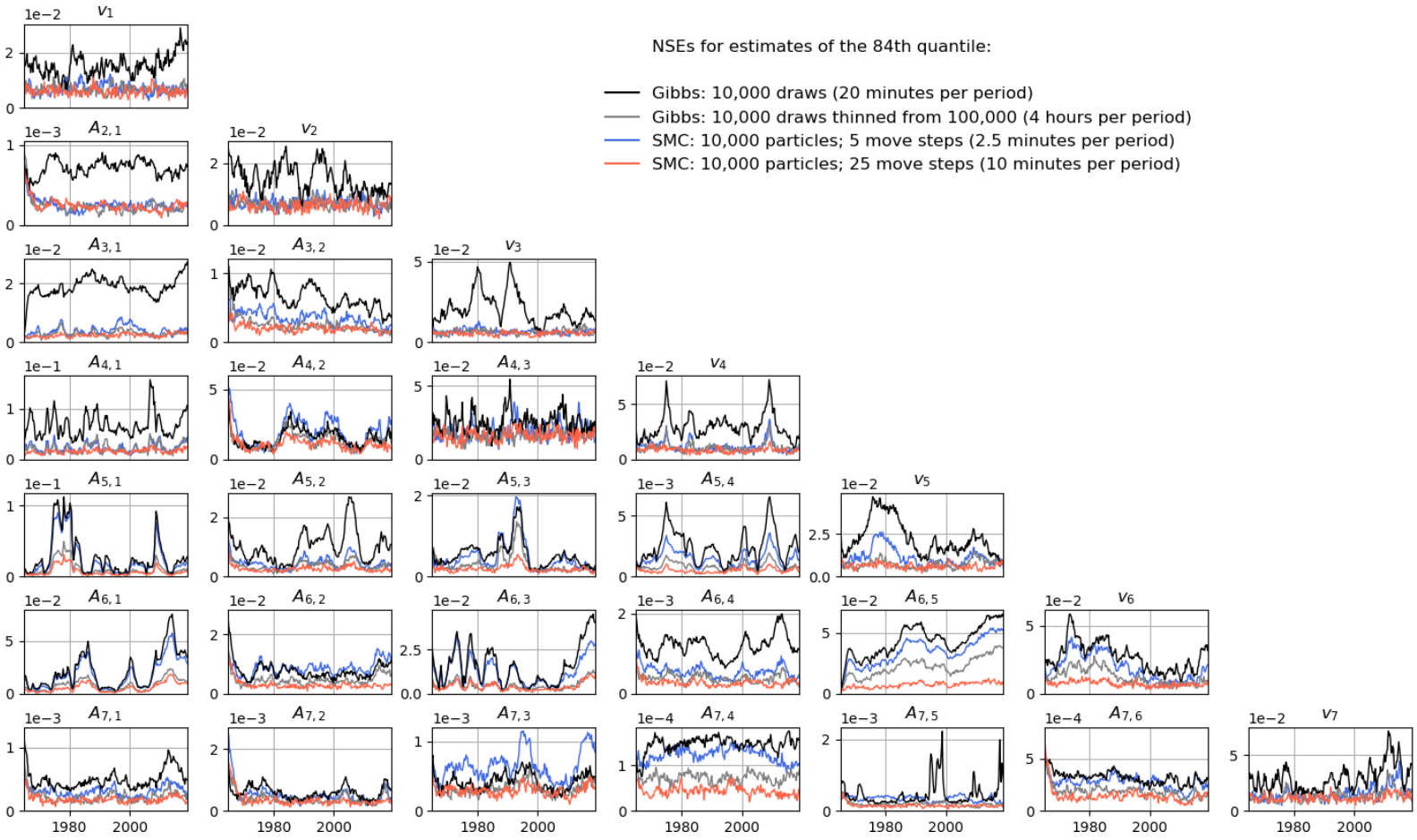

FIGURE 18.—NSEs of Posterior 84th Percentile

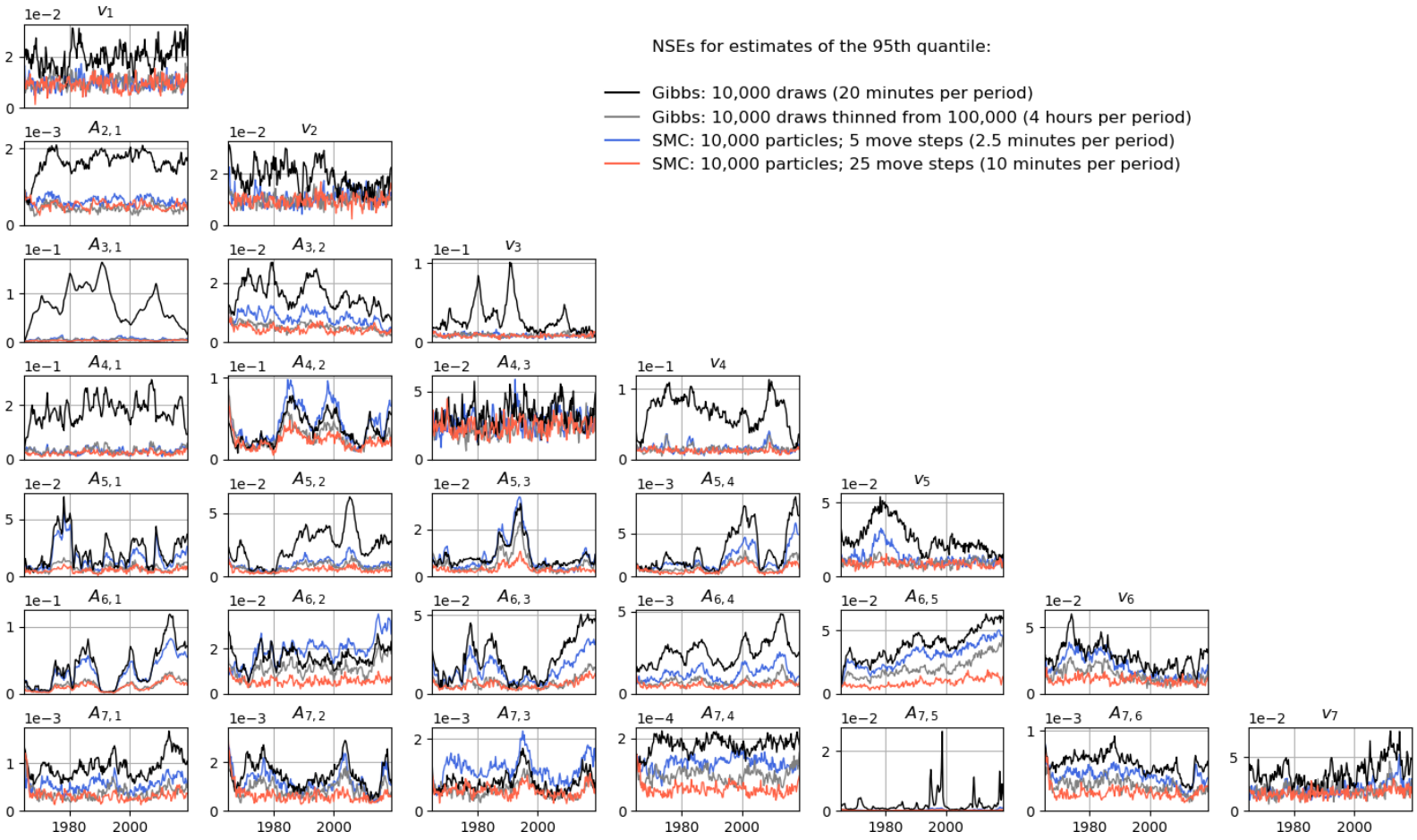

FIGURE 19.—NSEs of Posterior 95th Percentile

References

Aruoba, S. Borağan and Jesús Fernández-Villaverde (2015). "A comparison of programming languages in macroeconomics." *Journal of Economic Dynamics and Control*, 58, pp. 265 – 273. doi:https://doi.org/10.1016/j.jedc.2015.05.009.

Aruoba, S. Borağan and Jesús Fernández-Villaverde (2018). "A comparison of programming languages in macroeconomics: an update." *Mimeo, University of Pennsylvania*.

Berzuini, Carlo, Nicola G. Best, Walter R. Gilks, and Cristiana Larizza (1997). "Dynamic conditional independence models and markov chain monte carlo methods." *Journal of the American Statistical Association*, 92(440), pp. 1403–1412. doi:10.1080/01621459.1997.10473661.

Bezanson, J., A. Edelman, S. Karpinski, and V. Shah (2017). "Julia: A fresh approach to numerical computing." *SIAM Review*, 59(1), pp. 65–98. doi:10.1137/141000671.

Bognanni, Mark (2018). "A class of time-varying parameter structural vars for inference under exact or set identification." *Federal Reserve Bank of Cleveland, Working Paper no 18-11*. doi:https://doi.org/10.26509/frbc-wp-201811.

Bognanni, Mark and Edward Herbst (2017). "A sequential monte carlo approach to inference in multiple-equation markov-switching models." *Journal of Applied Econometrics*, 33(1), pp. 126–140. doi:10.1002/jae.2582.

Carriero, Andrea, Todd E. Clark, and Massimiliano Marcellino (2016). "Common drifting volatility in large bayesian vars." *Journal of Business & Economic Statistics*, 34(3), pp. 375–390. doi:10.1080/07350015.2015.1040116.

Chan, Joshua C. C. and Eric Eisenstat (2018). "Bayesian model comparison for time-varying parameter vars with stochastic volatility." *Journal of Applied Econometrics*, 33(4), pp. 509–532. doi:10.1002/jae.2617.

Chopin, N., P. E. Jacob, and O. Papaspiliopoulos (2013). "SMC2: An efficient algorithm for sequential analysis of state space models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3), pp. 397–426. doi:10.1111/j.1467-9868.2012.01046.x.

Chopin, Nicolas (2004). "Central limit theorem for sequential monte carlo methods and its application to bayesian inference." *The Annals of Statistics*, 32(6), pp. 2385–2411. doi:10.1214/009053604000000698.

Clark, Todd E. (2011). "Real-time density forecasts from bayesian vector autoregressions with stochastic volatility." *Journal of Business & Economic Statistics*, 29(3), pp. 327–341. doi:10.1198/jbes.2010.09248.

Clark, Todd E. and Francesco Ravazzolo (2015). "Macroeconomic forecasting performance under alternative specifications of time-varying volatility." *Journal of Applied Econometrics*, 30(4), pp. 551–575. doi:10.1002/jae.2379.

Creal, Drew (2012). "A survey of sequential monte carlo methods for economics and finance." *Econometric Reviews*, 31(3), pp. 245–296. doi:10.1080/07474938.2011.607333.

Del Moral, Pierre, Arnaud Doucet, and Ajay Jasra (2006). "Sequential monte carlo samplers." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3), pp. 411–436. doi:10.1111/j.1467-9868.2006.00553.x.

Del Moral, Pierre, Arnaud Doucet, and Ajay Jasra (2012). "On adaptive resampling strategies for sequential monte carlo methods." *Bernoulli*, 18(1), pp. 252–278. doi:10.3150/10-BEJ335.

Del Negro, Marco and Giorgio E. Primiceri (2015). "Time varying structural vector autoregressions and monetary policy: A corrigendum." *The Review of Economic Studies*, 82(4), pp. 1342–1345. doi:10.1093/restud/rdv024.

Djurić, P. M. and J. Miguez (2002). "Sequential particle filtering in the presence of additive gaussian noise with unknown parameters." In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. II–1621–II–1624. doi:10.1109/ICASSP.2002.5744928.

Douc, R. and O. Cappe (2005). "Comparison of resampling schemes for particle filtering." In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pp. 64–69. doi:10.1109/ISPA.2005.195385.

Doucet, Arnaud, Nando de Freitas, and Neil Gordon, editors (2001). *Sequential Monte Carlo in Practice*. Statistics for Engineering and Information Science. Springer. doi:10.1007/978-1-4757-3437-9.

Doucet, Arnaud and Adam Johansen (2011). "A tutorial on particle filtering and smoothing: Fifteen years later." In Dan Crisan and Boris Rozovskiĭ, editors, *Oxford Handbook of Nonlinear Filtering*. Oxford University Press.

Durham, Garland and John Geweke (2014). "Adaptive sequential posterior simulators for massively parallel computing environments." *Bayesian Model Comparison (Advances in Econometrics)*. doi:10.1108/s0731-905320140000034003.

Durham, Garland, John Geweke, Susan Porter-Hudak, and Fallaw Sowell (2019). "Bayesian inference for arfima models." *Journal of Time Series Analysis*, 40(4), pp. 388–410. doi:10.1111/jtsa.12443.

Fearnhead, Paul (2002). "Markov chain monte carlo, sufficient statistics, and particle filters." *Journal of Computational and Graphical Statistics*, 11(4), pp. 848–862. doi:10.1198/106186002835.

Fernández-Villaverde, Jesús and David Zarruk Valencia (2018). "A practical guide to parallelization in economics." Working Paper 24561, National Bureau of Economic Research. doi:10.3386/w24561.

Geweke, John (2004). "Getting it right." *Journal of the American Statistical Association*, 99(467), pp. 799–804. doi:10.1198/016214504000001132.

Giannone, Domenico, Michele Lenza, and Giorgio E. Primiceri (2015). "Prior selection for vector autoregressions." *The Review of Economics and Statistics*, 97(2), pp. 436–451. doi:10.1162/REST_a_00483.

Gilks, Walter R. and Carlo Berzuini (2001). "Following a moving target—monte carlo inference for dynamic bayesian models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1), pp. 127–146. doi:10.1111/1467-9868.00280.

Gordon, Neil J, David J Salmond, and Adrian FM Smith (1993). "Novel approach to nonlinear/non-gaussian bayesian state estimation." In *IEE Proceedings F (radar and signal processing)*, volume 140, pp. 107–113. IET. doi:10.1049/ip-rsn:19990255.

Herbst, Edward and Frank Schorfheide (2015). *Bayesian Estimation of DSGE Models*. Princeton University Press, Princeton. doi:10.23943/princeton/9780691161082.001.0001.

Kantas, Nikolas, Arnaud Doucet, Sumeetpal Singh, Jan Maciejowski, and Nicolas Chopin (2015). "On particle methods for parameter estimation in state-space models." *Statistical Science*, 30(3), pp. 328 – 351. doi:10.1214/14-STS511.

Kitagawa, Genshiro (1998). "A self-organizing state-space model." *Journal of the American Statistical Association*, 93(443), pp. 1203 – 1215. doi:10.2307/2669862.

Koop, Gary and Simon M. Potter (2007). "Estimation and Forecasting in Models with Multiple Breaks." *The Review of Economic Studies*, 74(3), pp. 763–789. doi:10.1111/j.1467-937X.2007.00436.x.

Liu, Jane and Mike West (2001). "Combined parameter and state estimation in simulation-based filtering." In *Sequential Monte Carlo Methods in Practice*. Springer. doi:10.1007/978-1-4757-3437-9_10.

Murray, Lawrence M., Anthony Lee, and Pierre E. Jacob (2016). "Parallel resampling in the particle filter." *Journal of Computational and Graphical Statistics*, 25(3), pp. 789–805. doi:10.1080/10618600.2015.1062015.

Omori, Yasuhiro, Siddhartha Chib, Neil Shephard, and Jouchi Nakajima (2007). "Stochastic volatility with leverage: Fast and efficient likelihood inference." *Journal of Econometrics*, 140(2), pp. 425 – 449. doi:https://doi.org/10.1016/j.jeconom.2006.07.008.

Primiceri, Giorgio E. (2005). "Time Varying Structural Vector Autoregressions and Monetary Policy." *The Review of Economic Studies*, 72(3), pp. 821–852. doi:10.1111/j.1467-937X.2005.00353.x.

Robert, Christian (2007). *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Science & Business Media.

Robert, Christian and George Casella (2004). *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer.

Robert, Christian P (1995). "Simulation of truncated normal variables." *Statistics and Computing*, 5(2), pp. 121–125. doi:10.1007/BF00143942.

Sims, Christopher A., Daniel F. Waggoner, and Tao Zha (2008). "Methods for inference in large multiple-equation markov-switching models." *Journal of Econometrics*, 146(2), pp. 255 – 274. doi:10.1016/j.jeconom.2008.08.023.

Sims, Christopher A. and Tao Zha (2006). "Were there regime switches in u.s. monetary policy?" *The American Economic Review*, 96(1), pp. 54–81. doi:10.1257/000282806776157678.

Smets, Frank and Rafael Wouters (2007). "Shocks and frictions in us business cycles: A bayesian dsge approach." *American Economic Review*, 97(3), pp. 586–606. doi:10.1257/aer.97.3.586.

Storvik, Geir (2002). "Particle filters for state-space models with the presence of unknown static parameters." *IEEE Transactions on Signal Processing*, 50(2), pp. 281–289. doi:10.1109/78.978383.