Bayesian Analysis (2016) **11**, Number 1, pp. 125–149

GPU-Accelerated Bayesian Learning and Forecasting in Simultaneous Graphical Dynamic Linear Models

Lutz Gruber* and Mike West†

Abstract. We discuss Bayesian analysis of dynamic models customized to learning and prediction with increasingly high-dimensional time series. A new framework of simultaneous graphical dynamic models allows the decoupling of analyses into those of a parallel set of univariate time series dynamic models, while flexibly modeling time-varying, cross-series dependencies and volatilities. The strategy allows for exact analysis of univariate time series models that are then coherently linked to represent the full multivariate model. Computation uses importance sampling and variational Bayes ideas, and is ideally suited to GPU-based parallelization. The analysis and its GPU-accelerated implementation is scalable with time series dimension, as we demonstrate in an analysis of a 400-dimensional financial time series.

Keywords: decoupling models, high-dimensional time series, importance sampling, parallel computing, recoupling models, variational Bayes.

1 Introduction

Scaling on-line Bayesian analyses of multivariate dynamic models to increasingly high-dimensions demands new modeling and computational strategies. A key need is for sequential analysis methods that are computationally accessible while applying to models of interest in many application areas – e.g., macroeconomics (Koop, 2012; Nakajima and West, 2013a), financial portfolio studies (Aguilar and West, 2000; Zhou et al., 2014), commercial and governmental forecasting (Agarwal et al., 2010; Queen, 1994; Anacleto et al., 2013), large scale-networks in energy demand forecasting (Hosking et al., 2013), and neuroscience (Trejo et al., 2007; Prado, 2010). This paper addresses this need.

We introduce *Simultaneous Graphical Dynamic Linear Models (SGDLMs)* and develop their Bayesian analyses utilizing GPU-based computation. Our strategy is to exploit fast, efficient sequential learning in a set of *decoupled* univariate dynamic linear models that capture cross-series contemporaneous dependencies via a sparse and dynamic simultaneous equations formulation. We then *recouple* these parallel univariate analyses using importance sampling-based reweighting of sets of direct simulations from the univariate models. The recoupling analysis defines coherent inference and forecasting of the multivariate series, and we then move ahead in time using decoupled models based on a variational Bayes strategy.

*Center for Mathematics, Technical University of Munich, Munich, Germany, gruber@ma.tum.de

†Department of Statistical Science, Duke University, Durham, NC, USA

 DOI: 10.1214/15-BA946

As defined in Section 2, SGDLMs consist of flexible state-space models of individual series. Separately, each is amenable to efficient, closed form sequential filtering and forecasting (Prado and West, 2010; West and Harrison, 1997). Then, SGDLMs integrate multivariate dependencies across series at each time point based on the *simultaneous equations* representation, defining a new model class in the growing field of sparse modeling of volatility matrices. Unlike Cholesky-style and factor models (e.g., Aguilar and West, 2000; Lopes et al., 2010; Pitt and Shephard, 1999; Lopes and Polson, 2010; Nakajima and West, 2013b), SGDLMs do not require an ordering of the series, which can be an obstacle to model specification. Compared to dynamic graphical models of precision matrices (e.g., Quintana and West, 1987; Carvalho and West, 2007; Wang and West, 2009), SGDLMs are scalable computationally, and flexible in allowing for patterns of change in volatility matrices that are specified implicitly via state-space models for sets of simultaneous regression parameters. Compared to standard multi- and matrix-variate DLMs (West and Harrison, 1997, Chapter 16.4), SGDLMs inherently define sparse representations underlying time-varying variance matrices, and allow for differing sets of predictors in each univariate series.

The recoupling analysis uses exact, within-series simulation of states and parameters combined with importance sampling (IS) reweighting for inference and prediction of the multivariate series. We then exploit an efficient and accurate variational Bayes (VB) strategy (e.g., Jaakkola and Jordan, 2000; Wand et al., 2011, and West and Harrison, 1997, Section 12.3.4) to decouple and proceed to the next time point. Importantly, we monitor and adjust for the VB approximation using information from the IS step.

A main interest is in scaling-up in the number of time series. We exploit (C++/CUDA) GPU computation (Suchard et al., 2010; Lee et al., 2010) that is ideally suited to the analysis. Univariate time series model updates and simulations are performed in parallel at each time point, and then recoupled for coherent inference and forecasting before decoupling again to move to the next time point and re-parallelization. The overall modeling and computational strategy is scalable with time series dimension as a result. We discuss computational demands with theoretical and empirical benchmarks, illustrate the GPU implementation, and demonstrate how the implementation enables real-time Bayesian analysis of a 400-dimensional daily stock return time series.

2 SGDLMs

2.1 Model Structure: Definitions and Notation

Series-Specific Form Consider the m-dimensional time series $\mathbf{y}_t = (y_{1t}, \ldots, y_{mt})'$, $t = 1, 2, \ldots$. Each univariate series y_{jt} is represented via a linear, normal state-space model with a traditional variance discount model of stochastic volatility (West and Harrison, 1997; Prado and West, 2010). With the convention that all vectors are columns, the basic model form is:

$$y_{jt} = \mathbf{F}'_{jt}\boldsymbol{\theta}_{jt} + \nu_{jt} = \mathbf{x}'_{jt}\boldsymbol{\phi}_{jt} + \mathbf{y}'_{sp(j),t}\boldsymbol{\gamma}_{jt} + \nu_{jt}, \tag{1}$$

$$\boldsymbol{\theta}_{jt} = \mathbf{G}_{jt}\boldsymbol{\theta}_{j,t-1} + \boldsymbol{\omega}_{jt}, \tag{2}$$

where the observation error $\nu_{jt} \sim N(0, \lambda_{jt}^{-1})$ and state evolution error $\boldsymbol{\omega}_{jt} \sim N(\mathbf{0}, \mathbf{W}_{jt})$ are independent, zero mean normals and are independent of all past such terms and across all series $i \neq j$. Predictor vectors $\mathbf{x}_{jt}$ and $\mathbf{y}_{sp(j),t}$ are catenated to define $\mathbf{F}'_{jt} = (\mathbf{x}'_{jt},\ \mathbf{y}'_{sp(j),t})$; corresponding state vectors $\boldsymbol{\phi}_{jt}$ and $\boldsymbol{\gamma}_{jt}$ similarly define $\boldsymbol{\theta}'_{jt} = (\boldsymbol{\phi}'_{jt},\ \boldsymbol{\gamma}'_{jt})$. Here $\mathbf{x}_{jt}$ is a series-specific vector of exogenous predictors, while $\mathbf{y}_{sp(j),t}$ is a vector of contemporaneous values of some of the other series, indexed by $sp(j) \subseteq \{1{:}m\} \setminus \{j\}$ and called the *simultaneous parental set* for series j. State vector $\boldsymbol{\phi}_{jt}$ has dimension $p_{j\phi}$, state vector $\boldsymbol{\gamma}_{jt}$ has dimension $p_{j\gamma} = |sp(j)|$, so that $\boldsymbol{\theta}_{jt}$ has dimension $p_j = p_{j\phi} + p_{j\gamma}$. The model context assumes the parental sets, and hence their dimensions, are fixed over time (at least, fixed over the period of time chosen for analysis) as part of the model specification. Conditional on the state vector and exogenous predictors, write $\mu_{jt} = \mathbf{x}'_{jt}\boldsymbol{\phi}_{jt}$. The state vector evolves according to the linear evolution with state matrix $\mathbf{G}_{jt}$. The states $\boldsymbol{\theta}_{jt}$ and λ_{jt} are learned sequentially, while all other quantities are specified through prior or model choices.

Equations (1) and (2) define a set of coupled, dynamic simultaneous equations that cohere across $j = 1{:}m$, representing a set of dynamic structural equations for the multivariate model of $\mathbf{y}_t$ (e.g., Palomo et al., 2007, and references therein).

Across-Series Form Define $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_{1t}, \ldots, \boldsymbol{\theta}_{mt}\}$ and $\boldsymbol{\Lambda}_t = \{\lambda_{1t}, \ldots, \lambda_{mt}\}$, the sets of m state vectors and precisions, and

$$\boldsymbol{\mu}_t = \begin{pmatrix} \mu_{1t} \\ \mu_{2t} \\ \vdots \\ \mu_{m-1,t} \\ \mu_{mt} \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Gamma}_t = \begin{pmatrix} 0 & \gamma_{1,2,t} & \gamma_{1,3,t} & \cdots & \gamma_{1,m,t} \\ \gamma_{2,1,t} & 0 & \gamma_{2,3,t} & \cdots & \gamma_{2,m,t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma_{m-1,1,t} & \cdots & \gamma_{m-1,m-2,t} & 0 & \gamma_{m-1,m,t} \\ \gamma_{m,1,t} & \gamma_{m,2,t} & \cdots & \gamma_{m,m-1,t} & 0 \end{pmatrix}$$

where we extend the γ_* notation so that $\gamma_{jht} = 0$ for each $h \notin sp(j)$, $j = 1{:}m$.

It follows that

$$\mathbf{y}_t \sim N(\mathbf{A}_t\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \tag{3}$$

where

$$\mathbf{A}_t = (\mathbf{I} - \boldsymbol{\Gamma}_t)^{-1} \quad \text{and} \quad \boldsymbol{\Omega}_t \equiv \boldsymbol{\Sigma}_t^{-1} = (\mathbf{I} - \boldsymbol{\Gamma}_t)'\boldsymbol{\Lambda}_t(\mathbf{I} - \boldsymbol{\Gamma}_t). \tag{4}$$

Practical models will typically have small parental sets $sp(j)$ so the resulting $\boldsymbol{\Gamma}_t$ matrix will be sparse. That, coupled with the state evolution models for the $\boldsymbol{\gamma}_{jt}$, defines a flexible class of multivariate volatility models for the implied variance matrix $\boldsymbol{\Sigma}_t$ and its inverse – the precision matrix – $\boldsymbol{\Omega}_t$. Very sparse $\boldsymbol{\Gamma}_t$ can imply (albeit less) sparse precision matrices; the correspondence of zeros in $\boldsymbol{\Omega}_t$ with conditional independencies in the resulting Gaussian graphical models (Carvalho and West, 2007) underlie the designation of this class of models as *simultaneous graphical dynamic linear models*. With even modest m, practical models will have relatively small parental sets. If the maximum parental set size is k, the model has mk nonzero elements in $\boldsymbol{\Gamma}_t$ so that $k < (m-1)/2$ means $\boldsymbol{\Omega}_t$ is not over-parametrized. Our motivation for these models and their expected utility is in problems with increasingly large m; our financial time series example in Section 5 has $m = 400, k = 10$, so represents 79,800 precision parameters in terms of just 4,000 simultaneous parental parameters.

We note connections with the use of simultaneous/structural specifications in spatial analysis on lattice data. Simultaneous autoregressive (SAR) models define joint distributions of outcomes on a spatial lattice via univariate conditional models based on sparse simultaneous parental (or neighboring) sets in a form similar to that adopted here (e.g., Anselin, 1988; Oliveira and Song, 2008; Whittle, 1954). Part of the inspiration for the work here derives from the utility of such models in spatial studies, especially with regard to scalability to larger problems (Mukherjee et al., 2014).

2.2 Sequential Learning: Structure and Challenges

Sequential analysis moves over time t and updates summary posterior distributions for model state vectors and precisions as new data is observed. At time $t-1$, denote historical data and information by $\mathcal{D}_{t-1}$. Evolving to time t, this information set updates to $\mathcal{D}_t = \{\mathbf{y}_t,\ \mathcal{I}_t,\ \mathcal{D}_{t-1}\}$ where $\mathcal{I}_t$ denotes any additional information or model changes used between times $t-1$ and t, for example, the specification of the state evolution matrices $\mathbf{G}_{jt}$, evolution variances $\mathbf{W}_{jt}$, or changes in the simultaneous parental sets $sp(j)$ (West and Harrison, 1989, 1997, Chapter 11). Then, analysis over times $t-1$ to t involves: (i) using the time $t-1$ posterior $p(\boldsymbol{\Theta}_{t-1}, \boldsymbol{\Lambda}_{t-1}|\mathcal{D}_{t-1})$ to infer the prior for time t, namely $p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{I}_t, \mathcal{D}_{t-1})$; (ii) using this prior to compute forecast distributions for $\mathbf{y}_t$ and future outcomes beyond time t, as desired; (iii) on moving to time t, updating to the current posterior $p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{D}_t)$.

The full multivariate model raises computational challenges due to the nonlinearities in state vectors in (3) and (4). The time t likelihood function for $\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t$ is directly derived from the m-variate normal density of (3) as

$$p(\mathbf{y}_t|\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t) \propto |\mathbf{I} - \boldsymbol{\Gamma}_t| \prod_{j=1:m} p(y_{jt}|\boldsymbol{\theta}_{jt}, \lambda_{jt}) \tag{5}$$

where the product is of normal densities from the set of univariate models, namely $y_{jt} \sim N(\mathbf{F}'_{jt}\boldsymbol{\theta}_{jt}, \lambda_{jt}^{-1})$ for $j = 1{:}m$. As a result, the time t updated posterior is

$$p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{D}_t) \propto |\mathbf{I} - \boldsymbol{\Gamma}_t|\ p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{I}_t, \mathcal{D}_{t-1}) \prod_{j=1:m} p(y_{jt}|\boldsymbol{\theta}_{jt}, \lambda_{jt}). \tag{6}$$

The determinant factor here induces the computational challenges. Apart from the special cases of *compositional models* (Nakajima and West, 2013a; Zhao and West, 2014) in which $\boldsymbol{\Gamma}_t$ is triangular with a diagonal of zeros and so $|\mathbf{I} - \boldsymbol{\Gamma}_t| = 1$, this determinant term contributes to the likelihood for the $\boldsymbol{\gamma}_{jt}$ vectors. In very sparse models, the determinant term can tend to be quite diffuse as a function of $\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t$ when compared to the product of individual likelihood terms. However, it matters generally; it arises theoretically to ensure positive definiteness and symmetry of $\boldsymbol{\Omega}_t$. In our financial time series example in Section 5, we demonstrate some of the negative practical consequences of ignoring this term.

3 Model Decoupling/Recoupling Strategy

3.1 Motivation and Summary

The forms of (5) and (6) suggest opportunity to exploit separate, parallel analyses of each univariate series in order to define a sequential computational strategy for the full multivariate model. Each univariate DLM of (1) and (2) is a linear, normal state-space model for $\boldsymbol{\theta}_{jt}$ that, when coupled with the traditional variance discount model for stochastic volatilities $\sqrt{\lambda_{jt}^{-1}}$, is amenable to standard forward filtering and forecasting analysis in closed form (West and Harrison, 1997). Directly implemented for series j without regard to the other univariate models, this analysis involves sequentially updated normal/gamma priors $p(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{D}_{t-1})$, and posteriors $p(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{D}_t)$, with simple, closed-form updates. Were we to simply use these separate univariate DLMs in parallel and assume independence across series j, then the implied joint priors and posteriors for the $\{\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t\}$ would factorize. Thus, the naive approximation of ignoring the determinant factor in (5) and (6) is akin to running a univariate DLM on each series individually. We use this idea to define a computational strategy that improves on this naive approach while retaining the analytical tractability of the time evolution and update steps.

The steps involved in our *decoupling/recoupling* strategy are fully detailed in Section 3.2. In summary here, standing at time t before observation of $\mathbf{y}_t$, the analysis proceeds as follows:

A. At time t, adopt *decoupled* priors $p(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{I}_t, \mathcal{D}_{t-1})$ assumed independent over $j = 1{:}m$.

B. To predict $\mathbf{y}_{t+k}$ into the future $k = 0, 1, \ldots$, simulate these independent priors and use sampled values to evaluate aspects of full multivariate forecast distributions.

C. At time t on observing $\mathbf{y}_t$, perform parallel, independent updates to posteriors in the m DLMs and take their product to yield a *naive* posterior approximation $\tilde{p}(\boldsymbol{\Theta}_{jt}, \boldsymbol{\Lambda}_{jt}|\mathcal{D}_t)$.

D. *Recouple* the analyses by evaluating the exact posterior using importance sampling.

E. *Decouple* the series by emulating the exact posterior by a product of margins over $j = 1{:}m$ using variational Bayes.

F. Apply state evolutions independently over $j = 1{:}m$ to move to time $t + 1$.

Note that, in special cases when $\boldsymbol{\Gamma}_t$ is chosen to be – or just happens to be – diagonal, we have a compositional (directed) graphical model specification, with $|\mathbf{I} - \boldsymbol{\Gamma}_t| = 1$. The analysis is then closed-form: the naive posterior from C equals the exact posterior, and steps D and E can be omitted. In the kinds of practical problems of focus – with more than a few series – the compositional approach is typically a non-starter since it requires that the modeler can define a strict ordering of the series. While this can be done based

on substantive reasoning with relatively few series, choosing an ordering is otherwise challenging and arbitrary, and results heavily dependent on its choice.

3.2 Model Emulation: Decoupling for Forward Filtering

A. Time t Prior The prior at time t for model states and volatilities is a product of a decoupled set of m conjugate normal/gamma priors (see Appendix A), namely

$$p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t | \mathcal{I}_t, \mathcal{D}_{t-1}) = \prod_{j=1:m} p_{jt}(\boldsymbol{\theta}_{jt}, \lambda_{jt} | \mathcal{I}_t, \mathcal{D}_{t-1}) \tag{7}$$

where $p_{jt}(\cdot|\cdot)$ denotes the density function of

$$(\boldsymbol{\theta}_{jt}, \lambda_{jt} | \mathcal{I}_t, \mathcal{D}_{t-1}) \sim NG(\mathbf{a}_{jt}, \mathbf{R}_{jt}, r_{jt}, c_{jt}). \tag{8}$$

This uses standard notation (Prado and West, 2010) for the normal/gamma

$$(\boldsymbol{\theta}_{jt} | \lambda_{jt}, \mathcal{I}_t, \mathcal{D}_{t-1}) \sim N(\mathbf{a}_{jt}, \mathbf{R}_{jt}/(c_{jt}\lambda_{jt})),$$
$$(\lambda_{jt} | \mathcal{I}_t, \mathcal{D}_{t-1}) \sim G(r_{jt}/2, r_{jt}c_{jt}/2),$$

in which $N(\mathbf{a}, \mathbf{A})$ is multivariate normal with mean $\mathbf{a}$ and variance matrix $\mathbf{A}$, and $G(r, rc)$ is gamma with shape r, scale rc, and mean $r/(rc) = 1/c$. The implied $\boldsymbol{\theta}_{jt}$ margin of (8) is multivariate T with r_{jt} degrees of freedom, mode $\mathbf{a}_{jt}$ and scale matrix $\mathbf{R}_{jt}$; the marginal variance matrix is $\mathbf{R}_{jt} r_{jt}/(r_{jt} - 2)$ in usual cases that $r_{jt} > 2$.

B. Time t Predictions The one-step ahead predictive distribution is efficiently simulated by drawing from the set of m independent normal/gamma priors above, so defining a simulation sample $\{\boldsymbol{\Theta}_t^r, \boldsymbol{\Lambda}_t^r\}$ from this emulating prior, where the superscript r indexes Monte Carlo samples for prediction, with $r = 1{:}R$ for some (large) sample size R. Each sampled value then defines Monte Carlo values of one-step forecast moments $\mathbf{A}_t^r \boldsymbol{\mu}_t^r, \boldsymbol{\Sigma}_t^r$ in (3) and (4). Predictions more than one-step ahead follow similarly. Conditional on sampled moments, the resulting conditionally normal predictive distributions can be summarized or simulated for predictive inferences.

C. Naive Time t Posterior Updates Standard updating equations applied independently and in parallel lead to

$$(\boldsymbol{\theta}_{jt}, \lambda_{jt} | \mathcal{D}_t) \sim NG(\widetilde{\mathbf{m}}_{jt}, \widetilde{\mathbf{C}}_{jt}, \widetilde{n}_{jt}, \widetilde{s}_{jt}) \tag{9}$$

with density functions denoted by $\widetilde{p}_{jt}(\cdot|\cdot)$. See Appendix A for details and the explicit updating formulæ. The resulting naive posterior approximation to $p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t | \mathcal{D}_t)$ is then

$$\widetilde{p}(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t | \mathcal{D}_t) = \prod_{j=1:m} \widetilde{p}_{jt}(\boldsymbol{\theta}_{jt}, \lambda_{jt} | \mathcal{D}_t), \tag{10}$$

which ignores the determinant term in (6).

D. Recoupling to Exact Time t Posterior We know that the exact posterior is

$$p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t | \mathcal{D}_t) \propto |\mathbf{I} - \boldsymbol{\Gamma}_t| \prod_{j=1:m} \widetilde{p}_{jt}(\boldsymbol{\theta}_{jt}, \lambda_{jt} | \mathcal{D}_t). \tag{11}$$

Now, draw N independent samples from the m-variate naive posterior approximation in (10); the product form of the density can be exploited for independent and parallelized sampling from the m within-series posteriors in (9); these are then combined to define the full sample $\{\boldsymbol{\Theta}_t^i, \boldsymbol{\Lambda}_t^i\}$. Compute and normalize sample weights $\alpha_{ti} \propto |\mathbf{I} - \boldsymbol{\Gamma}_t^i|$. Use these samples and weights

$$\{\boldsymbol{\Theta}_t^i,\ \boldsymbol{\Lambda}_t^i,\ \alpha_{ti}\}, \quad i = 1{:}N, \tag{12}$$

as an importance sample to compute Monte Carlo estimates of features of the exact joint posterior distribution in (11). Denote this importance sampling, Monte Carlo-based approximation by $p_{\mathrm{MC}}(\cdot|\cdot)$.

Note that the importance sample weights do not depend on the values of the simulated $\boldsymbol{\phi}_{jt}^i, \lambda_{jt}^i$; they depend only on the simultaneous coefficients $\boldsymbol{\gamma}_{jt}$. This reflects the view that the parallel conjugate models will effectively emulate the full multivariate model in inferring series-specific states and volatilities, while some "corrections" will be needed for inferences on the parental states that explicitly define cross-series structure. Importance sampling is the natural and elegant approach to making these corrections. The univariate series/models are *decoupled* for updates and direct simulation, and then *recoupled* for importance sampling targeting the exact posterior.

E. Decoupling of Time t Posterior To move ahead to the next time point, *decouple* the posterior from part D into a product of conjugate forms across series $j = 1{:}m$. A standard variational Bayes (VB) approach, or mean-field approximation (Jaakkola and Jordan, 2000) emulates the exact posterior by an independent product of normal/gammas

$$p(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{D}_t) \propto \prod_{j=1:m} p_{jt}(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{D}_t) \tag{13}$$

with components

$$(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{D}_t) \sim NG(\mathbf{m}_{jt}, \mathbf{C}_{jt}, n_{jt}, s_{jt}). \tag{14}$$

The parameters of this variational Bayes posterior are chosen to minimize the Kullback–Leibler divergence[1] $KL_{p|p_{\mathrm{MC}}}$ of $p(\cdot|\cdot)$ from $p_{\mathrm{MC}}(\cdot|\cdot)$. Using $E[\cdot]$ to denote expectations under $p_{\mathrm{MC}}(\cdot|\cdot)$, standard theory (e.g., West and Harrison, 1997, Section 12.3) implies that:

- $\mathbf{m}_{jt} = E[\lambda_{jt}\boldsymbol{\theta}_{jt}]/E[\lambda_{jt}]$,
- $\mathbf{V}_{jt} = E[\lambda_{jt}(\boldsymbol{\theta}_{jt} - \mathbf{m}_{jt})(\boldsymbol{\theta}_{jt} - \mathbf{m}_{jt})']$,
- $d_{jt} = E[\lambda_{jt}(\boldsymbol{\theta}_{jt} - \mathbf{m}_{jt})'\mathbf{V}_{jt}^{-1}(\boldsymbol{\theta}_{jt} - \mathbf{m}_{jt})]$,

[1]For any random quantity $\mathbf{z}$, the KL divergence of a distribution with density $g(\mathbf{z})$ from one with density $p(\mathbf{z})$ is $KL_{g|p} = -E_p[\log\{p(\mathbf{z})/g(\mathbf{z})\}]$. Here densities are continuous, discrete or mixed and have common support, and $E_p[\cdot]$ is the expectation under $p(\cdot)$.

- n_{jt} is the unique value that satisfies

$$\log(n_{jt} + p_j - d_{jt}) - \psi(n_{jt}/2) - (p_j - d_{jt})/n_{jt} - \log(2E[\lambda_{jt}]) + E[\log \lambda_{jt}] = 0,$$

- $s_{jt} = (n_{jt} + p_j - d_{jt})/(n_{jt}E[\lambda_{jt}])$, and
- $\mathbf{C}_{jt} = s_{jt}\mathbf{V}_{jt}$.

These are easily computed, with n_{jt} requiring a (trivial) iterative numerical approach; see Appendix B.4. Both the conceptual basis and technical aspects of mapping to sets of conjugate forms has a long history in the Bayesian dynamic modeling and forecasting literature (e.g., Harrison and Stevens, 1971; Alspach and Sorenson, 1972; Harrison and Stevens, 1976; Smith and West, 1983; West and Harrison, 1997, Section 12.3.4).

F. Evolution to Time $t+1$ Moving ahead one time point, the states evolve via independent models of (2). This results in evolved priors as given in (7) in part A above, but with time index t updated to $t+1$. Formulæ are given in Appendix A, simply following standard DLM theory and notation (West and Harrison, 1997).

3.3 KL Divergence and IS-VB Strategy

The VB-based posterior of (13) represents an improvement over the initial $\widetilde{p}(\cdot|\cdot)$ of (10) as it minimizes $KL_{g|p_{\text{MC}}}$ over all $g(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t)$ that are products of m normal/gamma forms for the $(\boldsymbol{\theta}_{jt}, \lambda_{jt})$, $j = 1{:}m$. It turns out that the importance sampling weights α_{ti} in (12) provide a direct assessment of the value of the divergence $KL_{\widetilde{p}|p_{\text{MC}}}$. This general but, apparently, not well-known result relating KL divergences to importance sampling weights is of utility here as well as more broadly. Specifically, write $H_N = \sum_{i=1:N} \alpha_{ti} \log(N\alpha_{ti})$, the entropy of the importance sampling weights α_{ti} relative to a set of N uniform weights – one measure of efficacy of importance samplers (e.g., West, 1993). It is easily shown that, as $N \to \infty$, $H_N \to KL_{\widetilde{p}|p_{\text{MC}}}$; hence, the relative entropy gives a direct estimate of the optimized KL divergence. It can also be shown that $H_N \leq N \sum_{i=1:N} \alpha_{ti}^2 - 1 = N/S_N - 1$ where S_N is effective sample size; for large N, the limiting value of S_N/N is bounded above by $1/(1 + KL_{\widetilde{p}|p_{\text{MC}}})$.

Hence H_N gives an estimate of the upper bound of the minimized divergence; if H_N is already small – based on calibrating to effective sample size as above – then we are assured of closeness of the revised VB-based posterior approximation. Furthermore, the KL divergence of any subset of parameters $(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t)$ cannot exceed the divergence on the full set. This makes the latter an operational upper bound on divergences of the approximating marginal posteriors in any of the m individual models. If the overall approximation is good, we do not have to monitor the margins on models $j = 1{:}m$.

A further positive theoretical feature relates to evolution from time t to $t+1$. The KL-optimized product of normal/gamma posteriors for $(\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t|\mathcal{D}_t)$ evolves to a similar analytic form for the time $t+1$ prior $p(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Lambda}_{t+1}|\mathcal{I}_{t+1}, \mathcal{D}_t)$. Now, we know that KL divergence decreases through convolutions; hence, the divergence of this normal/gamma

product for $(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Lambda}_{t+1})$ is a better approximation of the exact time $t+1$ prior than was the case for the time t posterior. If we have a high-quality posterior approximation at time t, then the situation only improves following evolution.

4 GPU-Accelerated Implementation

4.1 General Comments

The analysis strategy of Section 3 is ideally suited to distributed implementation on computers with graphics processing units (GPUs). GPUs feature hundreds or thousands of compute cores that can be used to execute single instruction, multiple data (SIMD) operations in a massively parallel mode (currently common multi-core desktop processors typically have no more than eight cores). Whenever the same set of numerical operations has to be performed many times on different data, GPU-accelerated implementations offer the potential to vastly outperform CPU equivalents, based on distributing these computations to cores in parallel. Our model and computational development, and accompanying code, contribute to the growing body of literature linked to Bayesian statistical computations that are inherently enabled via GPU implementations (e.g., Suchard et al., 2010; Lee et al., 2010) due to being simply ideally suited to the GPU hardware/software model; these references also discuss the relative speed-up that can be achieved over CPU computation. We note some specifics related to each of the steps in the analysis of Section 3.

4.2 Predictive Computations

Computations for predictive distributions (Section 3-B) are immediately parallelizable, exploiting the decoupling/recoupling strategy. For one-step ahead predictions, Monte Carlo draws $\{\boldsymbol{\theta}_t^r, \lambda_t^r\}$, $r = 1{:}R$, from the independent priors at time t in (7) are simulated in parallel. The sampled states and volatilities are then sent to the CPU to combine and compute the implied Monte Carlo values of one-step forecast moments $\mathbf{A}_t^r \boldsymbol{\mu}_t^r, \boldsymbol{\Sigma}_t^r$ in (3) and (4). It is then trivial to numerically summarize and/or simulate the one-step ahead predictive distribution $p(\mathbf{y}_t|\mathcal{I}_t, \mathcal{D}_{t-1})$ from this Monte Carlo sample of means and variance matrices. Predictions more than one-step ahead follow similarly, and again exploit distributed computation as the series-specific states and volatilities evolve independently in state (4).

4.3 Posterior Update Computations

Updates and simulations of naive posterior approximations in Section 3-C,D are each immediately parallelizable. The analytic update computations of parameters of the naive posterior approximations are trivially computed in parallel, and then fed into parallel cores for simulation of the m posteriors in parallel; this is distributable over both m and the Monte Carlo sample size N to maximally utilize the capacity of available GPU cores. Following simulation, the Monte Carlo samples are recoupled, i.e., returned to the CPU for evaluation and normalization of the importance sampling weights α_{ti}.

Based on the multivariate importance samples $\{\boldsymbol{\Theta}_t^i, \boldsymbol{\Lambda}_t^i, \alpha_{ti}\}$, we can then trivially compute summaries of the weights to monitor Monte Carlo accuracy, including the effective sample size and entropy measures S_N, H_N discussed in Section 3.3.

4.4 Computational Costs

Let M represent Monte Carlo sample size, whether $M = R$ for forward sampling for prediction, or $M = N$ in importance sampling for posterior updates (often, we will simply take $R = N$ so that M is the common value). Scaling computations in M is then a critical interest.

The lead complexity of sampling $\{\boldsymbol{\Theta}_t, \boldsymbol{\Lambda}_t\}$ is $\mathcal{O}\big(Mmp_{\max}^2\big)$ where $p_{\max} = \max_{j=1:m} p_j$. Computing the determinant $|\mathbf{I} - \boldsymbol{\Gamma}_*|$ is the most expensive operation, with a cost of $\mathcal{O}\big(Mm^3\big)$. In k-step forecasting for $k \geq 1$, simulation of states $\{\boldsymbol{\Theta}_{t+k}, \boldsymbol{\Lambda}_{t+k}\}$ uses previously sampled states $\{\boldsymbol{\Theta}_{t+k-1}, \boldsymbol{\Lambda}_{t+k-1}\}$ and incurs additional costs of $\mathcal{O}\big(Mmp_{\max}^3\big)$ for each k. Then inverting $\mathbf{I} - \boldsymbol{\Gamma}_{t+k}$ costs $\mathcal{O}\big(Mm^3\big)$. So the overall computational cost of forecasting at each of k-steps ahead is $\mathcal{O}\big(kM(m^3 + mp_{\max}^2)\big)$ as $M \to \infty$. The order of computational costs in M, m is the same for posterior decoupling/recoupling updates and for forecasting, while the actually incurred costs will increase linearly with the numbers of steps k that we choose to forecast.

5 Evaluation: Stock Return Study

5.1 Data and Study Set-Up

We analyze daily log-returns of $m = 400$ S&P stocks. In a simple class of SGDLMs, we evaluate 1-step ahead forecasts for both selected individual series and across all series. We study of the effects of ignoring the coupling of the set of simultaneous model equations, to bear out the utility of the decoupling/recoupling strategy. We also compare the results with those from a benchmark analysis using the standard Wishart discount model of multivariate stochastic volatility (WDLM; West and Harrison, 1997, Chapter 16.4).

Our data represent $m = 400$ current members of the S&P 500 index, restricting to those 400 that were continuously listed from October 2000 to October 2013, our study period. The data are daily log-returns, which are differences in daily log-prices. We use the first 845 daily observations, $T_1 = 1{:}845$ (up to December 2003) as training data for an initial exploratory analysis to define the simultaneous parental sets for each of the $m = 400$ series. Based on these chosen parental sets, we then use the following 522 daily observations, $T_2 = 846{:}1{,}367$ (from January 2004 through December 2005) as further training data to evaluate and select suitable discount factors for the dynamic models, and to provide priors for analysis of the following test data. The test data are the remaining 2,044 observations, $T_{\text{test}} = 1{,}368{:}3{,}411$ (from January 2006 through October 2013). This provides an honest sequential forecasting analysis and evaluation on this substantial series of hold-out/test data. In both training and test data analyses,

the Monte Carlo sample sizes R, for forecasting, and N, for IS-based posterior updates, were set at $R = N = 10{,}000$.

5.2 Model Structure and Specification via Training Data Analysis

SGDLM Form For each of the 400 univariate series, we use the local-level (random walk) for a time-varying trend DLM (West and Harrison, 1997, Chapter 2) common in studies of financial returns (e.g., Aguilar and West, 2000; Nakajima and West, 2013b). This is coupled with a specified set of 10 simultaneous parents, whose coefficients also evolve via random walks. In (1), we then have $p_{j\phi} = 1$, $\mathbf{x}_{jt} = 1$ and $p_{j\gamma} = 10$. The state evolution models of (2) have $\mathbf{G}_{jt} = \mathbf{I}$, the 11-dimensional identity matrix. We use the standard block discounting approach to specify the variance matrices $\mathbf{W}_{jt}$ (West and Harrison, 1997, Section 6.3.2). Specifically, each $\mathbf{W}_{jt}$ is defined using two discount factors $\delta_\phi, \delta_\gamma$, each in $(0, 1)$, that determine evolution variance matrix block components for level and simultaneous parental coefficients, respectively.

Simultaneous Parental Sets We used the initial 3 years of data, $T_1 = 1{:}845$ over 2003, for exploratory analysis to select the $sp(j)$ for each $j = 1{:}400$. On this initial training data set, we simply ran separate, univariate DLMs with a local level and with *all* the remaining 399 series as simultaneous parents. We then chose $p_j = 10$ series to define each $sp(j)$ by selecting those with the largest estimated effect sizes over the later part of the training data.

The set of simultaneous parents obviously plays a key role in model fit and forecasting. Here, we emphasize that more formal model selection is not a theme in the current paper. In practice, we operate with a chosen set of parental sets over given periods of time, refreshing/modifying the parental sets periodically via off-line analysis, and/or using multiple such sets – a restricted number – and engaging in model averaging. A future paper will address this question of model uncertainty. The current paper takes the parental sets as given – based on detailed exploratory analysis of training data – the focus and contributions involving the SGDLM modeling innovation and the decoupling/recoupling computational strategy. We show in this example that the enhancements under this strategy are not merely theoretical considerations, but significantly improve forecasting in this $m = 400$-dimensional setting.

WDLM Form The benchmark model for comparison is a local-level DLM with Wishart discount-based multivariate volatility (West and Harrison, 1997, Chapter 16.4) for the full 400-dimensional time series. This uses a discount factor δ to define the evolution variances of the local levels, and a discount factor β for the Wishart discount-based evolution of the full 400×400 observation variance matrix. Initial priors are taken as $\mathbf{m}_0 = 0$, $\mathbf{C}_0 = 0.001$, $\mathbf{S}_0 = 0.1\mathbf{I}$ and $n_0 = 5$ in the notation of the above reference, and the resulting forward filtering and forecasting equations are as detailed in Theorem 16.4 of West and Harrison (1997).

Discount Factor Specification We ran the SGDLM analysis on the second training set of data, $T_2 = 846{:}1{,}367$. This initialized at $t = 846$ with priors of (8) is based on:

$r_{j,846} = 5$ and $s_{j,845} = 0.001$, $\mathbf{a}_{j,846} = \mathbf{0}$ and $\mathbf{R}_{j,846} = \text{diag}(0.0001, 0.01, \ldots, 0.01)$. This analysis was used to explore the impact of varying the state discount factors $\delta_\phi, \delta_\gamma$ and the volatility discount $\beta_j = \beta$, all assumed the same across series j. This settled on chosen values $\beta_j = 0.98$, $\delta_\phi = 0.98, \delta_\gamma = 0.99$ based on standard evaluation of one-step ahead forecasting accuracy.

We then reran the analysis but without the IS-VB step of the computations, i.e., simply using the parallel, independently updated set of models. This analysis led to choices $\beta_j = 0.98$ and $\delta_\phi = 0.98$ again, but we found improved 1-step ahead forecasting with an appropriately higher discount factor $\delta_\gamma = 0.999$ on the parental predictors. Below we evaluate forecasting on the hold-out/test data using this incoherent/approximate set of independent model as well using the full decoupled/recoupled analysis. This differential choice of discount factors makes this an honest comparison as we begin the analysis of test data with optimized parameter specification for each strategy.

A parallel analysis of the benchmark WDLM was similarly evaluated, and led to optimal discount factors of $\delta = 0.98$ and $\beta = 0.9975$.

5.3 Forecasts of Stock Returns: Test Data Analysis

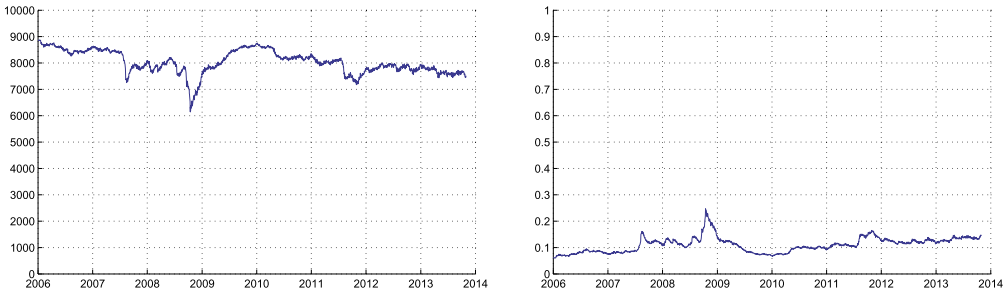
We review sequential learning and forecasting in the test data period, $T_{\text{test}} = 1{,}368{:}3{,}411$, with a number of summaries. This includes aspects of Monte Carlo accuracy, and forecast assessments for individual time series as well as in the aggregate. Part of this includes analyzing coverage rates of 1-step ahead forecast intervals and validation against observed values. The initial priors at time $t = 1{,}368$ are, in each analysis, simply those evolved from the corresponding posteriors at time $t - 1$, the last day of the training data period.

Analysis of the IS-VB Strategy

Figure 1 shows the effective sample size S_N of the IS-VB step at each time point t. Based on the $N = 10{,}000$ samples, S_N exceeds 7,000 during approximately 98% of the test time period T_{test}. There is a short-lived drop to about 6,100 during a period of extreme market stress following the collapse of Bear Sterns in September 2008. In the range from 7,000 to 9,000, the typical effective sample size indicates excellent performance of the importance sampler. Figure 1 also shows the corresponding IS-based estimate H_N of the KL divergence $KL_{\widetilde{p}|p_{\text{MC}}}$ of $\widetilde{p}(\cdot)$ from the Monte Carlo approximation $p_{\text{MC}}(\cdot)$ at each time. Bounding the KL of the variational Bayes posterior and the exact posterior from above implies that the variational Bayes posterior will be very close to the exact posterior.

Aggregate Analysis

Table 1 shows the average coverage rates of forecasts across all series and the entire test data time period $T_{\text{test}} = 1{,}368{:}3{,}411$. Forecast intervals from the benchmark WDLM are broadly similar to those from the SGDLM without IS-VB decoupling/recoupling



Effective sample size S_N KL divergence $KL_{\widetilde{p}|p}$

Figure 1: S_N and $KL_{\widetilde{p}|p}$ during the test data time period T_{test}.

(steps D and E of Section 3; indicated as "no IS-VB" in the table). The full SGDLM analysis using the importance sampling/variational Bayes strategy (indicated simply as "IS-VB" in the table) is more accurate – across all coverage levels – than the analysis that does not use the IS-VB strategy.

Forecast interval	99.0%	95.0%	90.0%	80.0%	50.0%	20.0%	10.0%
IS-VB	98.4%	95.6%	92.4%	85.5%	59.7%	27.2%	14.4%
no IS-VB	99.1%	97.6%	96.0%	92.4%	76.3%	41.8%	23.5%
WDLM	99.2%	98.2%	97.0%	94.4%	79.7%	43.4%	24.4%

Table 1: Coverage of centered forecast intervals across all series $j = 1{:}m$ and the entire test data time period T_{test}.

Close-Up Analysis of Individual Stocks

We now focus on series-specific forecasting performance for stock returns of six well-known companies: Apple, Bank of America, General Electric, McDonald's, Pfizer and Starbucks.

Trend Figure 2 shows the 60-day tracking moving average of the daily log-returns, comparing empirical trends with those forecast under the SGDLM with IS-VB, the SGDLM without IS-VB, and under the WDLM. The three models perform very similarly. A more complex model with series-specific predictors $\mathbf{x}_{jt}$ would lead to noticeable performance improvement of the SGDLMs relative to the WDLM, as the latter requires the same predictors across all series $j = 1{:}m$. We see no obvious advantage of using the posterior decoupling/recoupling in this aspect of the analysis. This is to be expected: the importance sampling weights are defined by $|\mathbf{I} - \mathbf{\Gamma}_t|$, and these values are not affected by values of the trend parameters; as a result, the marginal posteriors will be adequately estimated without the IS-VB steps.

Volatility We represent volatility via standard deviations of returns. Figure 2 overlays the observed 60-day tracking moving average of volatility for the six selected stocks,

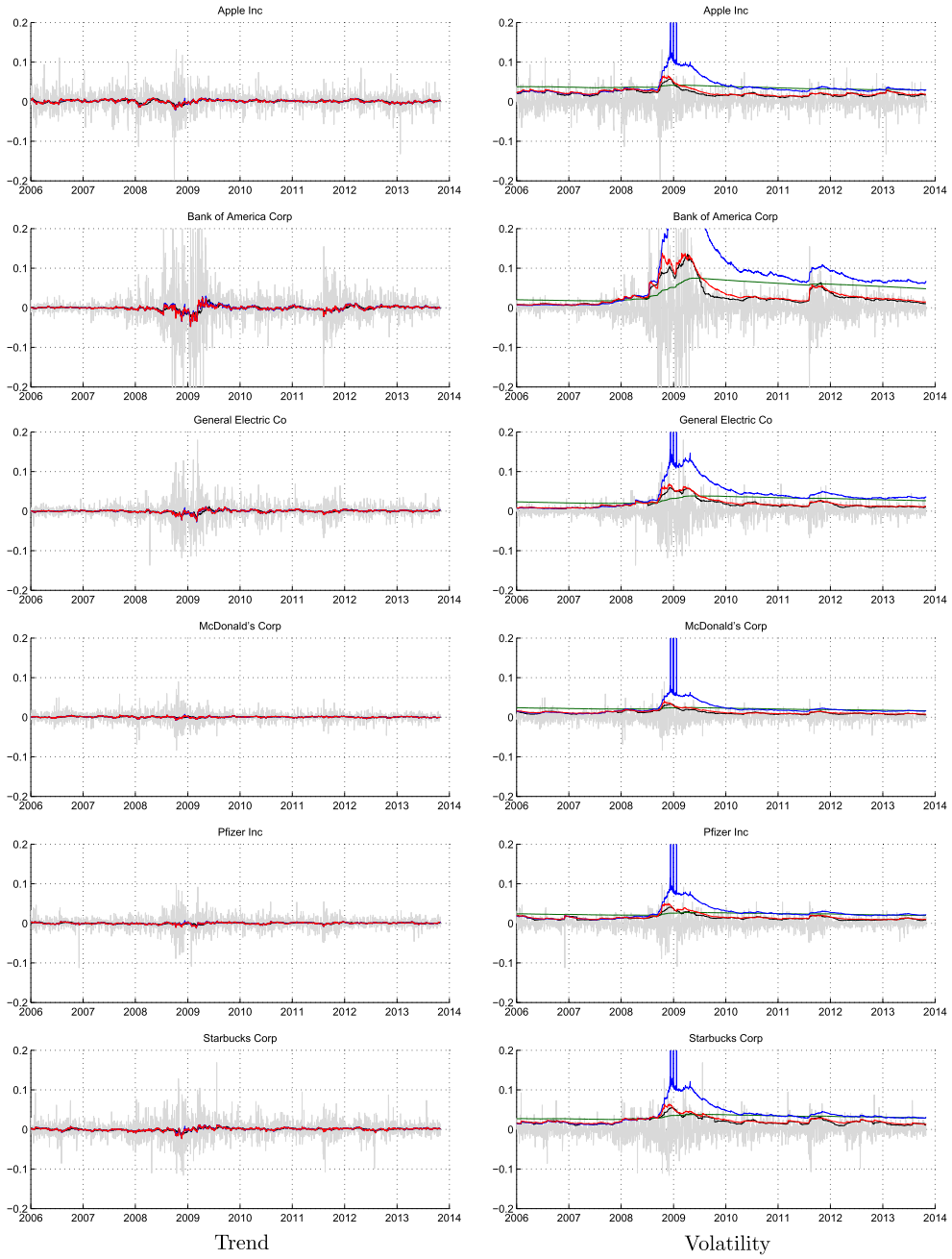


Figure 2: Comparison of the realized 60 day moving average/volatility and forecast returns/volatilities. The observed returns are in gray, the observed trend is in black, the results from the WDLM analysis are in green, these from the full SGDLM analysis are in red, and those from analysis without VB are in blue.

with volatilities as estimated from our analyses. The volatility forecasts from SGDLM analysis with and without IS-VB track each other closely until the onset of the financial crisis in the fall of 2008. Thereafter, without IS-VB – i.e., ignoring the cross-series constraints – the volatility forecasts skyrocket and never return to historically normal levels. In stark contrast to the positive, the full SGDLM analysis generates forecast volatilities in extremely good agreement with the empirically evaluated levels, before, throughout and after the financial crisis and recessionary years. Volatility forecasts from the 400-dimensional WDLM are comparatively unresponsive to changes over time, and just do not reflect key aspects of the empirically estimated volatilities. The WDLM consistently and significantly over-estimates the empirical volatility both before and after the financial crisis; and, during the period of heightened market stress, the volatility forecasts do not adequately reflect higher realized volatilities.

Coverage of Forecast Intervals Coverage rates of forecast intervals for the six selected companies are reported in Table 2. The results confirm at these individual levels the findings from the aggregate analysis. While forecast intervals up to 95% tend to

Forecast interval	99.0%	95.0%	90.0%	80.0%	50.0%	20.0%	10.0%
				Apple Inc			
IS-VB	98.4%	95.3%	92.4%	86.0%	59.4%	26.0%	13.6%
no IS-VB	99.0%	97.9%	95.8%	92.4%	74.3%	40.3%	20.4%
WDLM	99.6%	99.1%	98.1%	95.6%	79.4%	43.7%	23.1%
				Bank of America Corp			
IS-VB	98.2%	95.1%	91.9%	85.9%	60.9%	27.2%	14.6%
no IS-VB	98.2%	96.1%	94.5%	91.3%	79.1%	51.3%	30.9%
WDLM	97.8%	96.1%	94.6%	91.0%	80.0%	47.0%	26.3%
				General Electric Co			
IS-VB	98.2%	94.7%	91.1%	83.9%	58.6%	25.8%	12.9%
no IS-VB	98.9%	97.5%	95.2%	92.2%	77.4%	44.9%	25.6%
WDLM	98.7%	98.0%	96.8%	94.0%	80.9%	45.5%	24.9%
				McDonald's Corp			
IS-VB	98.5%	96.1%	92.8%	86.4%	59.3%	26.5%	13.3%
no IS-VB	99.1%	98.4%	96.6%	92.9%	73.7%	38.6%	20.3%
WDLM	99.7%	99.2%	98.5%	97.1%	82.3%	44.2%	24.7%
				Pfizer Inc			
IS-VB	98.5%	95.5%	92.3%	85.4%	60.5%	27.0%	14.1%
no IS-VB	99.6%	98.0%	96.6%	93.0%	77.5%	40.7%	21.0%
WDLM	99.5%	98.8%	98.1%	95.4%	80.6%	43.5%	23.9%
				Starbucks Corp			
IS-VB	98.2%	95.5%	92.7%	86.0%	60.3%	27.4%	13.9%
no IS-VB	98.8%	97.0%	95.7%	92.0%	74.2%	39.7%	21.8%
WDLM	99.2%	98.0%	96.5%	94.0%	76.9%	40.4%	22.3%

Table 2: Coverage of centered forecast intervals of individual stock returns averaged over the test data time period $T_{\text{test}} = 1{,}368{:}3{,}411$.

be over-estimated by either SGDLM, the IS-VB decoupling/recoupling analysis generally improves forecasting performance, while also yielding more precise forecast intervals. Averaged across the test data period T_{test}, the forecast intervals from the WDLM analysis are similar to those from the analysis of the SGDLM without IS-VB decoupling/recoupling, again reflecting the aggregate results.

5.4 Realized Computation Time

Section 4.4 noted theoretical considerations concerning computational loads. That is complemented here by an empirical assessment based on rerunning multiple analyses using varying numbers of series m and sizes of the parental sets. Table 3 summarizes the empirical run times across several values, in each case generating $N = M = 10{,}000$ posterior and 1-step forecast samples at each time point. The scaling of the run times is roughly in line with our theoretical estimates, considering some fixed time for memory transfers and communication overheads for coordinating four GPUs. The times were measured on a 2012 computer with four NVidia Tesla C2050 GPUs with 448 CUDA cores each.

	$p_j = 5$		$p_j = 10$		$p_j = 20$	
	Posterior	Forecast	Posterior	Forecast	Posterior	Forecast
$m = 50$	0.07	0.07	0.11	0.08	0.20	0.10
$m = 100$	0.16	0.25	0.21	0.26	0.43	0.32
$m = 200$	0.50	1.17	0.61	1.20	1.04	1.31
$m = 400$	2.13	6.75	2.32	6.78	3.22	7.01

Table 3: Realized run times (seconds) of SGDLM analysis and forecasting with $R = N = 10{,}000$.

6 Additional Comments

The multivariate dynamic model formulation via SGDLMs is persuasive in terms of the theoretical ability to flexibly represent – in state-space forms – the dynamics of individual series coupled with contemporaneous, cross-series multivariate dependencies. The framework conceptually allows for scalability to higher dimensional series and completely frees the modeler from conceptual and technical constraints encountered in existing models. With that outlook, while also recognizing the inherent opportunities for model decoupling as part of an overall analysis and with the insight that individual, univariate model analyses can often come close to representing core aspects of the full multivariate model, we defined the decoupling/recoupling strategy that builds on importance sampling and variational Bayes to define computationally efficient and practically effective analyses. The computational efficacy arises from massively distributed computation, for which GPU hardware is ideally suited; we have presented the ideas and key details of GPU-enhanced computations, and provide freely available software for interested researchers to follow-up. The practical effectiveness is demonstrated in the 400-dimensional time series study, and underpinned by the discussion of theoretical

questions of importance sampling and variational Bayes/KL divergence-based approximation of unknowable exact posteriors.

Our analyses show that the SGDLM – with importance sampling and variational Bayes or without – can significantly improve model adequacy and forecast accuracy relative to the standard matrix-normal WDLM. Then, we also show substantial improvements in forecasting performance using SGDLMs with the overlay of the IS-VB strategy. Importantly, this latter benefit is realized at reasonable computational costs: the run-time is only increased by about a third in this case study. On a rather standard 2012 desktop (with four NVidia Tesla C2050 GPUs having 448 CUDA cores each), posterior updating and forecasting with $N = M = 10{,}000$ Monte Carlo samples at each time step for our 400-dimensional time series complete in less than 10 seconds per step. This easily allows for real-time analysis in intervals less than one minute and will scale to several hundreds and low thousands of series on current and emerging commodity desktop/laptop machines.

Current and future work includes the refinement of our existing Matlab interface, and development of an R interface to our GPU-accelerated implementation of Bayesian learning and forecasting of SGDLMs. Current applied and methodological questions under study are questions of parental set selection – which again must involve a focus on practicalities and move away from a purely theoretical but practically unworkable "global model averaging" perspective. Here again forecasting performance coupled with innovations in computational strategies will likely be key to practical progress. A further potential direction is to consider the development of methods of sequential Monte Carlo (SMC), based on particle filtering and learning concepts, as mooted by a referee. This seems a propitious direction for methodology development, especially in view of the potential to alleviate some of the inherent degeneracy issues faced by SMC methods through the key and central decoupling/recoupling approach we have introduced.

Appendix A: DLM Update and Evolution Equations

We give summary details of the equations defined by evolution and updating steps in the set of m univariate DLMs.

Evolution Equations

Standing at time $t-1$, we have series$-j$ specific normal/gamma posteriors

$$(\boldsymbol{\theta}_{j,t-1}, \lambda_{j,t-1}|\mathcal{D}_{t-1}) \sim NG(\mathbf{m}_{j,t-1}, \mathbf{C}_{j,t-1}, n_{j,t-1}, s_{j,t-1})$$

as in (13). Evolving to time t, the state vector $\boldsymbol{\theta}_{j,t-1}$ undergoes a linear state evolution and the precision $\lambda_{j,t-1}$ undergoes a coupled gamma discount evolution based on a specified discount factor β_j. The implied prior for the next time point is then

$$(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{I}_t, \mathcal{D}_{t-1}) \sim NG(\mathbf{a}_{jt}, \mathbf{R}_{jt}, r_{jt}, c_{jt}) \tag{15}$$

as in (8), with parameters given by:

$$\mathbf{a}_{jt} = \mathbf{G}_{jt}\mathbf{m}_{j,t-1},\ \mathbf{R}_{jt} = \mathbf{G}_{jt}\mathbf{C}_{j,t-1}\mathbf{G}'_{jt} + \mathbf{W}_{jt},\ c_{jt} = s_{j,t-1} \text{ and } r_{jt} = \beta_j n_{j,t-1}.$$

The models in our application specify evolution variance matrices using one single discount factor δ_j for each series. In this case, the evolution variance matrices are set as $\mathbf{W}_{jt} = \mathbf{G}_{jt}\mathbf{C}_{j,t-1}\mathbf{G}'_{jt}(1/\delta_j - 1)$ resulting in $\mathbf{R}_{jt} = \mathbf{G}_{jt}\mathbf{C}_{j,t-1}\mathbf{G}'_{jt}/\delta_j$. Here the $\delta_j \in (0, 1]$ and, typically, take larger values on this range (West and Harrison, 1997, Chapter 6).

Updating Equations

With the normal/gamma prior of (15) above, the implied normal/gamma posterior is that in (9). Specifically,

$$(\boldsymbol{\theta}_{jt}, \lambda_{jt}|\mathcal{D}_t) \sim NG(\widetilde{\mathbf{m}}_{jt}, \widetilde{\mathbf{C}}_{jt}, \widetilde{n}_{jt}, \widetilde{s}_{jt})$$

with defining parameters computed using standard updating equations (Prado and West, 2010, Section 14.3), as follows:

First, compute the following:

1-step ahead forecast error,	$e_{jt} = y_{jt} - \mathbf{F}'_{jt}\mathbf{a}_{jt}$,
1-step ahead forecast variance factor,	$q_{jt} = c_{jt} + \mathbf{F}'_{jt}\mathbf{R}_{jt}\mathbf{F}_{jt}$,
Adaptive coefficient vector,	$\mathbf{A}_{jt} = \mathbf{R}_{jt}\mathbf{F}_{jt}/q_{jt}$,
Volatility update factor,	$z_{jt} = (r_{jt} + e^2_{jt}/q_{jt})/(r_{jt} + 1)$.

Then, compute the posterior parameters:

Posterior mean vector,	$\widetilde{\mathbf{m}}_{jt} = \mathbf{a}_{jt} + \mathbf{A}_{jt}e_{jt}$,
Posterior covariance matrix factor,	$\widetilde{\mathbf{C}}_{jt} = (\mathbf{R}_{jt} - \mathbf{A}_{jt}\mathbf{A}'_{jt}q_{jt})z_{jt}$,
Posterior degrees of freedom,	$\widetilde{n}_{jt} = r_{jt} + 1$,
Posterior residual variance estimate,	$\widetilde{s}_{jt} = z_{jt}c_{jt}$.

Appendix B: C++/CUDA Implementation Details

We provide an overview of the GPU implementation of the complete filtering/forecasting analysis, noting a number of technical aspects and requirements as well as giving some flavor of the structure of C++/CUDA programming.[2]

B.1 Key Features of CUDA Implementation

Wherever possible, our implementation uses batched functions from the CUBLAS library to perform the linear algebra operations for forward filtering and forecasting. Batched functions are a recent innovation introduced into the CUBLAS library. They accelerate operations on a large number of small matrices by bundling them into a sin-

[2]Our implementation and code is based on a computer with an NVidia CUDA-enabled GPU with a compute capability of at least 2.0. Interested users may run the code, so long as CUDA runtime, CUBLAS and CURAND libraries of Version 5.5 or newer are installed. We also have a user-friendly Matlab interface compatible with Matlab versions as early as R2010a; though the C++/CUDA is free-standing, some researchers may be interested in accessing GPU facilities via Matlab.

gle function call. This reduces the overhead of initiating many small operations on the GPU individually. In particular, we use:

- `cublasDgemmBatched(...)` for batched matrix-matrix multiplications,
- `cublasDgetrfBatched(...)` for batched LU factorization,
- `cublasDgetriBatched(...)` for batched matrix inversion.

We developed customized kernels to achieve maximum performance for operations that fall outside the scope of current batched CUBLAS functions; see below for details.

B.2 Multiple Device Parallelization

This presents the key points regarding GPU parallelization using multiple GPU devices. On top of massively parallel execution of SIMD operations in kernels, CUDA streams can be used to organize sequences of operations. The operations within each stream are executed sequentially, but different streams are executed concurrently on the GPU. Furthermore, different streams can be assigned to different GPU devices, allowing multiple GPU computing. Concurrent execution and data transfers allow the most efficient usage of GPU resources.

A call to the function `cudaSetDevice(k)` activates the kth GPU device. This means all future GPU calls will be executed on this device until the current device is changed by another call to `cudaSetDevice(...)`. Our implementation sets up one CUDA stream per GPU. In our importance sampling steps to generate a posterior sample of size N and then forecast simulation of M samples, each of K GPUs will compute N/K importance samples and then N/M direct samples; this is parallelization by particles.

The key functions from the CUDA API that organize asynchronous GPU execution are:

- `cudaStreamCreate(...)` to create a stream,
- `cudaMemcpyAsync(...)` to asynchronously move data between computer and GPU,
- `cudaStreamSynchronize(...)` to wait until a stream's execution is completed,
- `cudaStreamDestroy(...)` to destroy a stream.

Calls to CUDA kernels, most API function calls, and asynchronous memory copy operations return control to the host immediately. This allows us to send commands to different GPUs in a `for` loop and still have them executed in parallel. However, it is necessary to wait until the operations are completed before returning results; this is where the function `cudaStreamSynchronize(...)` comes in. This operates after the GPU operations that are to be executed in parallel are sent to their respective devices.

Below is an outline of a multiple device parallelized program.

Pseudo Code

for GPU device $k = 1 : K$ **do**
  Call `cudaSetDevice(k)` to activate the kth GPU.
  Call `cudaStreamCreate(stream_k)` to create a stream on the kth GPU.
  Call `cudaMemcpyAsync(..., stream_k)` to asynchronously copy input arguments onto device memory.
end for
for GPU device $k = 1 : K$ **do**
  Call `cudaSetDevice(k)` to activate the kth GPU.
  Call the Variational Bayes posterior estimation subroutine to generate N/K importance samples, and then the forecasting simulation subroutine to generate M forecast samples.
end for
for GPU device $k = 1 : K$ **do**
  Call `cudaSetDevice(k)` to activate the kth GPU.
  Call `cudaMemcpyAsync(..., stream_k)` to asynchronously copy results to host memory.
end for
for GPU device $k = 1 : K$ **do**
  Call `cudaSetDevice(k)` to activate the kth GPU.
  Call `cudaStreamSynchronize(stream_k)` to wait until the computations and memory transfers are completed.
  Call `cudaStreamDestroy(stream_k)` to destroy the stream on the kth GPU.
end for
Combine and post-process the results on the CPU, if applicable.

B.3 Generation of Gamma Variates

While the CUDA toolkit provides basic random number generators for the uniform and normal distributions, it is up to the developer to implement other distributions. Our implementation of a gamma random number generator uses a standard rejection sampling algorithm (Marsaglia and Tsang, 2000), requiring one uniform and one normal random number at each step. With expected acceptance probabilities over 90%, running the scheme $2n$ times to generate n Gamma random numbers will almost certainly suffice.

We generate $2n$ uniforms and normals with just one call to `curandGenerateUniformDouble` and `curandGenerate NormalDouble`, respectively. Bundled calls to these CUDA toolkit functions are the most efficient way to generate random numbers on the GPU. Generating enough random numbers for $2n$ proposals leads to highly efficient GPU processing without unnecessary synchronization with the host device.

Specifically, a block of CUDA threads is tasked with generating a batch of $k < n$ gamma random numbers. The batch size is typically chosen as the maximum number of parallel threads the GPU can evaluate, that is, $k = 512$ or $k = 1024$. That batch size is large enough so that there is going to be at least one rejected proposal. This means that at least one thread will have to try a second attempt for acceptance. On the other hand, k is large enough that $2k$ proposals will yield at least k acceptances.

The architecture of a GPU is such that no thread of a block is available when at least one thread is busy. The second attempt of at least one thread makes the entire

block unavailable for other operations. Parallel processing of the threads means that it does not cost additional time to let all threads of the same block generate a second proposal. Furthermore, synchronization within a block is very fast. We will count the number of acceptances and rejections of the first attempt and fill up the k-size vector of gamma random variables in the second attempt.

B.4 Computing n_{jt} in Variational Bayes

In the VB optimization of Section 3.2, the degrees-of-freedom parameter n_{jt} is defined implicitly by

$$\log(n_{jt} + p_j - d_{jt}) - \psi(n_{jt}/2) - (p_j - d_{jt})/n_{jt} - \log(2E[\lambda_{jt}]) + E[\log \lambda_{jt}] = 0. \quad (16)$$

Our C++/CUDA program implements a CUDA kernel for the Newton–Raphson method (Atkinson, 1989) to solve for n_{jt}, initialized at $\widetilde{n}_{jt}$. This uses standard numerical approximations for the digamma and trigamma functions (Abramowitz and Stegun, 1972), which we implemented for this problem as the CUDA toolkit does not provide implementations of these functions. These computations are parallelized over series $j = 1{:}m$.

B.5 Memory Management

C++/CUDA programming requires proper memory management to avoid segmentation faults as well as memory leaks. Memory management on GPUs is very similar to memory management on the host device/CPUs.

Memory is allocated using the `malloc` or `cudaMalloc` commands and freed using `free` or `cudaFree`. This is a conceptually trivial task. However, keeping track of all allocated memory poses a real challenge when more involved software projects branch into different paths. To address this, we developed a universal and lightweight helper class, `memory_manager`, that automates memory management. This class keeps pointers to allocated memory in a C++/boost list and provides methods to allocate, and keep track of, memory as well as a `clear` method to free all memory it tracks.

Each function starts its own instance of our `memory_manager` to manage memory that is not to live beyond the lifetime of that function. At every branch that ends a function, we simply invoke the `clear` method of the respective function's `memory_manager` to free that memory.

Furthermore, there is a global instance of `memory_manager` to manage the large memory blocks needed for simulation of forecasting and variational Bayes approximation. Allocating and freeing that memory at every time step t would be prohibitively time-consuming.

B.6 Matlab Interface

MathWorks provides a C library, `mex.h`, to allow the development of a Matlab interface to C, C++ and CUDA software. This library provides functionality to read Matlab input, write Matlab output and write Matlab messages.

Recent versions of the Matlab Parallel Computing toolbox have another library for Matlab GPU input/output, `mxGPUArray.h`, that also allows the allocation and freeing of GPU memory. However, in contrast to our memory manager, Matlab's memory tracking only avoids memory leaks when Matlab ends, not while the function is active.

We decided against the use of the Matlab GPU library for three reasons. First, up to Matlab R2013b, Matlab ships with an outdated version of the CUDA toolkit library that does not yet provide the latest batch parallelism functionality that we use. Second, by not relying on the Matlab GPU library, our functions can be compiled and run by users that do not license the parallel computing toolbox. Third, we add backwards compatibility to earlier versions of Matlab that do not come with GPU capabilities by programming the GPU interface ourselves. This has been tested with Matlab versions as early as R2010a without any problems. We will of course revisit the questions as new versions of Matlab come along.

References

Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. New York: Dover Publications. 145

Agarwal, D., Chen, D., Lin, L., Shanmugasundaram, J., and Vee, E. (2010). "Forecasting high-dimensional data." In: Elmagarmid, A. K. and Agrawal, D. (eds.), *Proceedings of the ACM SIGMOD International Conference on Management of Data, Indianapolis, USA*, 1003–1012. ACM. 125

Aguilar, O. and West, M. (2000). "Bayesian dynamic factor models and portfolio allocation." *Journal of Business & Economic Statistics*, 18(3): 338–357. 125, 126, 135

Alspach, D. L. and Sorenson, H. W. (1972). "Nonlinear Bayesian estimation using Gaussian sum approximations." *IEEE Transactions on Automatic Control*, 17: 439–448. 132

Anacleto, O., Queen, C., and Albers, C. J. (2013). "Multivariate forecasting of road traffic flows in the presence of heteroscedasticity and measurement errors." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 62: 251–270. MR3045876. doi: http://dx.doi.org/10.1111/j.1467-9876.2012.01059.x. 125

Anselin, L. (1988). *Spatial econometrics: Methods and models*. Dordrecht: Kluwer Academic Publishers. 128

Atkinson, K. (1989). *An Introduction to Numerical Analysis*. New York: Wiley. MR1007135. 145

Carvalho, C. M. and West, M. (2007). "Dynamic matrix-variate graphical models." *Bayesian Analysis*, 2(1): 69–98. MR2289924. doi: http://dx.doi.org/10.1214/07-BA204. 126, 127

Harrison, P. J. and Stevens, C. F. (1971). "A Bayesian approach to short-term forecasting." *Operations Research Quarterly*, 22: 341–362. MR0292225. 132

— (1976). "Bayesian forecasting (with discussion)." *Journal of the Royal Statistical Society: Series B (Methodological)*, 38: 205–247. MR0655429. 132

Hosking, J., Natarajan, R., Ghosh, S., Subramanian, S., and Zhang, X. (2013). "Short-term forecasting of the daily load curve for residential electricity usage in the Smart Grid." *Applied Stochastic Models in Business and Industry*, 29(6): 604–620. MR3151661. doi: http://dx.doi.org/10.1002/asmb.1987. 125

Jaakkola, T. and Jordan, M. I. (2000). "Bayesian parameter estimation via variational methods." *Statistics and Computing*, 10: 25–27. 126, 131

Koop, G. (2012). "Using VARs and TVP-VARs with many macroeconomic variables." *Central European Journal of Economic Modelling and Econometrics*, 4: 143–167. 125

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). "On the utility of graphics cards to perform massively parallel simulation with advanced Monte Carlo methods." *Journal of Computational & Graphical Statistics*, 19(4): 769–789. 126, 133

Lopes, H. F., McCulloch, R. E., and Tsay, R. (2010). "Cholesky stochastic volatility." Technical report, University of Chicago, Booth Business School. 126

Lopes, H. F. and Polson, N. G. (2010). "Bayesian inference for stochastic volatility." In: Böcker, K. (ed.), *Rethinking Risk Measurement and Reporting: Uncertainty, Bayesian Analysis and Expert Judgement*, 515–551. Risk Books/Incisive Financial Publishing Ltd: London. 126

Marsaglia, G. and Tsang, W. W. (2000). "A simple method for generating gamma variables." *ACM Transactions on Mathematical Software*, 26(3): 363–372. MR1809948. doi: http://dx.doi.org/10.1145/358407.358414. 144

Mukherjee, C., Kasibhatla, P. S., and West, M. (2014). "Spatially-varying SAR models and Bayesian inference for high-resolution lattice data." *Annals of the Institute of Statistical Mathematics*, 66: 473–494. MR3211871. doi: http://dx.doi.org/10.1007/s10463-013-0426-9. 128

Nakajima, J. and West, M. (2013a). "Bayesian analysis of latent threshold dynamic models." *Journal of Business & Economic Statistics*, 31: 151–164. MR3055329. doi: http://dx.doi.org/10.1080/07350015.2012.747847. 125, 128

— (2013b). "Bayesian dynamic factor models: Latent threshold approach." *Journal of Financial Econometrics*, 11: 116–153. 126, 135

Oliveira, V. D. and Song, J. J. (2008). "Bayesian analysis of simultaneous autoregressive models." *The Indian Journal of Statistics*, 70-B: 323–350. MR2563993. 128

Palomo, J., Dunson, D. B., and Bollen, K. (2007). "Bayesian structural equation modeling." In: Lee, S. Y. (ed.), *Handbook of Latent Variables and Related Models*, volume 1 of *Handbook of Computing and Statistics with Applications*, 163–188. North Holland. 127

Pitt, M. and Shephard, N. (1999). "Time varying covariances: A factor stochastic volatility approach (with discussion)." In: Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M. (eds.), *Bayesian Statistics VI*, 547–570. Oxford University Press. MR1724873. 126

Prado, R. (2010). "Multi-state models for mental fatigue." In: O'Hagan, A. and West, M. (eds.), *The Handbook of Applied Bayesian Analysis*, 845–874. Oxford University Press. MR2790366. 125

Prado, R. and West, M. (2010). *Time Series: Modeling, Computation & Inference*. Chapman & Hall/CRC Press. MR2655202. 126, 130, 142

Queen, C. M. (1994). "Using the multiregression dynamic model to forecast brand sales in a competitive product market." *Journal of the Royal Statistical Society: Series D (The Statistician)*, 43: 87–98. 125

Quintana, J. M. and West, M. (1987). "An analysis of international exchange rates using multivariate DLMs." *Journal of the Royal Statistical Society: Series D (The Statistician)*, 36: 275–281. 126

Smith, A. F. M. and West, M. (1983). "Monitoring renal transplants: An application of the multi-process Kalman filter." *Biometrics*, 39: 867–878. 132

Suchard, M. A., Wang, Q., Chan, C., Frelinger, J., Cron, A. J., and West, M. (2010). "Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures." *Journal of Computational and Graphical Statistics*, 19(2): 419–438. MR2758309. doi: http://dx.doi.org/10.1198/jcgs.2010.10016. 126, 133

Trejo, L. J., Knuth, K., Prado, R., Rosipal, R., Kubitz, K., Kochavi, R., Matthews, B., and Zhang., Y. (2007). "EEG-based estimation of mental fatigue: Convergent evidence for a three-state model." In: Schmorrow, D. D. and Reeves, L. M. (eds.), *Augmented Cognition, HCII 2007, LNAI 4565*, 201–211. Berlin Heidelberg: Springer-Verlag. 125

Wand, M. P., Ormerod, J. T., Padoan, S. A., and Fuhrwirth, R. (2011). "Mean field variational Bayes for elaborate distributions." *Bayesian Analysis*, 6: 847–900. MR2869967. 126

Wang, H. and West, M. (2009). "Bayesian analysis of matrix normal graphical models." *Biometrika*, 96: 821–834. MR2564493. doi: http://dx.doi.org/10.1093/biomet/asp049. 126

West, M. (1993). "Approximating posterior distributions by mixtures." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 54: 553–568. MR1224405. 132

West, M. and Harrison, P. J. (1989). "Subjective intervention in formal models." *Journal of Forecasting*, 8: 33–53. 128

— (1997). *Bayesian Forecasting & Dynamic Models*. Springer Verlag, 2nd edition. MR1482232. 126, 128, 129, 131, 132, 134, 135, 142

Whittle, P. (1954). "On stationary processes in the plane." *Biometrika*, 41: 434–449. MR0067450. 128

Zhao, Z. Y. and West, M. (2014). "Dynamic compositional regression modelling: Application in financial time series forecasting and portfolio decisions." Technical report, Department of Statistical Science, Duke University. 128

Zhou, X., Nakajima, J., and West, M. (2014). "Bayesian forecasting and portfolio decisions using dynamic dependent sparse factor sparse." *International Journal of Forecasting*, 30: 963–980. 125

Acknowledgments

Lutz F. Gruber: Research developed as a Fulbright Visiting Scholar in the Department of Statistical Science at Duke University, partly supported under the Fulbright Program for Foreign Students.

Mike West: Research partially supported by the US National Science Foundation [grant DMS-1106516].

Any opinions, findings and conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of the Fulbright Foundation or the NSF.

We thank the Editor, an Associate Editor and a referee for constructive comments on our manuscript.