A limited-memory acceleration strategy for MCMC sampling in hierarchical Bayesian calibration of hydrological models

George Kuczera,[1] Dmitri Kavetski,[1] Benjamin Renard,[2] and Mark Thyer[1]

Received 11 December 2009; accepted 19 February 2010; published 17 July 2010.

[1] Hydrological calibration and prediction using conceptual models is affected by forcing/response data uncertainty and structural model error. The Bayesian Total Error Analysis methodology uses a hierarchical representation of individual sources of uncertainty. However, it is shown that standard multiblock "Metropolis-within-Gibbs" Markov chain Monte Carlo (MCMC) samplers commonly used in Bayesian hierarchical inference are exceedingly computationally expensive when applied to hydrologic models, which use recursive numerical solutions of coupled nonlinear differential equations to describe the evolution of catchment states such as soil and groundwater storages. This note develops a "limited-memory" algorithm for accelerating multiblock MCMC sampling from the posterior distributions of such models using low-dimensional jump distributions. The new algorithm exploits the decaying memory of hydrological systems to provide accurate tolerance-based approximations of traditional "full-memory" MCMC methods and is orders of magnitude more efficient than the latter.

Citation: Kuczera, G., D. Kavetski, B. Renard, and M. Thyer (2010), A limited-memory acceleration strategy for MCMC sampling in hierarchical Bayesian calibration of hydrological models, *Water Resour. Res.*, *46*, W07602, doi:10.1029/2009WR008985.

1. Introduction

[2] Characterizing uncertainties in streamflow predicted using conceptual rainfall-runoff (CRR) models is a key research and operational challenge [e.g., *Clark et al.*, 2008; *Vrugt et al.*, 2005]. Bayesian total error analysis (BATEA) explicitly characterizes forcing, response, and structural errors using a hierarchical formulation [*Kavetski et al.*, 2006; *Kuczera et al.*, 2006], which generally results in high-dimensional posterior distributions with hundreds or more latent variables. More generally, very high dimensional hierarchical models have been reported in hydrology and elsewhere [*Cressie et al.*, 2009; *Reichert and Mieleitner*, 2009].

[3] Hierarchical inferences are usually implemented using Gibbs sampling or, more generally, multiblock Markov chain Monte Carlo (MCMC) samplers. However, many statistical applications and software (e.g., BUGS [*Gilks et al.*, 1994]) are not well suited to hydrologic models, which use recursive numerical solutions of coupled nonlinear differential equations [e.g., *Kavetski et al.*, 2003]. For such models, hierarchical (e.g., input-error sensitive) Bayesian inference using standard multiblock MCMC (e.g., "Metropolis-within-Gibbs") is computationally expensive even for moderate calibration data lengths (e.g., a few years of daily data).

[4] This note shows why standard multiblock MCMC is prohibitively inefficient for full Bayesian CRR model inference and presents a general solution strategy. Following an outline of BATEA, we review multiblock samplers, emphasizing their computational cost given the recursive time-stepping nature of CRR models. A more efficient "limited-memory" MCMC algorithm is then designed and illustrated using the common GR4J model [*Perrin et al.*, 2003]. We conclude with a comment on a hybrid strategy for efficient MCMC-based Bayesian hierarchical inference of CRR models.

2. Outline of the BATEA Framework

2.1. Data Uncertainty

[5] Consider a time series of length N_t, $\boldsymbol{X} = \{X_{(m)}; m = 1,.., N_t\}$, where $X_{(m)}$ is the forcing at the mth time step. Next, consider N epochs $\{(m_i, m_{i+1} - 1); i = 1,.., N\}$, where m_i is the time-step index of the start of the ith epoch (e.g., storm or daily [*Thyer et al.*, 2009]). The observed forcing is $\tilde{\boldsymbol{X}}_i = \{\tilde{X}_{(m)}; m = m_i,.., m_{i+1} - 1\}$ and the true forcing is $\boldsymbol{X}_i$, while $\tilde{\boldsymbol{Y}}_i$ and $\boldsymbol{Y}_i$ are, respectively, the observed and true responses. Also, let $\boldsymbol{X}_{1:i} = \{\boldsymbol{X}_k; k = 1,.., i\}$.

[6] Let a function $\boldsymbol{X}_i = I(\tilde{\boldsymbol{X}}_i|\varphi_i)$ relate actual and observed forcings, for example, $\boldsymbol{X}_i = \varphi_i\tilde{\boldsymbol{X}}_i$ for all steps of the ith epoch [*Kavetski et al.*, 2006], where φ_i is an epoch-dependent multiplicative error, treated as a latent (unobservable) variable with "hyperdistribution"

$$\varphi_i \sim p(\varphi|\boldsymbol{\Phi}), \tag{1}$$

where the "hyperparameters" $\boldsymbol{\Phi}$ describe, for example, the mean and variance of φ.

[7] Also consider a streamflow error model [*Thyer et al.*, 2009],

$$\tilde{\boldsymbol{Y}}_i \sim p(\tilde{\boldsymbol{Y}}|\boldsymbol{\Xi}), \tag{2}$$

[1]School of Engineering, University of Newcastle, Callaghan, New South Wales, Australia.
[2]UR HHLY, Hydrology-Hydraulics, Cemagref, Lyon, France.

0043-1397/10/2009WR008985

where Ξ characterize response errors (e.g., variance of rating curve errors).

2.2. CRR Models and Their Recursive Structure

[8] The CRR model $\boldsymbol{H}$ maps the forcings into simulated responses $\hat{\boldsymbol{Y}}_i$. The majority of CRR models are based on numerical solutions of initial-value differential equations (DEs) describing time changes in conceptual stores $\boldsymbol{S}$ such as groundwater, soil, and stream, connected via hypothesized fluxes $\boldsymbol{g}$ [e.g., *Kavetski et al.*, 2003]

$$\hat{\boldsymbol{Y}}_i = \boldsymbol{H}(\boldsymbol{X}_{1:i}, \boldsymbol{\lambda}_{1:i}, \boldsymbol{\theta}) \tag{3a}$$

$$\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}t} = \boldsymbol{g}(\boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{\theta}, \boldsymbol{S}) \Rightarrow \boldsymbol{S}_{i+1} = \boldsymbol{f}(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, \boldsymbol{\theta}, \boldsymbol{S}_i) \tag{3b}$$

$$\hat{\boldsymbol{Y}}_{i+1} = \boldsymbol{h}(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, \boldsymbol{\theta}, \boldsymbol{S}_i) \tag{3c}$$

Equation (3) is formulated deterministically to conserve mass in each store [*Kuczera et al.*, 2006]. Here, $\boldsymbol{\theta}$ are the time-invariant CRR parameters and $\boldsymbol{\lambda}_i$ are epoch-specific CRR parameters,

$$\boldsymbol{\lambda}_i \sim p(\boldsymbol{\lambda}|\boldsymbol{\Lambda}), \tag{4}$$

where $\boldsymbol{\Lambda}$ are the CRR hyperparameters (e.g., means and variances of stochastic parameters).

[9] A key feature of virtually all CRR models is their recursive structure illustrated in equation (3). When applied to such models, BATEA is atypical of standard Bayesian hierarchical formulations [e.g., *Gelman et al.*, 2004; *Gilks et al.*, 1994] because the simulated response $\hat{\boldsymbol{Y}}_i$ depends on earlier epochs. For example, effects of a large rainfall error will persist because the induced storage errors affect streamflow over many subsequent steps.

2.3. BATEA Posterior Distribution

[10] BATEA infers the CRR parameters $\boldsymbol{\theta}$, latent variables $\{\boldsymbol{\varphi}, \boldsymbol{\lambda}\}$, and hyperparameters $\{\Phi, \boldsymbol{\Lambda}, \Xi\}$ given observed forcing-response data $\{\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}\}$, hypothesized error models and any prior information. To simplify the notation, define the complete set of N epoch-dependent latent variables $\boldsymbol{\omega}_{1:N} = \{\boldsymbol{\varphi}_{1:N}, \boldsymbol{\lambda}_{1:N}\}$, and the corresponding hyperparameters $\boldsymbol{\Omega} = \{\Phi, \boldsymbol{\Lambda}\}$. The BATEA posterior distribution is then

$$p(\boldsymbol{\theta}, \boldsymbol{\omega}_{1:N}, \boldsymbol{\Omega}, \Xi|\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}) \propto p(\tilde{\boldsymbol{Y}}|\boldsymbol{\theta}, \boldsymbol{\omega}_{1:N}, \Xi, \tilde{\boldsymbol{X}}) p(\boldsymbol{\omega}_{1:N}|\boldsymbol{\Omega}) p(\boldsymbol{\Omega}) p(\Xi) p(\boldsymbol{\theta}), \tag{5}$$

where $p(\tilde{\boldsymbol{Y}}|\boldsymbol{\theta}, \boldsymbol{\omega}_{1:N}, \Xi, \tilde{\boldsymbol{X}})$ is the likelihood function, $p(\boldsymbol{\omega}_{1:N}|\boldsymbol{\Omega})$ is the hyperdistribution of $\boldsymbol{\omega}_{1:N}$, and $p(\boldsymbol{\Omega})$, $p(\Xi)$, and $p(\boldsymbol{\theta})$ are priors [*Kuczera et al.*, 2006].

3. MCMC Methods for Hierarchical Bayesian Inference

3.1. General Metropolis-Hastings Sampler

[11] The Metropolis-Hastings (MH) algorithm is a general MCMC method for sampling from multivariate distributions [*Gelman et al.*, 2004]. If $p(\boldsymbol{\psi})$ is the target distribution [e.g., posterior (5)], the MH method samples a proposal $\boldsymbol{\psi}^{*(k+1)}$ from a jump distribution $J(\boldsymbol{\psi}|\boldsymbol{\psi}^{(k)})$ at the kth iteration. It accepts $\boldsymbol{\psi}^{(k+1)} = \boldsymbol{\psi}^{*(k+1)}$ with probability given by the jump ratio $r(\boldsymbol{\psi}^{*(k+1)} \mid \boldsymbol{\psi}^{(k)})$ below, otherwise $\boldsymbol{\psi}^{(k+1)} = \boldsymbol{\psi}^{(k)}$,

$$r\left(\boldsymbol{\psi}^{*(k+1)}|\boldsymbol{\psi}^{(k)}\right) = \frac{p\left(\boldsymbol{\psi}^{*(k+1)}\right) J\left(\boldsymbol{\psi}^{(k)}|\boldsymbol{\psi}^{*(k+1)}\right)}{p(\boldsymbol{\psi}^{(k)}) J\left(\boldsymbol{\psi}^{*(k+1)}|\boldsymbol{\psi}^{(k)}\right)}. \tag{6}$$

When the jump distribution J is symmetric (e.g., Gaussian centered on the current sample), the MCMC algorithm is referred to as a "Metropolis" scheme [*Gelman et al.*, 2004].

3.2. Multiblock MCMC Sampler

[12] The blocking of sampled variables considerably affects the efficiency of MCMC sampling [e.g., *Fu and Gomez-Hernandez*, 2009]. Sampling inferred quantities "all-at-once" leads to "single-block" schemes. However, since deriving and adapting efficient jump distributions for high-dimensional posteriors such as (5) is challenging, Bayesian literature and software tend to favor multiblock schemes using a sequence of low-dimensional jump distributions [*Gelman et al.*, 2004; *Gilks et al.*, 1994]. For BATEA, the following three-block sampler is natural:

[13] **Block 1:** Sample the hyperparameters $\boldsymbol{\Omega}^{(k+1)}$ from their conditional posterior

$$p\left(\boldsymbol{\Omega}|\boldsymbol{\omega}_{1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \Xi^{(k)}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}\right) \propto p\left(\boldsymbol{\omega}_{1:N}^{(k)}|\boldsymbol{\Omega}\right) p(\boldsymbol{\Omega}). \tag{7}$$

When conjugate hyperdistributions and priors are used, the conditional posterior (7) can be sampled analytically ("Gibbs sampling") [*Gelman et al.*, 2004]. More generally, however, a Metropolis acceptance-rejection step (section 2.1) is used to sample from the probability density function (pdf) in eqn (7).

[14] **Block 2:** Sample the latent variables $\boldsymbol{\omega}_{1:N}^{(k+1)}$ from their conditional posterior

$$p\left(\boldsymbol{\omega}_{1:N}|\boldsymbol{\theta}^{(k)}, \boldsymbol{\Omega}^{(k+1)}, \Xi^{(k)}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}\right) \propto p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\theta}^{(k)}, \boldsymbol{\omega}_{1:N}, \Xi^{(k)}, \tilde{\boldsymbol{X}}\right) p\left(\boldsymbol{\omega}_{1:N}|\boldsymbol{\Omega}^{(k+1)}\right). \tag{8}$$

The implementation of this step lies at the focus of this note and is detailed in the next section.

[15] **Block 3:** Sample the time-invariant CRR parameters and the output error parameters $(\boldsymbol{\theta}^{(k+1)}, \Xi^{(k+1)})$ from their conditional posterior

$$p\left(\boldsymbol{\theta}, \Xi|\boldsymbol{\omega}_{1:N}^{(k+1)}, \boldsymbol{\Omega}^{(k+1)}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}\right) \propto p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\theta}, \boldsymbol{\omega}_{1:N}^{(k+1)}, \Xi, \tilde{\boldsymbol{X}}\right) p(\boldsymbol{\theta}) p(\Xi) \tag{9}$$

Again, while conjugate probability models permit direct Gibbs sampling, in general sampling from (9) is implemented using a Metropolis iteration.

3.3. Epoch-by-Epoch MCMC Sampler

[16] Since direct sampling of $\boldsymbol{\omega}_{1:N}^{(k+1)}$ from its conditional posterior in Block 2 is usually impossible, MH sampling is used. The temporal structure of latent variables $\boldsymbol{\omega}$ suggests epoch-by-epoch sampling. For the jth epoch within the kth iteration, we sample $\boldsymbol{\omega}_j^{(k+1)}$ from the conditional posterior

$p(\boldsymbol{\omega}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Omega}^{(k+1)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}})$, with the MH jump ratio

$$r\left(\boldsymbol{\omega}_j^{*(k+1)}|\boldsymbol{\omega}_j^{(k)}\right) = \frac{p\left(\boldsymbol{\omega}_j^{*(k+1)}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Omega}^{(k+1)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{Y}}, \tilde{\boldsymbol{X}}\right)}{p\left(\boldsymbol{\omega}_j^{(k)}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Omega}^{(k+1)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{Y}}, \tilde{\boldsymbol{X}}\right)} \times \frac{J\left(\boldsymbol{\omega}_j^{(k)}|\boldsymbol{\omega}_j^{*(k+1)}\right)}{J\left(\boldsymbol{\omega}_j^{*(k+1)}|\boldsymbol{\omega}_j^{(k)}\right)}$$
$$= \frac{p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}\right)}{p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}\right)} \times \frac{p\left(\boldsymbol{\omega}_j^{*(k+1)}|\boldsymbol{\Omega}^{(k+1)}\right)}{p\left(\boldsymbol{\omega}_j^{(k)}|\boldsymbol{\Omega}^{(k+1)}\right)} \times \frac{J\left(\boldsymbol{\omega}_j^{(k)}|\boldsymbol{\omega}_j^{*(k+1)}\right)}{J\left(\boldsymbol{\omega}_j^{*(k+1)}|\boldsymbol{\omega}_j^{(k)}\right)}. \tag{10}$$

Such algorithms are often referred to as "Metropolis-within-Gibbs" [e.g., *Reichert and Mieleitner*, 2009; *Roberts and Rosenthal*, 2009].

3.4. The Likelihood Ratio: A Computational Bottleneck

[17] The evaluation of the likelihood ratio in (10),

$$\rho_j^{(k+1)} = \frac{p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}\right)}{p\left(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}\right)} \tag{11}$$

dominates the CPU cost of MCMC methods for physically motivated models $\boldsymbol{H}$ commonly used in environmental engineering contexts, because it requires the numerical solution of the (usually coupled nonlinear) differential equations underlying the model $\boldsymbol{H}$, which is far costlier than evaluating the hyperdistributions [e.g., *Fu and Gomez-Hernandez*, 2009].

[18] Ratio (11) can be expanded with respect to the epoch index j as

$$\begin{aligned}
\rho_j^{(k+1)} &= \prod_{i=1}^{N} \frac{q\left(\tilde{\boldsymbol{Y}}_i|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:N}^{(k)}\right)}{q\left(\tilde{\boldsymbol{Y}}_i|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:N}^{(k)}\right)} \\
&= \underbrace{\prod_{i=1}^{j-1} \frac{q\left(\tilde{\boldsymbol{Y}}_i|\boldsymbol{\omega}_{1:i}^{(k+1)}\right)}{q\left(\tilde{\boldsymbol{Y}}_i|\boldsymbol{\omega}_{1:i}^{(k+1)}\right)}}_{\text{past relative to } j} \times \underbrace{\frac{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}\right)}{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}\right)}}_{\text{present } j} \times \underbrace{\prod_{i=1}^{N-j} \frac{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+i:N}^{(k)}\right)}{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+i:N}^{(k)}\right)}}_{\text{future relative to } j} \\
&= \frac{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}\right)}{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}\right)} \times \prod_{i=1}^{N-j} \frac{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+i:N}^{(k)}\right)}{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+i:N}^{(k)}\right)},
\end{aligned} \tag{12}$$

where, for convenience, we define $q(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}, \boldsymbol{\omega}_j, \boldsymbol{\omega}_{j+1:N}) = p(\tilde{\boldsymbol{Y}}|\boldsymbol{\omega}_{1:j-1}, \boldsymbol{\omega}_j, \boldsymbol{\omega}_{j+1:N}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}})$.

[19] The "past relative to j" term drops out because epochs $1{:}j-1$ are causally independent from $\boldsymbol{\omega}_j$. However, the last term in (12) does not cancel out because changes in storages propagate into future epochs due to the recursive model structure (3),

$$\boldsymbol{s}_{j+1}^{*(k+1)} := f\left(\tilde{\boldsymbol{X}}_j, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{s}_j\right) \neq \boldsymbol{s}_{j+1}^{(k)} := f\left(\tilde{\boldsymbol{X}}_j, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{s}_j\right). \tag{13}$$

These memory effects critically impact on the computational efficiency of multiblock MCMC.

3.5. Implications for Hydrological Modeling

[20] Evaluation of (12) requires $N - j + 1$ epoch evaluations. Since the number of epochs is roughly proportional to the number of time steps, $N \propto N_t$, the computational cost for a fixed number of multiblock MCMC samples is $O(N_t^2)$, that is, is approximately quadratic (Figure 1). Standard multiblock MCMC hence quickly becomes very expensive even for moderate-length calibrations. Even the GR4J model [*Perrin et al.*, 2003], used operationally in French forecasting systems, with typical individual runtimes below 1 s for a few years of data, would require days or weeks to calibrate using input-error sensitive approaches. For distributed models, the cost is even more staggering (e.g., a single SWAT model run simulating soil moisture may require several minutes in operational settings [*Tolson and Shoemaker*, 2007]). Expected runtimes of standard multiblock MCMC analysis could then exceed months! Given the growing interest in using CRR models to gain insights into catchment dynamics and structural model errors [*Clark et al.*, 2008], which requires calibration of multiple model configurations in multiple catchments while accounting for input errors, such a computational burden is a serious practical impediment and is discouraging to a practitioner.

3.6. A Limited-Memory MCMC Sampler

[21] Reducing the computational cost of the inference requires addressing the storage memory issue. Simply ignoring it cuts the computational cost up to by a factor of N:

$$\rho_j^{(k+1)} = \frac{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}\right)}{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}\right)}. \tag{14}$$

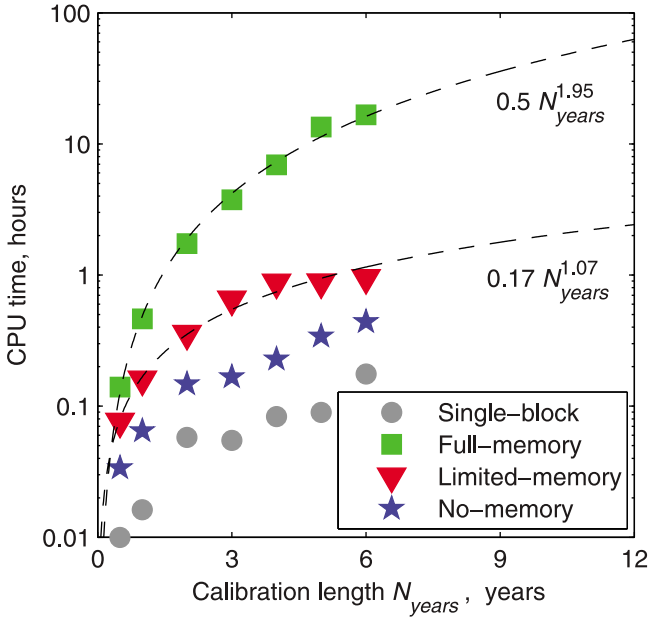

Figure 1. CPU time to generate 10,000 MCMC samples from the BATEA posterior of GR4J, as a function of the calibration data length. A 2.0 GHz laptop CPU with 1 GB of RAM was used.

However, applying (14) to a recursive model (3) can seriously alter its posterior distribution, degrading the quality of the inference. Fortunately, a much better alternative is possible.

[22] Note that the memory effect (13) decays over time because the CRR model "forgets" differences in the initial conditions at the jth epoch expressed by (13). The likelihood ratio can then be approximated by marching forward from epoch j and terminating after $M_j^{(k+1)}(\tau) \ll N$ epochs, when the likelihood ratio converges to within a prespecified numerical tolerance τ. Convergence can be tested as follows,

$$\rho_j^{(k+1)}(\tau) = \frac{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}\right)}{q\left(\tilde{\boldsymbol{Y}}_j|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}\right)} \times \prod_{i=1}^{M_j^{(k+1)}(\tau)} \frac{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)}\right)}{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)}\right)}, \quad (15)$$

$$M_j^{(k+1)}(\tau) := \min[i] \text{ such that } \left|\log\left(\frac{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)}\right)}{q\left(\tilde{\boldsymbol{Y}}_{j+i}|\boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)}\right)}\right)\right| \le \tau. \quad (16)$$

This approach exploits the decaying memory of CRR models and any other model based on (stable) initial-value DEs such as (3): the influence of initial conditions vanishes over time.

[23] We refer to algorithm (15) as the "limited-memory" MCMC sampler. The naming is inspired by "limited-memory" quasi-Newton methods for large-scale optimization [*Nocedal and Wright*, 1999], which also exploit the decaying memory of convergent recursive relations.

4. Empirical Assessment

4.1. Experimental Setup

[24] We now compare four MCMC samplers for CRR model inference, including three multiblock samplers differing in model memory treatment: (i) "full-memory" (12), (ii) "no-memory" (14), and (iii) limited-memory (15) with $\tau = 10^{-3}$. For consistency, all multiblock schemes sample one variable at a time using univariate Gaussian jump pdfs. A single-block Metropolis with a multivariate Gaussian jump pdf is used to independently confirm the accuracy of the samplers and to motivate an efficient hybrid MCMC strategy. Since the single-block Metropolis was pretuned over a series of trial runs, its practical computational cost is notably higher than may appear solely from the reported CPU time for generating the output samples.

[25] Note that the important topic of adaption of jump distributions [e.g., *Roberts and Rosenthal*, 2009; *Vrugt et al.*, 2009] lies largely outside the scope of this technical note, which focuses strictly on accelerating the evaluation of the jump ratio. The multiblock samplers were tuned based on jump rates [*Gelman et al.*, 2004], while the single-block Metropolis was preoptimized using the results of the multiblock sampler. The same statistical models and assumptions were employed in all MCMC methods, ensuring that differences between the multiblock samplers are dominated by the treatment of model memory.

[26] The accuracy and efficiency of the samplers were stringently verified using a synthetic case study. The "true" inputs comprised 6 years of observed daily rainfall and potential evapotranspiration for the 129 km^2 Yzeron catchment (France). The GR4J model [*Perrin et al.*, 2003] simulated the "true" daily streamflow using known "true" parameters. The "observed" streamflow was corrupted with 10% heteroscedastic Gaussian errors, while the "observed" rainfall was corrupted with log-normal multiplicative errors, $\log_e \varphi \sim N(0.0, 0.25^2)$.

4.2. Results and Discussion

[27] Figure 1 reports the CPU time to generate 10,000 MCMC samples and its dependence on the calibration data length, while Figure 2 shows the posterior distributions estimated from 1 year of data (86 epochs). Figure 1 lucidly illustrates the rapid CPU cost growth of the full-memory sampler with increasing calibration periods. As expected from algorithmic considerations (section 2.5), CPU time increases approximately quadratically with N_t, prohibiting the use of long calibration data sets. However, while the no-memory algorithm drastically cuts the CPU time, it provides a very poor approximation of the actual posterior. Even allowing a one-epoch memory [M = 1 in equation (15)] results in a significantly misspecified mode and a markedly overestimated posterior uncertainty, while the no-memory approximation was off the charts. This confirms that uncontrollably modifying the model to discard its history is unacceptable.

[28] In contrast, the limited-memory algorithm provides a very close approximation to the distributions obtained using the full-memory and single-block schemes. This confirms the

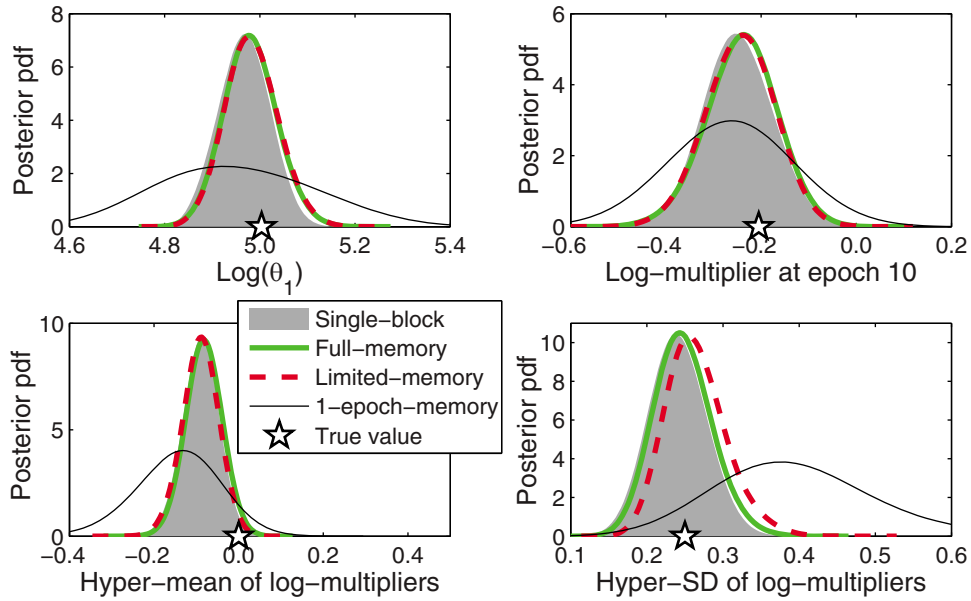

Figure 2. Posterior distributions of selected quantities estimated using different MCMC samplers (100,000 samples). "Hyper-SD" is the standard deviation of hyperdistribution (1).

robustness of the convergence test (16), which ensures that the jump ratio of the limited-memory algorithm is within a tolerance of the jump ratio of the full-memory method. Note that while both the full-memory and single-block methods converge to the target posterior (5), for a finite number of samples they inevitably exhibit minor discrepancies due to (1) different autocorrelation structure of "epoch-by-epoch" versus "all-at-once" sampling and (2) histogram smoothing to estimate the underlying probability densities. Since in practice MCMC methods are seldom run to perfect convergence (nor is this even feasible in most cases), the discrepancies in Figure 2 are within MCMC sampling variability and other approximation errors. Importantly, tightening the tolerance τ forces a progressively closer agreement between the limited-memory method and its full-memory counterpart.

[29] The CPU cost of the limited-memory sampler is near linear with respect to calibration length. In particular, it was only 2–4 times slower than the no-memory sampler. In general, the computational acceleration of the limited-memory approximation depends on the calibration data and its epochs, the catchment response time, the CRR model, and the limited-memory tolerance τ. In this study, Figure 1 suggests an acceleration by a factor of 20 for the GR4J model applied to 6 years of daily data (463 epochs) with memory tolerance $\tau = 10^{-3}$.

[30] Finally, the single-block ("all-at-once") sampler with *pretuned* jump distributions is generally more efficient than multiblock schemes because it requires only a single CRR model run per sample. However, in the absence of tuning it can be very inefficient and slowly convergent because a poorly selected high-dimensional jump distribution can lead to particularly poor mixing of the MCMC chains (e.g., see *Fu and Gomez-Hernandez* [2009] for an analysis of the effect of block-size on MCMC convergence). Moreover, adapting a high-dimensional jump distribution creates a considerable overhead not reported in this technical note because it is case specific and depends on the MCMC initialization and adaption strategies.

[31] Given the difficulty in tuning high-dimensional jump distributions, a hybrid MCMC strategy that exploits the limited-memory multiblock sampler to estimate a good jump distribution for a single-block Metropolis sampler can be advantageous. Since the multiblock sampler uses simple univariate Gaussian distributions in all blocks, their variances can be readily estimated and tuned. Once sufficient samples have been obtained, the entire covariance matrix can be estimated and kept fixed in a single-block Metropolis sampler. The design and evaluation of the hybrid MCMC strategy will be detailed in a separate study.

5. Concluding Remarks

[32] Hierarchical methods such as BATEA hold considerable promise for environmental modeling (see *Cressie et al.* [2009] for a state-of-the-art discussion). However, standard multiblock MCMC samplers (e.g., Metropolis-within-Gibbs) commonly used in the Bayesian hierarchical literature are computationally infeasible for recursive hydrological models simulating time-evolving storages (e.g., soil and groundwater). A careful "limited-memory" implementation of the jump ratio in the multiblock MCMC algorithm, exploiting the decaying memory of hydrological systems, overcomes the computational inefficiency, while controlling the accuracy using a numerical tolerance. We stress the broad applicability of the limited-memory acceleration strategy detailed in this note: it can be exploited by other hierarchical Bayesian MCMC formulations [*Cressie et al.*, 2009], and, more generally, it can be used for other computationally expensive recursive models with decaying memory.

References

Clark, M. P., A. G. Slater, D. E. Rupp, R. A. Woods, J. A. Vrugt, H. V. Gupta, T. Wagener, and L. E. Hay (2008), Framework for Understanding Structural Errors (FUSE): A modular framework to diagnose differences between hydrological models, *Water Resour. Res.*, *44*, W00B02, doi:10.1029/2007WR006735.

Cressie, N., C. A. Calder, J. S. Clark, J. M. ver Hoef, and C. K. Wikle (2009), Accounting for uncertainty in ecological analysis: the strengths and limitations of hierarchical statistical modeling, *Ecol. Appl.*, *19*(3), 553–570.

Fu, J., and J. J. Gomez-Hernandez (2009), A blocking Markov Chain Monte Carlo method for inverse stochastic hydrogeological modeling, *Math Geosci.*, *41*, 105–128.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin (2004), *Bayesian Data Analysis*, Chapman.

Gilks, W. R., A. Thomas, and D. J. Spiegelhalter (1994), A language and program for complex Bayesian modeling, *Statistician*, *43*(1), 169–177.

Kavetski, D., G. Kuczera, and S. W. Franks (2003), Semidistributed hydrological modeling: A "saturation path" perspective on TOPMODEL and VIC, *Water Resour. Res.*, *39*(9), 1246, doi:10.1029/2003WR002122.

Kavetski, D., G. Kuczera, and S. W. Franks (2006), Bayesian analysis of input uncertainty in hydrological modeling, 1. Theory, *Water Resour. Res.*, *42*(3), W03407, doi:10.1029/2005WR004368.

Kuczera, G., D. Kavetski, S. Franks, and M. Thyer (2006), Towards a Bayesian total error analysis of conceptual rainfall-runoff models: Characterising model error using storm-dependent parameters, *J. Hydrol.*, *331*(1–2), 161–177.

Perrin, C., C. Michel, and V. Andreassian (2003), Improvement of a parsimonious model for streamflow simulation, *J. Hydrol.*, *279*(1–4), 275–289.

Reichert, P., and J. Mieleitner (2009), Analyzing input and structural uncertainty of nonlinear dynamic models with stochastic, time-dependent parameters, *Water Resour. Res.*, *45*, W10402, doi:10.1029/2009WR007814.

Roberts, G. O., and J. S. Rosenthal (2009), Examples of adaptive MCMC, *J. Comput. Graphical Statistics*, *18*(2), 349–367.

Thyer, M., B. Renard, D. Kavetski, G. Kuczera, S. Franks, and S. Srikanthan (2009), Critical evaluation of parameter consistency and predictive uncertainty in hydrological modelling: A case study using Bayesian total error analysis, *Water Resour. Res.*, *45*, W00B14, doi:10.1029/2008WR006825.

Tolson, B. A., and C. A. Shoemaker (2007), Dynamically dimensioned search algorithm for computationally efficient watershed model calibration, *Water Resour. Res.*, *43*(1), W01413, doi:10.1029/2005WR004723.

Vrugt, J. A., C. G. H. Diks, H. V. Gupta, W. Bouten, and J. M. Verstraten (2005), Improved treatment of uncertainty in hydrologic modeling: Combining the strengths of global optimization and data assimilation, *Water Resour. Res.*, *41*(1), W01017, doi:10.1029/2004WR003059.

Vrugt, J. A., C. J. F. ter Braak, C. G. H. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon (2009), Accelerating Markov Chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling, *Int. J. Nonlinear Sci. Numer. Simul.*, *10*(3), 273–290.

D. Kavetski, G. Kuczera, and M. Thyer, School of Engineering, University of Newcastle, Callaghan, NSW 2308, Australia.

B. Renard, UR HHLY, Hydrology-Hydraulics, Cemagref, 3 bis quai Chauveau, CP 220, F-69336 Lyon CEDEX 09, France.