



HAL
open science

A limited-memory acceleration strategy for mcmc sampling in hierarchical bayesian calibration of hydrological models

G. Kuczera, D. Kavetski, Benjamin Renard, M. Thyer

► **To cite this version:**

G. Kuczera, D. Kavetski, Benjamin Renard, M. Thyer. A limited-memory acceleration strategy for mcmc sampling in hierarchical bayesian calibration of hydrological models. *Water Resources Research*, 2010, 46, 15 p. 10.1029/2009WR008985 . hal-00506554

HAL Id: hal-00506554

<https://hal.science/hal-00506554v1>

Submitted on 28 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

A LIMITED-MEMORY ACCELERATION STRATEGY FOR MCMC SAMPLING IN HIERARCHICAL BAYESIAN CALIBRATION OF HYDROLOGICAL MODELS

George Kuczera¹, Dmitri Kavetski¹, Benjamin Renard² and Mark Thyer¹

1 School of Engineering, University of Newcastle, Callaghan, NSW, 2308, Australia

2 Cemagref, UR HHLY, Hydrology-Hydraulics. 3 bis quai Chauveau - CP 220, F-69336 Lyon, France

Abstract

Hydrological calibration and prediction using conceptual models is affected by forcing/response data uncertainty and structural model error. The Bayesian Total Error Analysis methodology (BATEA) uses a hierarchical framework to represent individual sources of uncertainty. However, it is shown that standard multi-block “Metropolis-within-Gibbs” samplers commonly used in traditional Bayesian hierarchical Markov Chain Monte Carlo (MCMC) are exceedingly computationally expensive when applied to hydrologic models based on recursive numerical solution of coupled nonlinear differential equations describing the evolution of catchment states such as soil and groundwater storages. This note develops a “limited-memory” algorithm for accelerating multi-block 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.

27 Introduction

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

35 Hierarchical inferences are usually implemented using Gibbs sampling, or, more generally,
36 multi-block Markov Chain Monte Carlo (MCMC). However, many statistical applications and
37 software [e.g., BUGS, *Gilks et al.*, 1994] are not well suited to hydrologic modeling, which
38 generally requires inference in coupled nonlinear differential equations [e.g., *Kavetski et al.*,
39 2003]. For such models, hierarchical (e.g., input-error sensitive) Bayesian inference using
40 standard multi-block MCMC (e.g., “Metropolis-within-Gibbs”) is computationally expensive
41 even for moderate calibration data lengths (e.g., a few years of daily data).

42 This note shows why standard multi-block MCMC samplers are inefficient for full Bayesian
43 CRR inference and presents a general solution strategy. Following an outline of BATEA, we
44 detail multi-block samplers with an emphasis on their computational cost given the recursive
45 time stepping nature of CRR models. A more efficient “limited-memory” MCMC algorithm is
46 then designed and illustrated using the common GR4J model. We conclude with a comment on
47 a hybrid strategy for efficient MCMC-based Bayesian hierarchical inference of CRR models.

48 1 Outline of the BATEA framework

49 1.1 Data uncertainty

50 Consider a time series of length N_t , $X = \{X_{(m)}; m = 1, \dots, N_t\}$, where $X_{(m)}$ is the forcing at the

51 m th time step. Next, consider N epochs $\{(m_i, m_{i+1} - 1); i = 1, \dots, N\}$, where m_i is the time step
 52 index of the start of the i^{th} epoch (e.g., storm or daily [Thyer *et al.*, 2009]). The observed forcing
 53 is $\tilde{X}_i = \{\tilde{X}_{(m)}; m = m_i, \dots, m_{i+1} - 1\}$ and the true forcing is X_i , while \tilde{Y}_i and Y_i are, respectively,
 54 the observed and true responses.

55 Let a function $X_i = I(\tilde{X}_i | \varphi_i)$ relate actual and observed forcings, e.g., $X_i = \varphi_i \tilde{X}_i$ for all steps
 56 of the i^{th} storm event [Kavetski *et al.*, 2006], where φ_i is a storm-dependent multiplicative error,
 57 treated as a latent variable with “hyper-distribution”

$$58 \quad \varphi_i \sim p(\varphi | \Phi) \quad (1)$$

59 where the “hyper-parameters” Φ describe, e.g., the mean and variance of φ .

60 A streamflow error model must also be specified [Thyer *et al.*, 2009],

$$61 \quad \tilde{Y}_i \sim p(\tilde{Y} | \Xi) \quad (2)$$

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

63 **1.2 CRR models and their recursive structure**

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

$$68 \quad \begin{aligned} \hat{Y}_i &= H(X_{1:i}, \lambda_{1:i}, \theta) & (a) \\ \frac{dS}{dt} &= g(X, \lambda, \theta, S) \Rightarrow S_{i+1} = f(X_i, \lambda_i, \theta, S_i) & (b) \\ \hat{Y}_{i+1} &= h(X_i, \lambda_i, \theta, S_i) & (c) \end{aligned} \quad (3)$$

69 Eqn (3) is formulated deterministically to conserve mass in each store [*Kuczera et al.*, 2006].

70 Here, θ are the time-invariant CRR parameters and λ_i are epoch-specific CRR parameters,

$$71 \quad \lambda_i \sim p(\lambda | \mathcal{A}) \quad (4)$$

72 where \mathcal{A} are the CRR hyper-parameters, e.g., means and variances of stochastic parameters.

73 A key feature of virtually all CRR models is their recursive structure illustrated in eqn (3).

74 When applied to such models, BATEA is atypical of standard Bayesian hierarchical

75 formulations [e.g., *Gelman et al.*, 2004; *Gilks et al.*, 1994] because the simulated response \hat{Y}_i

76 depends on earlier epochs. For example, effects of a large rainfall error will persist because the

77 induced storage errors affect streamflow over many subsequent steps.

78 **1.3 BATEA posterior distribution**

79 BATEA infers the CRR parameters θ , latent variables $\{\varphi, \lambda\}$ and hyper-parameters

80 $\{\Phi, \mathcal{A}, \Xi\}$ given observed forcing-response data $\{\tilde{X}, \tilde{Y}\}$ and any prior information. To simplify

81 the notation, define the complete set of N epoch-dependent latent variables $\omega_{1:N} = \{\varphi_{1:N}, \lambda_{1:N}\}$,

82 and the corresponding hyper-parameters $\Omega = \{\Phi, \mathcal{A}\}$. The BATEA posterior pdf is then

$$83 \quad p(\theta, \omega_{1:N}, \Omega, \Xi | \tilde{X}, \tilde{Y}) \propto p(\tilde{Y} | \theta, \omega_{1:N}, \Xi, \tilde{X}) p(\omega_{1:N} | \Omega) p(\Omega) p(\Xi) p(\theta) \quad (5)$$

84 where $p(\tilde{Y} | \theta, \omega_{1:N}, \Xi, \tilde{X})$ is the likelihood function, $p(\omega_{1:N} | \Omega)$ is the hyper-distribution of

85 $\omega_{1:N}$, and $p(\Omega)$, $p(\Xi)$ and $p(\theta)$ are priors [*Kuczera et al.*, 2006].

86 **2 MCMC methods for hierarchical Bayesian inference**

87 **2.1 General Metropolis-Hastings sampler**

88 The Metropolis-Hastings (MH) algorithm is a general MCMC method for sampling from

89 multivariate distributions [*Gelman et al.*, 2004]. If $p(\psi)$ is the target distribution, e.g.,

90 posterior (5), the MH method samples a proposal $\boldsymbol{\psi}^{*(k+1)}$ from a jump distribution $J(\boldsymbol{\psi} | \boldsymbol{\psi}^{(k)})$
 91 at the k th iteration. It accepts $\boldsymbol{\psi}^{(k+1)} = \boldsymbol{\psi}^{*(k+1)}$ with probability given by the jump ratio
 92 $r(\boldsymbol{\psi}^{*(k+1)} | \boldsymbol{\psi}^{(k)})$ below, otherwise $\boldsymbol{\psi}^{(k+1)} = \boldsymbol{\psi}^{(k)}$,

$$93 \quad r(\boldsymbol{\psi}^{*(k+1)} | \boldsymbol{\psi}^{(k)}) = \frac{p(\boldsymbol{\psi}^{*(k+1)}) J(\boldsymbol{\psi}^{(k)} | \boldsymbol{\psi}^{*(k+1)})}{p(\boldsymbol{\psi}^{(k)}) J(\boldsymbol{\psi}^{*(k+1)} | \boldsymbol{\psi}^{(k)})} \quad (6)$$

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

96 **2.2 Multi-block MCMC sampler**

97 The blocking of sampled variables considerably affects the efficiency of MCMC sampling [e.g.,
 98 *Fu and Gomez-Hernandez*, 2009]. Sampling inferred quantities ‘‘all-at-once’’ leads to ‘‘single-
 99 block’’ schemes. However, since deriving and adapting efficient jump distributions for high-
 100 dimensional posteriors such as (5) is challenging, Bayesian literature and software tend to favor
 101 multi-block schemes using a sequence of low-dimensional jump distributions [Gelman *et al.*,
 102 2004; Gilks *et al.*, 1994]. For BATEA, the following three-block sampler is natural:

103 **Block 1:** Sample the hyper-parameters $\boldsymbol{\Omega}^{(k+1)}$ from their conditional posterior

$$104 \quad p(\boldsymbol{\Omega} | \boldsymbol{\omega}_{1:N}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{\Xi}^{(k)}, \tilde{\boldsymbol{X}}, \tilde{\boldsymbol{Y}}) \propto p(\boldsymbol{\omega}_{1:N}^{(k)} | \boldsymbol{\Omega}) p(\boldsymbol{\Omega}) \quad (7)$$

105 where the simplification occurs due to the hierarchical structure of (5).

106 When conjugate hyper-distributions and priors are used, conditional posterior (7) can be
 107 sampled analytically (‘‘Gibbs sampling’’) [Gelman *et al.*, 2004]. More generally, however, a
 108 Metropolis acceptance-rejection step (Section 2.1) is used to sample from pdf (7).

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

$$110 \quad p(\omega_{1:N} | \theta^{(k)}, \Omega^{(k+1)}, \Xi^{(k)}, \tilde{X}, \tilde{Y}) \propto p(\tilde{Y} | \theta^{(k)}, \omega_{1:N}, \Xi^{(k)}, \tilde{X}) p(\omega_{1:N} | \Omega^{(k+1)}) \quad (8)$$

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

112 **Block 3:** Sample the time-invariant CRR parameters and the output error parameters

113 $(\theta^{(k+1)}, \Xi^{(k+1)})$ from their conditional posterior

$$114 \quad p(\theta, \Xi | \omega_{1:N}^{(k+1)}, \Omega^{(k+1)}, \tilde{X}, \tilde{Y}) \propto p(\tilde{Y} | \theta, \omega_{1:N}^{(k+1)}, \Xi, \tilde{X}) p(\theta) p(\Xi) \quad (9)$$

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

117 **2.3 Epoch-by-epoch MCMC sampler**

118 Since direct sampling of $\omega_{1:N}^{(k+1)}$ from its conditional posterior in Block 2 is usually impossible,

119 MH sampling is used within this block. The temporal structure of latent variables ω suggests
120 epoch-by-epoch sampling. For the j^{th} epoch within the k^{th} iteration, we sample $\omega_j^{(k+1)}$ from the

121 conditional posterior $p(\omega_j | \omega_{1:j-1}^{(k+1)}, \omega_{j+1:N}^{(k)}, \theta^{(k)}, \Omega^{(k+1)}, \Xi^{(k)}, \tilde{X}, \tilde{Y})$, with the MH jump ratio

$$122 \quad r(\omega_j^{*(k+1)} | \omega_j^{(k)}) = \frac{p(\omega_j^{*(k+1)} | \omega_{1:j-1}^{(k+1)}, \omega_{j+1:N}^{(k)}, \theta^{(k)}, \Omega^{(k+1)}, \Xi^{(k)}, \tilde{Y}, \tilde{X})}{p(\omega_j^{(k)} | \omega_{1:j-1}^{(k+1)}, \omega_{j+1:N}^{(k)}, \theta^{(k)}, \Omega^{(k+1)}, \Xi^{(k)}, \tilde{Y}, \tilde{X})} \times \frac{J(\omega_j^{(k)} | \omega_j^{*(k+1)})}{J(\omega_j^{*(k+1)} | \omega_j^{(k)})} \quad (10)$$

$$= \frac{p(\tilde{Y} | \omega_{1:j-1}^{(k+1)}, \omega_j^{*(k+1)}, \omega_{j+1:N}^{(k)}, \theta^{(k)}, \Xi^{(k)}, \tilde{X})}{p(\tilde{Y} | \omega_{1:j-1}^{(k+1)}, \omega_j^{(k)}, \omega_{j+1:N}^{(k)}, \theta^{(k)}, \Xi^{(k)}, \tilde{X})} \times \frac{p(\omega_j^{*(k+1)} | \Omega^{(k+1)})}{p(\omega_j^{(k)} | \Omega^{(k+1)})} \times \frac{J(\omega_j^{(k)} | \omega_j^{*(k+1)})}{J(\omega_j^{*(k+1)} | \omega_j^{(k)})}$$

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

125 **2.4 The likelihood ratio: A computational bottleneck**

126 The evaluation of the likelihood ratio in (10),

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

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

Given (2) and (3), ratio (11) can be expanded with respect to the epoch index j as

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

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

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),

$$s_{j+1}^{*(k+1)} := f(\tilde{\mathbf{X}}_j, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\theta}^{(k)}, s_j) \neq s_{j+1}^{(k)} := f(\tilde{\mathbf{X}}_j, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\theta}^{(k)}, s_j) \quad (13)$$

These memory effects critically impact on the computational efficiency of multi-block MCMC.

2.5 Implications for hydrological modeling

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 multi-block MCMC samples is $O(N_t^2)$, i.e., is approximately quadratic (Figure 1). Standard

144 multi-block MCMC hence quickly becomes very expensive even for moderate-length
 145 calibrations. Even the GR4J model [Perrin *et al.*, 2003], used operationally in French
 146 forecasting systems, with typical individual runtimes below 1 sec for a few years of data, would
 147 require days or weeks to calibrate using input-error sensitive approaches. For distributed
 148 models, the cost is even more staggering. E.g., a single SWAT model run simulating soil
 149 moisture in irrigated landscapes may require several minutes in operational settings [Tolson and
 150 Shoemaker, 2007]. Expected runtimes of standard multi-block MCMC analysis could then
 151 exceed months! Given the growing interest in using CRR models to gain insights into
 152 catchment dynamics and structural model errors [Clark *et al.*, 2008], which requires calibration
 153 of multiple model configurations in multiple catchments while accounting for input errors, such
 154 computational burden is a serious practical impediment.

155 **2.6 A limited-memory MCMC sampler**

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

$$158 \quad \rho_j^{(k+1)} = \frac{q(\tilde{\mathbf{Y}}_j | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)})}{q(\tilde{\mathbf{Y}}_j | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)})} \quad (14)$$

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

161 Note that the memory effect (13) decays over time because the CRR model “forgets”
 162 differences in the initial conditions at the j^{th} epoch expressed by (13). The likelihood ratio can
 163 then be approximated by marching forward from epoch j and terminating after $M_j^{(k+1)}(\tau) \ll N$
 164 epochs, when the likelihood ratio converges to within a pre-specified numerical tolerance τ .

$$165 \quad \rho_j^{(k+1)}(\tau) = \frac{q(\tilde{\mathbf{Y}}_j | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)})}{q(\tilde{\mathbf{Y}}_j | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)})} \times \prod_{i=1}^{M_j^{(k+1)}(\tau)} \frac{q(\tilde{\mathbf{Y}}_{j+i} | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)})}{q(\tilde{\mathbf{Y}}_{j+i} | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)})} \quad (15)$$

$$166 \quad M_j^{(k+1)}(\tau) := \min[i] \text{ such that } \left| \log \left(\frac{q(\tilde{Y}_{j+i} | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{*(k+1)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)})}{q(\tilde{Y}_{j+i} | \boldsymbol{\omega}_{1:j-1}^{(k+1)}, \boldsymbol{\omega}_j^{(k)}, \boldsymbol{\omega}_{j+1:j+i}^{(k)})} \right) \right| \leq \tau \quad (16)$$

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

169 We refer to algorithm (15) as the “limited-memory” MCMC sampler. The naming is inspired
170 by “limited-memory” quasi-Newton methods for large-scale optimization [Nocedal and Wright,
171 1999], which also exploit the decaying memory of convergent recursive relations.

172 **3 Empirical assessment**

173 **3.1 Experimental setup**

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

182 Note that the important topic of adaption of jump distributions [e.g., Roberts and Rosenthal,
183 2009] lies largely outside the scope of this technical note, which focuses strictly on accelerating
184 the evaluation of the jump ratio. The multi-block samplers were tuned based on jump rates
185 [Gelman et al., 2004], while the single-block Metropolis was pre-optimized using the results of
186 the multi-block sampler. The same statistical models and assumptions were employed in all
187 MCMC methods, ensuring that differences between the multi-block samplers were dominated
188 by the treatment of model memory. Note that while all methods converge to the target posterior

189 (5), for a finite number of samples they inevitably exhibit minor discrepancies due to: (i)
190 different autocorrelation structure of “epoch-by-epoch” versus “all-at-once” sampling, and (ii)
191 histogram smoothing to estimate the underlying probability densities.

192 The accuracy and efficiency of the samplers were stringently verified using a synthetic case
193 study. The “true” inputs comprised 6 years of observed daily rainfall and potential
194 evapotranspiration for the 144 km² Yzeron catchment (France). The GR4J model [Perrin *et al.*,
195 2003] simulated the “true” daily streamflow using known “true” parameters. The “observed”
196 streamflow was corrupted with 10% heteroscedastic Gaussian errors, while the “observed”
197 rainfall was corrupted with log-normal multiplicative errors, $\log_e \varphi \sim N(0.0, 0.25^2)$.

198 **3.2 Results and discussion**

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

209 In contrast, the limited-memory algorithm provides a very close approximation to the
210 distributions obtained using the full-memory and single-block Metropolis schemes. This
211 confirms the robustness of the convergence test (15), which ensures that the jump ratio of the
212 limited-memory algorithm is within a tolerance of the jump ratio of the full-memory method.

213 Since in practice MCMC methods are seldom run to perfect convergence (nor is this even
214 feasible in most cases), the discrepancies in Figure 2 are within MCMC sampling variability
215 and other approximation errors. Importantly, tightening the tolerance τ forces a progressively
216 closer agreement between the limited-memory method and its full-memory counterpart.

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

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

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

238 **4 Concluding remarks**

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

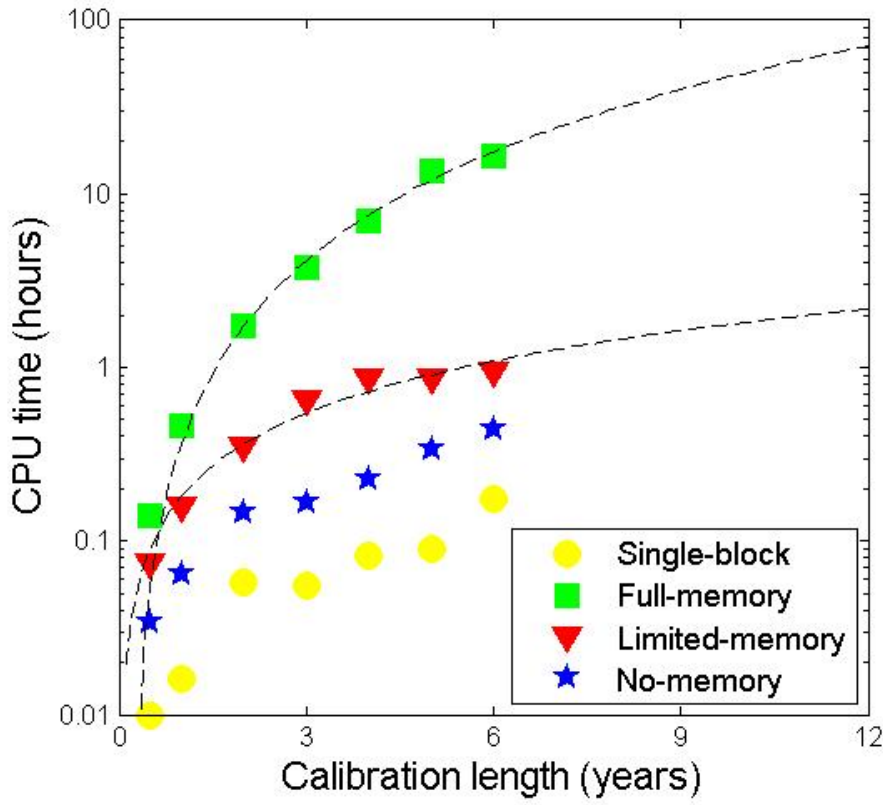
250 **References**

- 251 Clark, M. P., A. G. Slater, D. E. Rupp, R. A. Woods, J. A. Vrugt, H. V. Gupta, T. Wagener, and
252 L. E. Hay (2008), Framework for Understanding Structural Errors (FUSE): A modular
253 framework to diagnose differences between hydrological models, *Water Resources Res.*, *44*.
- 254 Cressie, N., C. A. Calder, J. S. Clark, J. M. ver Hoef, and C. K. Wikle (2009), Accounting for
255 uncertainty in ecological analysis: the strengths and limitations of hierarchical statistical
256 modeling, *Ecological Applications*, *19*(3), 553-570.
- 257 Fu, J., and J. J. Gomez-Hernandez (2009), A blocking Markov Chain Monte Carlo method for
258 inverse stochastic hydrogeological modeling, *Math Geosci*, *41*, 105-128.
- 259 Gelman, A., J. B. Carlin, H. S. Stern and D. B. Rubin (2004) *Bayesian data analysis*, Chapman.
- 260 Gilks, W. R., A. Thomas, and D. J. Spiegelhalter (1994), A Language and Program for
261 Complex Bayesian Modeling, *Statistician*, *43*(1), 169-177.

- 262 Kavetski, D., G. Kuczera, and S. W. Franks (2003), Semidistributed hydrological modeling: A
263 "saturation path" perspective on TOPMODEL and VIC, *Water Resources Research*, 39(9).
- 264 Kavetski, D., G. Kuczera, and S. W. Franks (2006), Bayesian analysis of input uncertainty in
265 hydrological modeling: 1. Theory, *Water Resources Research*, 42(3).
- 266 Kuczera, G., D. Kavetski, S. Franks, and M. Thyer (2006), Towards a Bayesian total error
267 analysis of conceptual rainfall-runoff models: Characterising model error using storm-
268 dependent parameters, *Journal of Hydrology*, 331(1-2), 161-177.
- 269 Perrin, C., C. Michel, and V. Andreassian (2003), Improvement of a parsimonious model for
270 streamflow simulation, *Journal of Hydrology*, 279(1-4), 275-289.
- 271 Reichert, P., and J. Mieleitner (2009), Analyzing input and structural uncertainty of nonlinear
272 dynamic models with stochastic, time-dependent parameters, *Water Resources Research*, 45.
- 273 Roberts, G. O., and J. S. Rosenthal (2009), Examples of adaptive MCMC, *Journal of*
274 *Computational and Graphical Statistics*, 18(2), 349-367.
- 275 Thyer, M., B. Renard, D. Kavetski, G. Kuczera, S. Franks, and S. Srikanthan (2009), Critical
276 evaluation of parameter consistency and predictive uncertainty in hydrological modelling: a
277 case study using bayesian total error analysis, *Water Resources Research*, 45.
- 278 Tolson, B. A., and C. A. Shoemaker (2007), Dynamically dimensioned search algorithm for
279 computationally efficient watershed model calibration, *Water Resources Research*, 43(1).
- 280 Vrugt, J. A., C. G. H. Diks, H. V. Gupta, W. Bouten, and J. M. Verstraten (2005), Improved
281 treatment of uncertainty in hydrologic modeling: Combining the strengths of global
282 optimization and data assimilation, *Water Resources Research*, 41(1).

283

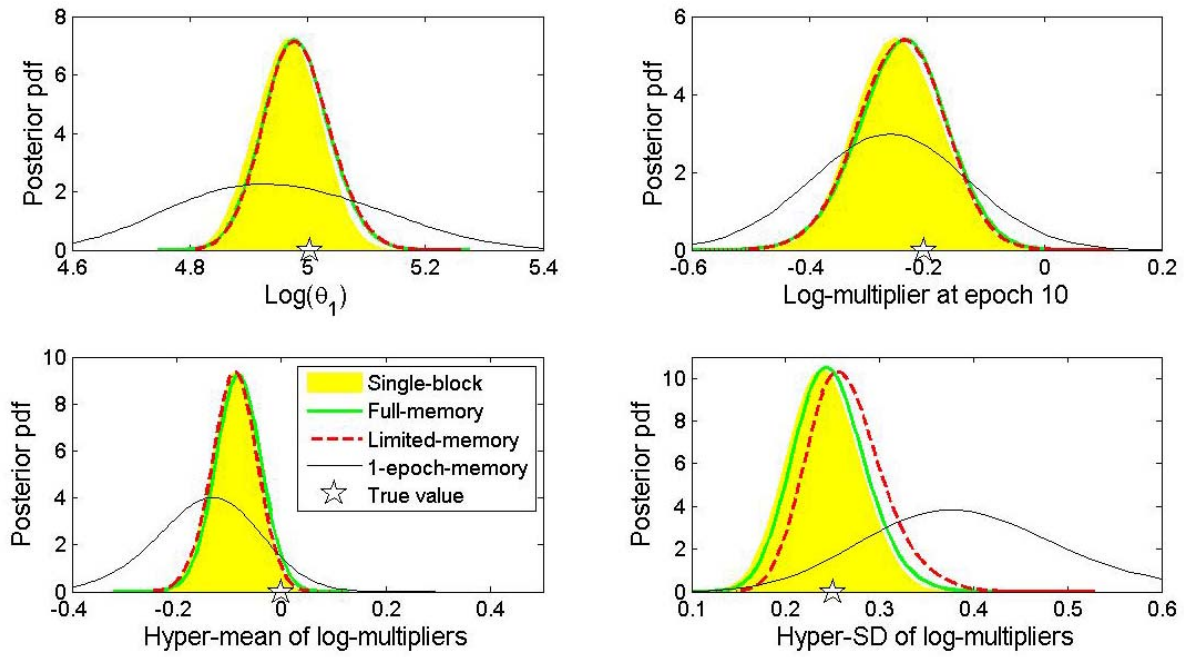
1
2
3
4



5
6
7
8

Figure 2. 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.

9
10



11

12 Figure 3. Posterior distributions of selected quantities estimated using different MCMC
13 samplers (100,000 samples). “Hyper-SD” is the standard deviation of hyper-distribution (1).

14