International Journal for Uncertainty Quantification, 2 (3): 215–237 (2012)

PARALLEL ADAPTIVE MULTILEVEL SAMPLING ALGORITHMS FOR THE BAYESIAN ANALYSIS OF MATHEMATICAL MODELS

Ernesto E. Prudencio[1,*] *& Sai Hung Cheung*[2]

[1] *Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, Texas, USA*

[2] *School of Civil and Environmental Engineering, Nanyang Technological University, Singapore*

Original Manuscript Submitted: 05/19/2011; Final Draft Received: 07/19/2011

In recent years, Bayesian model updating techniques based on measured data have been applied to many engineering and applied science problems. At the same time, parallel computational platforms are becoming increasingly more powerful and are being used more frequently by the engineering and scientific communities. Bayesian techniques usually require the evaluation of multi-dimensional integrals related to the posterior probability density function (PDF) of uncertain model parameters. The fact that such integrals cannot be computed analytically motivates the research of stochastic simulation methods for sampling posterior PDFs. One such algorithm is the adaptive multilevel stochastic simulation algorithm (AMSSA). In this paper we discuss the parallelization of AMSSA, formulating the necessary load balancing step as a binary integer programming problem. We present a variety of results showing the effectiveness of load balancing on the overall performance of AMSSA in a parallel computational environment.

KEY WORDS: *computational statistics, Markov chain Monte Carlo, adaptivity, Bayesian inference, stochastic modeling, model calibration*

1. INTRODUCTION

As computational power has been increasing for the last decades, and as parallel computing has become pervasive in the scientific and engineering communities, so has the applicability of Bayesian assessment of complex mathematical models and the use of companion Markov chain Monte Carlo (MCMC) [1–5] algorithms for sampling posterior probability density functions (PDFs). As the adoption of these algorithms becomes more popular, they tend naturally to migrate from uniprocessor to parallel computational environments, where many chains can be run at the same time, improving the statistical exploration of parameter spaces. However, such migration often has to be made with care in order to avoid computational load unbalancing; that is, in order to use parallel computational platforms efficiently.

Mathematical models serve two main purposes: representing a phenomenon and predicting the behavior of a system. The representation of a phenomenon usually involves (a) the comparison of model outputs and data measured at experimental scenarios from the phenomenon being modeled, as well as (b) the calibration of model parameters so that the agreement between model outputs and measured data is improved in some sense. Such calibrations involve the solution of so-called inverse problems [5–11]. The prediction of the behavior of a system, on the other hand, involves the solution of forward problems [12–18]: given (c) a model, eventually with calibrated parameters, and (d) a prediction scenario, what does the model tell us about the behavior of the modeled system and about the values of certain quantities of interest (QoIs)? Both inverse and forward problems can be treated deterministically or stochastically. In this paper we deal with stochastic treatments only.

*Correspond to Ernesto E. Prudencio, E-mail: prudenci@ices.utexas.edu, URL: http://www.ices.utexas.edu/

2152–5080/12/$35.00

During model construction, errors due to imperfect modeling and uncertainties due to incomplete information about the system and its environment always exist. Probability logic together with a Bayesian model updating approach provides a robust and rigorous framework for dealing with uncertainties. Probability logic [19, 20] is a multi-valued logic that extends Boolean propositional logic to the case of incomplete information [21]. It is consistent with the Bayesian point of view that probability represents a degree of belief in a proposition. Kolmogorov's axioms [22, 23], which are neutral with respect to the interpretation of probability, can be viewed as a special case where statements refer to uncertain membership of an object in a set [21]. A stochastic system based Bayesian framework for model updating was presented in [24, 25]. A key concept is a stochastic system model class ("model class" for short), which consists of a chosen set of probabilistic input-output models for a system and a chosen prior PDF. The prior PDF quantifies the initial probability of each predictive model in the set based on expert knowledge or past knowledge obtained from data. The probabilistic models represent a state of knowledge about the system, conditional on available information [1, 21, 24–27]. A model class can be viewed as a hypothesis that uses stochastic models and includes all assumptions and mathematical statements involved in the system modeling.

Let M_j be one model class. Bayes' theorem allows the update of the probability of each predictive model in M_j by combining measured data D with the prior PDF into the posterior PDF

$$\pi_{\text{posterior}}(\boldsymbol{\theta}|D, M_j) = \frac{f(D|\boldsymbol{\theta}, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)}{\pi(D|M_j)} = \frac{f(D|\boldsymbol{\theta}, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)}{\int f(D|\boldsymbol{\theta}, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta}},$$

where the denominator expresses the probability of getting the data D based on M_j and is called the evidence for M_j provided by D; $\pi_{\text{prior}}(\boldsymbol{\theta}|M_j)$ is the prior PDF of the predictive model $\boldsymbol{\theta}$ within M_j; and the likelihood function $f(D|\boldsymbol{\theta}, M_j)$ expresses the probability of getting D given the predictive model $\boldsymbol{\theta}$ within M_j. Stochastic models inside a model class M_j, therefore, can be compared. From now on we will omit the subscript j on the vector $\boldsymbol{\theta}$ of parameters for simplicity.

Different model choices that lead to different posterior PDFs also lead to different model classes. A multiple-stochastic system based Bayesian framework for model updating has been proposed in [1]. One can construct a set

$$\mathcal{M} = \{M_j, j = 1, \ldots, m\}$$

of *candidate model classes* and compare them with Bayes' theorem as well. Given the data D and the prior plausibility $p_{\text{prior}}(M_j|\mathcal{M})$ of M_j within $\mathcal{M}$, the posterior plausibility of such a model class is (e.g. [28])

$$p_{\text{posterior}}(M_j|D, \mathcal{M}) = \frac{\pi(D|M_j) \cdot p_{\text{prior}}(M_j|\mathcal{M})}{\sum_{j=1}^{m} \pi(D|M_j) \cdot p_{\text{prior}}(M_j|\mathcal{M})}.$$

The evidence of M_j thus influences its posterior plausibility. Also, the log evidence can be expressed as [28–30]

$$\ln[\pi(D|M_j)] = E[\ln(f(D|\boldsymbol{\theta}, M_j)] - E\left[\ln \frac{\pi_{\text{posterior}}(\boldsymbol{\theta}|D, M_j)}{\pi_{\text{prior}}(\boldsymbol{\theta}|M_j)}\right], \tag{1}$$

where the expectation is with respect to the posterior $\pi_{\text{posterior}}(\boldsymbol{\theta}|D, M_j)$ [1, 21, 26–30]. The posterior mean of the log likelihood is subtracted by the Kullback-Leibler divergence [31], which is always non-negative and measures the information gained about M_j from D. Equation (1) expresses a trade-off between the data fitness and the "complexity" of M_j and gives a quantitative information-theoretic version of the principle of model parsimony (Ockham's razor) [28–30]: simpler models that reasonably fit the data should be preferred over more complex models that only lead to slightly improved data fitness [32].

Bayesian model updating also allows us to make robust predictions [1, 21, 26, 27, 33], as follows. Given a candidate model class M_j, data D (e.g., past measurements on a data assimilation process) a vector $\mathbf{q}$ of QoIs, and a conditional PDF $\pi(\mathbf{q}|\boldsymbol{\theta}, D, M_j)$ of $\mathbf{q}$, the predictive PDF of $\mathbf{q}$ is [1, 30]

$$\pi_{\text{predicted}}(\mathbf{q}|D, M_j) = \int \pi(\mathbf{q}|\boldsymbol{\theta}, D, M_j) \cdot \pi_{\text{posterior}}(\boldsymbol{\theta}|D, M_j)\, d\boldsymbol{\theta}.$$

Given the set $\mathcal{M}$ of candidate model classes, the probabilistic information for $\mathbf{q}$ is [1, 28, 34, 35]

$$\pi(\mathbf{q}|D,\mathcal{M}) = \sum_{j=1}^{m} \pi(\mathbf{q}|D,M_j) \cdot p_{\text{posterior}}(M_j|D,\mathcal{M}),$$

an equation referred to as posterior model averaging in the Bayesian literature.

If a modeler (researcher, engineer, decision maker) has only one model class available (in the definition adopted above), has experimental data, and needs to make predictions, then her/his activities can be summarized as in Fig. 1: first she/he calibrates the model (that is, solves an inverse problem), and then makes predictions with it (that is, solves a forward problem).

On the other hand, if more than one model class is available, then it might be helpful to be able to compare different model classes, rank them according to some criteria, and use possibly a combination of their predictions. Such procedures are summarized in Fig. 2. It should be noted that the scientific community has been researching the parallelization of algorithms for both inverse and forward problems in order to handle high dimensional parameter spaces [6–11, 36–39].

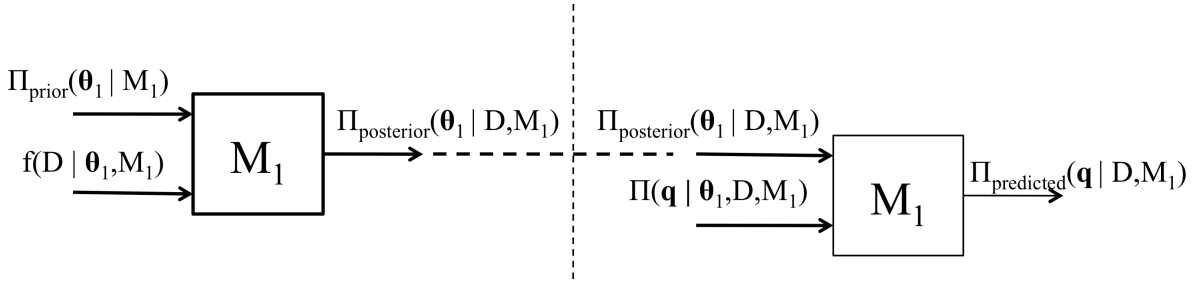


FIG. 1: The calibration-prediction procedure: calibration (solution of an inverse problem) on the left, followed by a prediction (solution of a prediction problem) on the right.

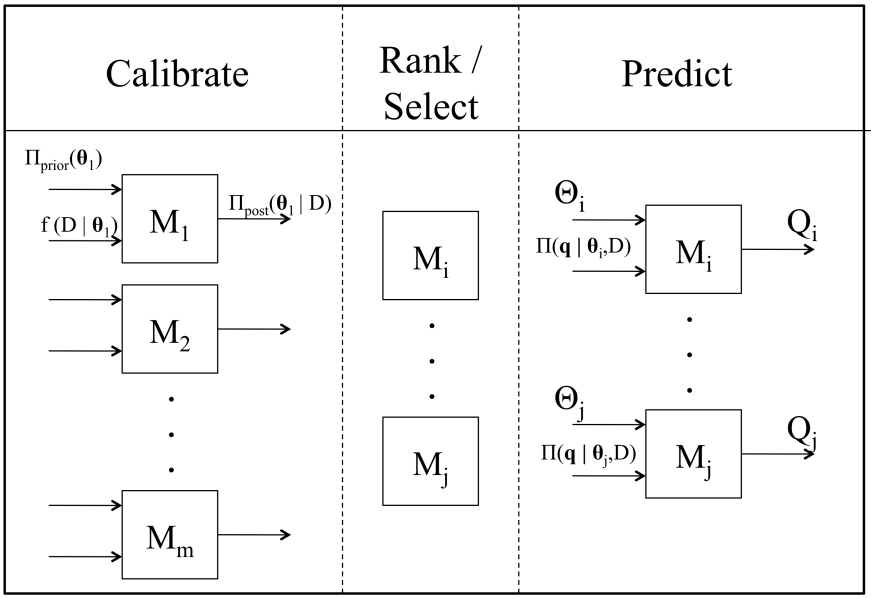


FIG. 2: When many candidate model classes are available, the calibration-prediction procedure of Fig. 1 gets augmented with a step involving the ranking and selection of model classes.

Although the formula (1) for the evaluation of $\pi_{\text{posterior}}(\cdot)$ at a particular parameter vector $\boldsymbol{\theta}$ is straightforward, the calibration-prediction procedure has an intrinsic difficulty, even if the model has "only" tens or hundreds of parameters: How can one use the posterior PDF as an input in a forward problem? One possible solution is to use sampling algorithms. If one is able to sample the posterior distribution correctly, then the samples can be used in a simple Monte Carlo approach; for instance, in order to provide valuable prediction information about the studied phenomenon. However, the challenge here is to "correctly" sample. The posterior PDF might have multiple modes [regions of high probability content (HPC)], and "naive" sampling algorithms may explore some modes fully while completely neglecting other modes. We will come back to this issue in Section 2.

The rest of the paper is organized as follows. After giving motivations in this Introduction, we give an overview of the adaptive multilevel stochastic simulation algorithm (AMSSA, [1, 2]) in Section 2, followed by a detailed analysis of potential load balancing issues in such an algorithm, the resulting proposed parallel AMSSA, and proposed load balancing approaches in Section 3. Section 4 then shows some problems that we used to test load balancing approaches, with corresponding results. We finish the paper with conclusions in Section 5.

2. THE ADAPTIVE MULTILEVEL STOCHASTIC SIMULATION ALGORITHM (AMSSA)

Section 1 sets the motivation for two important computational tasks:

$$\begin{cases} \text{generate samples of the posterior PDF } \pi_{\text{posterior}}\left(\boldsymbol{\theta}|D, M_j\right) \text{ in order to} \\ \text{forward propagate uncertainty and compute QoI random variables,} \end{cases} \tag{2}$$

and

$$\text{compute model evidence } \pi(D|M_j) = \int f(D|\boldsymbol{\theta}, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta}. \tag{3}$$

Regarding computational task (2), it is important to take into account potential multiple modes in the posterior, as highlighted in Section 1. One simple idea is to sample increasingly difficult intermediate distributions, accumulating "knowledge" from one intermediate distribution to the next, until the target posterior distribution is sampled. Assuming one is able to sample the prior, possible intermediate distributions are given by

$$\pi_{\text{int}}^{(\ell)}(\boldsymbol{\theta}|\mathbf{D}) = f(\mathbf{D}|\boldsymbol{\theta}, M_j)^{\tau_\ell} \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j),\ i = 0, 1, \ldots, L, \tag{4}$$

for a given $L > 0$ and a given sequence $0 = \tau_0 < \tau_1 < \ldots < \tau_L = 1$ of exponents. When $\tau_\ell = 0$, the distribution is the prior, and when $\tau_\ell = 1$, the distribution is the posterior. As ℓ increases from $\ell = 0$ until $\ell = L$, the distribution transitions from the initial prior to the final posterior. This algorithm is referred to as a multilevel algorithm where each step ℓ can be thought of as a level.

Regarding computational task (3), a naive approach would be to use Monte Carlo directly, but then the variance of the integral estimator would tend to be too high. But again, the intermediate distributions in Eq. (4) can be of help. First, one clearly has:

$$\begin{aligned} \int f(\boldsymbol{\theta}|D, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta} &= \int f \cdot \pi_{\text{prior}}\, d\boldsymbol{\theta} \\ &= \int f^{(1-\tau_{L-1})}\, f^{(\tau_{L-1}-\tau_{L-2})}\, \ldots\, f^{(\tau_2-\tau_1)}\, f^{\tau_1} \cdot \pi_{\text{prior}}\, d\boldsymbol{\theta}. \end{aligned}$$

Let us assume that the prior PDF is already normalized; i.e., it integrates to 1. If τ_1 is small enough, then Monte Carlo can be efficiently applied to the calculation of

$$c_1 \equiv \int f^{\tau_1} \cdot \pi_{\text{prior}}\, d\boldsymbol{\theta},$$

that is,

$$\int f(\boldsymbol{\theta}|D, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta} = c_1 \int f^{(1-\tau_{L-1})}\, f^{(\tau_{L-1}-\tau_{L-2})}\, \ldots\, f^{(\tau_2-\tau_1)}\, g_1\, d\boldsymbol{\theta},$$

where

$$g_1(\boldsymbol{\theta}) \equiv \frac{f(\boldsymbol{\theta}|D, M_j)^{\tau_1} \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)}{c_1}$$

is a normalized PDF. Next, if $\tau_2 - \tau_1$ is small enough, then Monte Carlo can be efficiently applied to the calculation of

$$c_2 \equiv \int f^{\tau_2 - \tau_1}\, g_1\, d\boldsymbol{\theta},$$

that is,

$$\int f(\boldsymbol{\theta}|D, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta} = c_2 c_1 \int f^{(1-\tau_{L-1})}\, f^{(\tau_{L-1}-\tau_{L-2})}\, \ldots\, f^{(\tau_3-\tau_2)}\, g_2\, d\boldsymbol{\theta},$$

where

$$g_2(\boldsymbol{\theta}) \equiv \frac{f(\boldsymbol{\theta}|D, M_j)^{\tau_2} \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)}{c_2 c_1}$$

is a normalized PDF. By repeating the process for all levels, one has

$$\pi(D|M_j) = \int f(\boldsymbol{\theta}|D, M_j) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}|M_j)\, d\boldsymbol{\theta} = c_L\, c_{L-1}\, \ldots\, c_2\, c_1.$$

For reasons of numerical stability, one rather computes the estimators

$$\tilde{c}_i = \ln c_i,\ i = 1, 2, \ldots, L,$$

that is, one has

$$\ln[\pi(D|M_j)] = \tilde{c_L} + \tilde{c_{L-1}} + \ldots + \tilde{c_2} + \tilde{c_1}.$$

Indeed, the values of c_i can be quite small, and the final evidence can easily get to a value smaller than the smallest positive number representable by the computer. Regarding the statistical accuracy of the final estimator for $\ln[\pi(D|M_j)]$, it is clear that it will be affected by the accuracy of each individual estimator $\tilde{c}_i$. An appropriate rate of increase of the exponent τ_ℓ between levels is a key ingredient on the controlling of such accuracies. This topic is more extensively discussed in [2].

Now let be given, for $\ell = 0, 1, \ldots, L$, (a) the total amount $n_{\text{total}}^{(\ell)} > 0$ of samples to be generated at the ℓth level [this notation is consistent with the notation n_{total} used in Eq. (12) below], and (b) the thresholds $0 < \beta_{\text{min}}^{(\ell)} < \beta_{\text{max}}^{(\ell)} < 1$ on the effective sample size $n_{\text{eff}}^{(\ell)}$ [defined in Eq. (7)] of the ℓth level. The adaptive multilevel algorithm can then be written as:

$$\left\{\begin{array}{l}
01.\ \text{Set } \ell = 0, \tau_\ell = 0; \\
02.\ \text{Sample prior distribution } n_{\text{total}}^{(0)} \text{ times}; \\
03.\ \text{While } \tau_\ell < 1 \text{ do } \{ \\
04.\qquad \text{Begin next level: set } \ell \leftarrow \ell + 1; \\
05.\qquad \text{Compute } \tau_\ell; \\
06.\qquad \text{Select, from previous level, initial positions for Markov chains}; \\
07.\qquad \text{Compute sizes of chains (so that sum of sizes} = n_{\text{total}}^{(\ell)}); \\
08.\qquad \text{Generate chains}; \\
09.\qquad \text{Compute } c_\ell; \\
10.\ \}
\end{array}\right. \tag{5}$$

One of the first versions of the adaptive multilevel algorithm was proposed in [40], where the adaptivity came from the fact that the samples in the previous level are used to construct a global proposal PDF for the generation of Markov chain samples in the current level. An improved version of the algorithm was later proposed in [41] with increased adaptivity coming from the fact that the exponents τ_ℓ do not need to be pre-determined; consequently, neither the total number L of levels. The evidence can be calculated simultaneously. Reference [1] improves the previous versions by modifying steps 05–08. Reference [2] further improves steps 05–07.

In order to briefly explain such steps, let us define for $k = 1, 2, \ldots, n_{\text{total}}^{(\ell)}$:

$$\boldsymbol{\theta}^{(\ell)[k]} = k\text{th sample at the } \ell\text{th level, } \ell = 0, 1, \ldots, L,$$

$$w^{(\ell)[k]} = f^{(\tau_\ell - \tau_{\ell-1})}(D|\boldsymbol{\theta}^{(\ell)[k]}, M_j),\ \ell = 1, \ldots, L,$$

$$\tilde{w}^{(\ell)[k]} = \frac{w^{(\ell)[k]}}{\sum_{s=1}^{n_{\text{total}}^{(\ell)}} w^{(\ell)[s]}},\ \ell = 1, \ldots, L, \tag{6}$$

$$n_{\text{eff}}^{(\ell)} = \frac{1}{\sum_{s=1}^{n_{\text{total}}^{(\ell)}} \left(\tilde{w}^{(\ell)[s]}\right)^2},\ \ell = 1, \ldots, L. \tag{7}$$

Step 05 in Algorithm (5) is accomplished by computing τ_l (e.g., through a bisection method) so that

$$\beta_{\text{min}}^{(\ell)} < \frac{n_{\text{eff}}^{(\ell)}}{n_{\text{total}}^{(\ell)}} < \beta_{\text{max}}^{(\ell)}.$$

Let us now define the discrete distribution

$$P^{(\ell)}(k) = \tilde{w}^{(\ell)[k]},\ k = 1, 2, \ldots, n_{\text{total}}^{(\ell)}. \tag{8}$$

Steps 06 and 07 in Algorithm (5) are accomplished by sampling (8) $n_{\text{total}}^{(\ell)}$ times. The selected indices k determine the samples $\boldsymbol{\theta}^{(\ell)[k]}$ to be used as initial positions, and the number of times an index k is selected determines the size of the chain beginning at $\boldsymbol{\theta}^{(\ell)[k]}$ [the sizes of these chains are denoted by a_i in Eq. (10) below, $i = 1, \ldots, N_t$]. In [2] we explain all these steps in more detail. Since the focus of this current paper is the load balancing of parallel runs of AMSSA, in [2] we also talk in more detail about the important issue of MCMC convergence. Proofs of MCMC convergence assume an infinite amount of samples. In practical MCMC runs, however, one can usually obtain only necessary conditions of convergence, specially in the case of complex nonlinear models with many unknown parameters, when there are practically no clues about the posterior joint PDF and about sufficient conditions of convergence. In the case of AMSSA, it should be noted that the chain of samples obtained per processor at each level is formed by a collection of smaller chains, each beginning at a different initial position. An appropriate rate of increase of the exponent τ_ℓ between levels will give a good combination of initial positions and chain sizes, so that the collection of all samples (of all processors) will represent well the intermediate posterior PDF at the end of each level.

3. THE PARALLEL ADAPTIVE MULTILEVEL STOCHASTIC SIMULATION ALGORITHM (PAMSSA)

Sampling algorithms following Eq. (4) can greatly benefit from parallel computing. At each level ℓ, many computing nodes can be used to sample the parameter space collectively. Beginning with $\ell = 0$, the computing nodes (a) sample $\pi_{\text{int}}^{(\ell)}(\boldsymbol{\theta}|\mathbf{D}, M_j)$; (b) select some of the generated samples ("knowledge") to serve as initial positions of Markov chains for the next distribution $\pi_{\text{int}}^{(\ell+1)}(\boldsymbol{\theta}|\mathbf{D}, M_j)$; and (c) generate the Markov chains for $\pi_{\text{int}}^{(\ell+1)}(\boldsymbol{\theta}|\mathbf{D}, M_j)$. The process (a)–(b)–(c) continues until the final posterior distribution is sampled, as schematized in Fig. 3. The selection process (b), as ℓ increases (see Fig. 8), tends to value samples that are located in the regions of high probability content, which gradually "appear" as τ_ℓ increases. So, as ℓ increases, if the "good" samples selected from the ℓth level to the $(\ell+1)$th level are not redistributed among computing nodes before the Markov chains for the $(\ell+1)$th level are generated, the "lucky" computing nodes (that is, the ones that had, already at the initial levels, samples in the final posterior regions of high probability content) will tend to accumulate increasingly more samples in the next levels. Therefore, as the exponent τ_ℓ increases, care needs to be taken in order to maintain balanced computational load among all computing nodes. This can be achieved by adding one extra step to Algorithm (5), as follows:

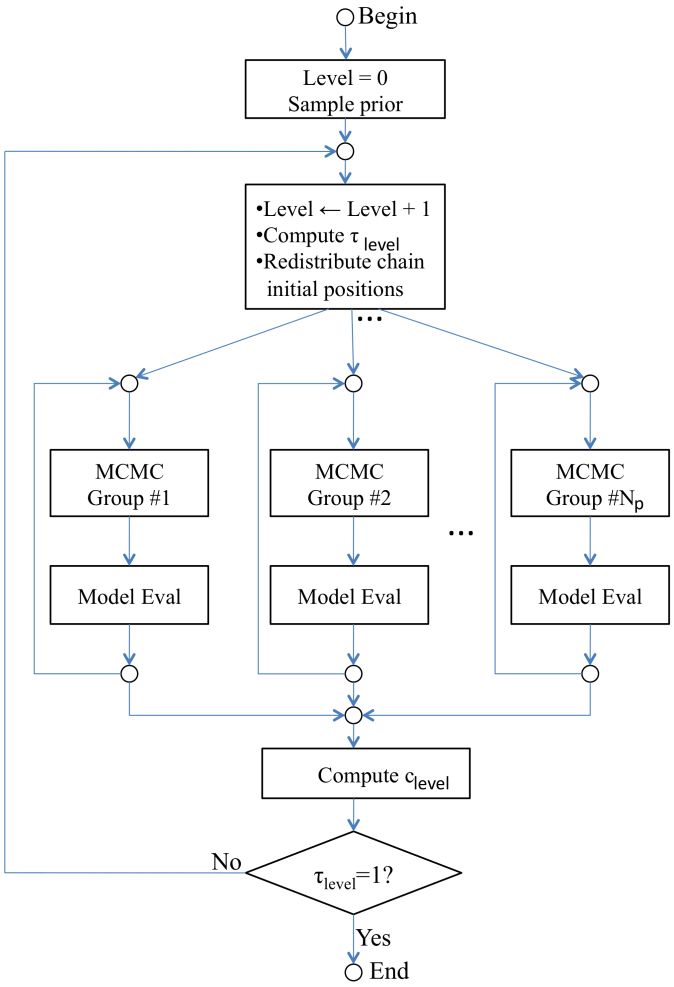


FIG. 3: Usage of N_p parallel chains to try to speed up Algorithms (5) and (9).

$$
\left\{
\begin{array}{l}
01.\ \text{Set } \ell = 0, \tau_\ell = 0; \\
02.\ \text{Sample prior distribution;} \\
03.\ \text{While } \tau_\ell < 1 \text{ do } \{ \\
04.\qquad \text{Begin next level: set } \ell \leftarrow \ell + 1; \\
05.\qquad \text{Compute } \tau_\ell; \\
06.\qquad \text{Select, from previous level, initial positions for Markov chains;} \\
07.\qquad \text{Compute sizes of chains;} \\
08.\qquad \textbf{Redistribute chain initial positions among processors}; \\
09.\qquad \text{Generate chains;} \\
10.\qquad \text{Compute } c_\ell; \\
11.\ \}
\end{array}
\right. \tag{9}
$$

Step 08 in Algorithm (9) is the main subject of this paper. It can be more formally formulated as follows. Assume there are (a) an executable program; (b) $N_t \geqslant 1$ unpartitionable computational tasks; and (c) $N_p \geqslant 1$ processors to execute the tasks, with $N_t > N_p$. The ith task consists of running the executable program a total of $a_i \geqslant 1$ times, $i = 1, \ldots, N_t$. That is,

$$a_i = \text{number of runs of the } i\text{th computational task}, i = 1, \ldots, N_t. \tag{10}$$

In the case of the current paper, for instance, the executable program would be a posterior PDF calculation; a computational task would be the generation of a chain; and a_i would be the size of the chain associated to the ith initial position selected in step 06 of Algorithm (9).

Before we proceed, we note that wherever the word "processor" appears in the paper, it should be understood as an eventual "group of processors," that is, a "processing group." This might occur, for instance, when the model is complicated enough to the point of requiring parallel computing to its evaluation at a chain candidate position. We assume, from now on, that (d) all processors have equal computational capabilities; (e) all calls to the executable program have equal computational demands; and (f) the redistribution of initial positions among processors is computationally negligible compared to one program run. Let us denote

$$n_j = \text{number of runs (to be) handled by the } j\text{th processor}, j = 1, \ldots, N_p, \tag{11}$$

$$n_{\text{total}} = \sum_{i=1}^{N_t} a_i = \sum_{j=1}^{N_p} n_j = \text{total number of runs}, \tag{12}$$

and

$$\overline{n} = \frac{n_{\text{total}}}{N_p} = \text{mean number of runs per processor}. \tag{13}$$

Running step 08 of Algorithm (9) is then equivalent of solving the following problem:

$$\begin{cases} \text{distribute the } N_t \text{ tasks among the } N_p \text{ processors so that each processor gets its total} \\ \text{number } n_j \text{ of program runs, } j = 1, \ldots, N_p, \text{ the closest possible to the mean } \overline{n}. \end{cases} \tag{14}$$

If step 08 of Algorithm (9) is not considered, one can easily get the situation schematized in Fig. 4, where b increases as the levels progress. Another way to see the importance of step 08 is through its effect on the scalability of AMSSA as many chains simultaneously explore the parameter space, as schematized in Fig. 5. If the load is not balanced, it might happen in an extreme case that only one processor ends up concentrating all chain positions after some levels, in which case all the other $N_p - 1$ processors will be idle; that is, a run with N_p processors will end up taking almost the same time as the run with one processor only (assuming, for fair comparison, that the total number of chain positions per level is kept the same between runs with different number of processors).

It should be noted that the complexity of problem (14) comes from the fact that the N_t tasks have potentially very different computational costs $a_1, a_2, \ldots, a_{N_t}$ [due to potentially different values $\tilde{w}^{(\ell)[k]}$, see Eq. (6)]. If all tasks were equally computationally expensive, the problem would simply consist of keeping N_t a multiple of N_p.

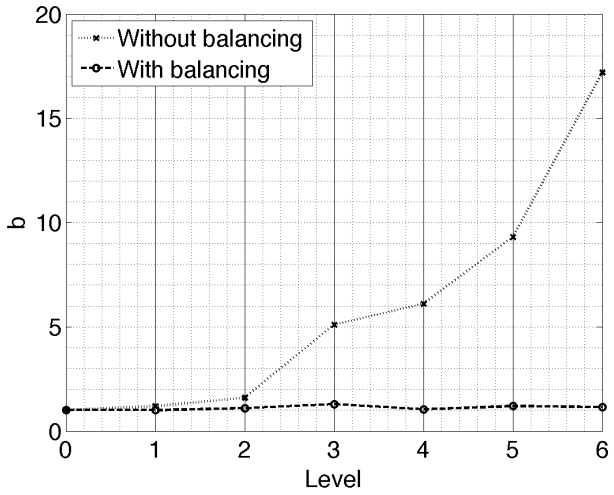


FIG. 4: Schematic evolution of the balancing ratio b, defined in Eq. (16), as the levels of parallel AMSSA Algorithm (9) progress.

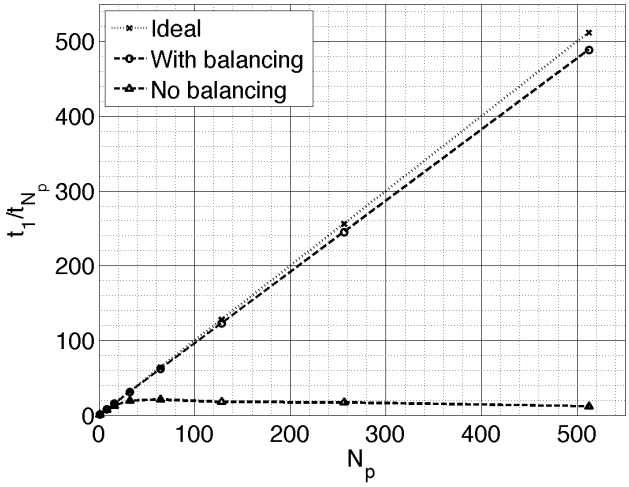


FIG. 5: Schematic overall time scalability of parallel AMSSA Algorithm (9), which depends on whether or not load balancing is used. Assuming that the total number of samples per level is kept the same among runs with different number of processors, t_1 represents the time required for the whole algorithm to run in one processor, while t_{N_p} represents the time required for the whole algorithm to run in N_p processors.

We now describe two possible ways to address problem (14). Section 3.1 briefly describes a binary integer programming (BIP) formulation, while Section 3.2 describes a heuristic approach.

3.1 A Binary Integer Programming Formulation

Problem (14) can be shown to be equivalent to the following BIP problem (see Appendix A for a detailed derivation):

$$\left\{ \begin{array}{l} \displaystyle\min_{\mathbf{r} \in \{0,1\}^{N_t N_p \times 1}} \quad n_1 \;=\; \mathbf{g}_1^T \cdot \mathbf{r}, \\ \\ \text{subject to} \quad \left\{ \begin{array}{rcl} \mathbf{G} \cdot \mathbf{r} & \leqslant & \mathbf{0}_{(N_p-1)\times 1}, \\ \mathbf{M} \cdot \mathbf{r} & = & \mathbf{1}_{N_t \times 1}, \end{array} \right. \end{array} \right. \tag{15}$$

where $\mathbf{G}$ and $\mathbf{M}$ are matrices formed with quantities 0, 1 and a_i. In Eq. (15), (a) the solution vector "$\mathbf{r}$" represents the assignment of the N_p processors to the N_t tasks (a value "1" means that a processor is assigned to a task); (b) the inequality constraints guarantee that all n_j are handled simultaneously (that is, not only the amount n_1 appearing in the objective function); while (c) the equality constraints guarantee that each ith task is assigned to only one processor. It should be noted that this problem is solved at every level. The number N_t of chains and their sizes might vary per level (in the tests of this paper we kept the sum of all N_t chain sizes $a_1, a_2, \ldots, a_{N_t}$ to be constant per level). At each level, problem (15) involves the number N_t of chains, the chain sizes a_i, and the number N_p of processors. Although the computational complexity of Eq. (15) does not depend on the number of unknown parameters, this BIP problem might be very difficult to solve, given its dimension $N_t \times N_p$ and its $N_t + N_p - 1$ constraints. Possible algorithms are the branch and bound method, cutting plane method, and branch and cut method [42]. In the future we intend to use libraries such as GNU GLPK [43] (which uses the branch and cut method) and Zoltan [44] to solve Eq. (15).

3.2 A Heuristic Approach

We now present a heuristic algorithm that we have derived. It was very easy to implement [in contrast to formulation (15)], and it gave satisfactory results in all our test problems of Section 4. Let us denote by

$$b = \frac{\max\limits_{1 \leqslant j \leqslant N_p} n_j}{\min\limits_{1 \leqslant j \leqslant N_p} n_j} \tag{16}$$

the (balancing) ratio between the maximum total computational work and minimum total computational work among all processors. Problem (14) can also be formulated as

$$\text{Distribute the } N_t \text{ tasks among the } N_p \text{ processors so that } b \text{ gets the closest possible to } 1. \tag{17}$$

The step 08 of Algorithm (9) begins with an original assignment of computational tasks (i.e., chain initial positions and corresponding chain sizes) to each processor. Indeed, the objective of step 08 is to reassign such computational tasks among the processors in order to improve their computational balance. Let us denote by $N_t^{(j)}$ the number of computational tasks assigned to the jth processor, and let $a_k^{(j)}$, $k = 1, 2, \ldots, N_t^{(j)}$, denote the number of program runs (i.e., chain sizes) of each of these computational tasks. Let us denote by

$$S = \{a_1, a_2, \ldots, a_{N_t}\}$$

the set of all N_t computational tasks that are (re)assigned, and by

$$\tilde{S} = \{a_1^{(1)}, a_2^{(1)}, \ldots, a_{N_t^{(1)}}^{(1)}, \ldots, a_1^{(N_p)}, a_2^{(N_p)}, \ldots, a_{N_t^{(N_p)}}^{(N_p)}\}$$

the set of all computational tasks assigned per processor, for all processors. It should be clear that there is a bijective correspondence between S and $\tilde{S}$. Also, the total number n_j of program runs assigned for the jth processor [see Eq. (11)] is

$$n_j = \sum_{k=1}^{N_t^{(j)}} a_k^{(j)}, \quad j = 1, \ldots, N_p.$$

We propose the following simple heuristic algorithm:

$$\left\{\begin{array}{l}
01.\ \text{Find } j_{\text{min}} = \displaystyle\arg\min_{1 \leqslant j \leqslant N_p} n_j; \\
02.\ \text{Find } j_{\text{max}} = \displaystyle\arg\max_{1 \leqslant j \leqslant N_p} n_j; \\
03.\ \text{Set } b = \dfrac{n_{j_{\text{max}}}}{n_{j_{\text{min}}}}; \\
04.\ \text{Set iteration} = 0; \\
05.\ \text{While } (b \neq 1) \text{ and } (\text{iteration} < n_{\text{total}}) \text{ do } \{ \\
06.\quad \text{Find } k_{\text{min}} = \displaystyle\arg\min_{1 \leqslant k \leqslant N_t^{(j_{\text{max}})}} a_k^{(j_{\text{max}})}; \\
07.\quad \textbf{Reassign } a_{k_{\text{min}}}^{(j_{\text{max}})} \text{ from the } j_{\text{max}}\text{-th to the } j_{\text{min}}\text{-th processor;} \\
08.\quad \text{Find } \tilde{j}_{\text{min}} = \displaystyle\arg\min_{1 \leqslant j \leqslant N_p} n_j; \\
09.\quad \text{Find } \tilde{j}_{\text{max}} = \displaystyle\arg\max_{1 \leqslant j \leqslant N_p} n_j; \\
10.\quad \text{Compute } \tilde{b} = \dfrac{n_{\tilde{j}_{\text{max}}}}{n_{\tilde{j}_{\text{min}}}}; \\
11.\quad \textbf{If } \tilde{\mathrm{b}} > \mathrm{b} \textbf{ then exit 'While' loop;} \\
12.\quad \text{Prepare for next iteration: set } j_{\text{min}} = \tilde{j}_{\text{min}},\ j_{\text{max}} = \tilde{j}_{\text{max}},\ b = \tilde{b}; \\
13.\quad \text{Set iteration} \leftarrow \text{iteration} + 1; \\
14.\ \}
\end{array}\right. \tag{18}$$

The idea behind Algorithm (18) is very simple: gradually reassign the smallest computational task from the processor with the biggest n_j (total computational work) to the processor with smallest total computational work, until

ratio b gets to value 1, or b increases, or too many iterations happen. The intuition behind setting the maximum number of iterations to n_{total} is to give a reasonable average chance for all N_t computational tasks to be used in the reassignment step 07 of Algorithm (18). Indeed, $N_t \leqslant n_{\text{total}}$, and such an upper limit on the amount of iterations has performed quite well in the examples of this paper (see Section 4), in the sense that the heuristic Algorithm (18) has guaranteed balancing ratio $b \leqslant 2$.

4. TEST PROBLEMS AND RESULTS

In this paper we study the load balancing issue in the context of three different problems, described below in Subsections 4.1–4.3. We report numerical experiments with the PAMSSA Algorithm (9) balanced with heuristic algorithm (18). The algorithms were coded on a C++/MPI environment using the QUESO library [39]. We performed experiments with three different amounts N_p of processors: one, eight, and 64. In all experiments we have used a total of 8,192 MCMC samples per level. When eight (or 64) processors were used, each processor generated approximately 1,024 (or 128) samples per level, although the total number of samples per level was always 8,192, as in the case of experiments with one processor only. We say "approximately" because the ideal balancing amount (1,024 or 128) of samples per processor is guaranteed only at the beginning of level 0. For all subsequent levels, the amount of samples per processor will get close to the ideal value as long as the balancing ratio b of Eq. (16) gets close to unity. It should be noted also that all chains in all processors are run without a burn-in phase.

4.1 A 1D Problem

Let us define $D = [-250.0,\ 250.0]$, and the three distributions $\pi_{\text{prior}} : D \to \mathbb{R}_+$, $f_1 : \mathbb{R} \to \mathbb{R}_+$ and $f_2 : \mathbb{R} \to \mathbb{R}_+$ by

$$\pi_{\text{prior}}(\boldsymbol{\theta}) = \frac{1}{|D|} \ = \ \frac{1}{500}, \quad \forall\ \boldsymbol{\theta} \in D,$$

$$f_1(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{(1/2)}\sqrt{|\mathbf{V}_1|}}\, e^{-(1/2)(\boldsymbol{\theta}-\boldsymbol{\mu}_1)^T\ \mathbf{V}_1^{-1}\ (\boldsymbol{\theta}-\boldsymbol{\mu}_1)}, \quad \forall\ \boldsymbol{\theta} \in \mathbb{R}, \tag{19}$$

and

$$f_2(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{(1/2)}\sqrt{|\mathbf{V}_2|}}\, e^{-(1/2)(\boldsymbol{\theta}-\boldsymbol{\mu}_2)^T\ \mathbf{V}_2^{-1}\ (\boldsymbol{\theta}-\boldsymbol{\mu}_2)}, \quad \forall\ \boldsymbol{\theta} \in \mathbb{R}, \tag{20}$$

where

$$\boldsymbol{\mu}_1 = 10, \quad \mathbf{V}_1 = 1^2,$$

and

$$\boldsymbol{\mu}_2 = 100, \quad \mathbf{V}_2 = 5^2.$$

In this example we are interested in sampling the posterior PDF given by

$$\pi_{\text{posterior}}(\boldsymbol{\theta}) \propto \left[\frac{1}{2} f_1(\boldsymbol{\theta}) + \frac{1}{2} f_2(\boldsymbol{\theta})\right] \cdot \pi_{\text{prior}}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) \cdot \pi_{\text{prior}}(\boldsymbol{\theta}).$$

Figure 6 shows the likelihood function $f(\boldsymbol{\theta})$. Figure 7 presents the results for one processor only. This 1D problem is very simple (low dimension), requiring just four levels to achieve exponent $\tau = 1$, as shown in Fig. 7(a). Figure 7(b) is shown just to highlight that runs with one processor have no balancing problem, obviously.

Figure 8 shows the intermediary likelihood functions for the four exponents of Fig. 7(a). The "sharp" corner that appears in Fig. 8 for the curve for $\tau = 0.00586$ is a result of the use of the two Gaussian distributions (19) and (20). Indeed, such a small value of τ exposes two "Gaussian" shapes: one around $\theta = 10$ and the other around $\theta = 100$. Also, although the PDF for the first exponent $\tau = 0.00586$ is not unimodal, the proximity of the two modes makes it feasible for standard MCMC algorithms to appropriately sample the distribution, in contrast to the difficulty of these same standard MCMC algorithms to handle the bimodal distribution of Fig. 6 directly.

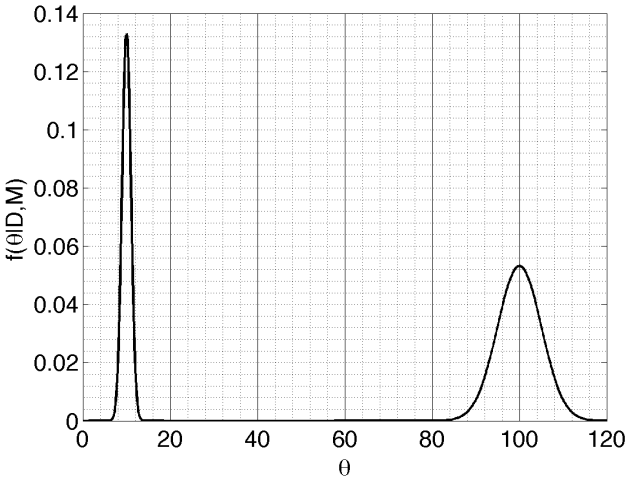
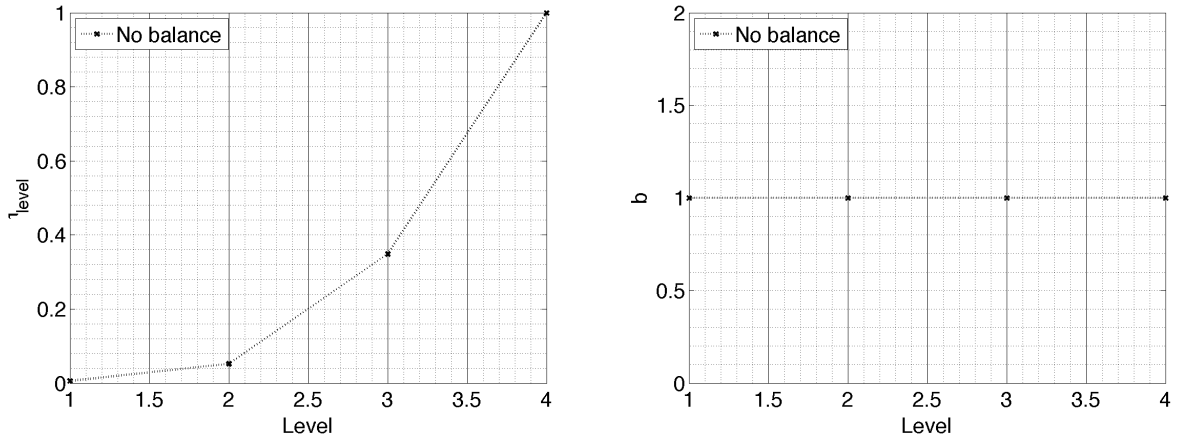


FIG. 6: Likelihood function of our 1D test problem.



(a) (b)

FIG. 7: Behavior of Algorithm (5) on the 1D problem with $N_p = 1$ processor: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16).

Figure 9 presents the results obtained with eight processors. Figure 9(a) shows the evolution of exponents, per level, for two cases: Algorithm (9) without balancing, and Algorithm (9) with heuristic balancing (18). The cases are similar to each other and to Fig. 9(a) as well. In Fig. 9(b) we see some unbalancing happening when the reassignment step [step 08 in Algorithm (9)] is not applied. However, this unbalancing is still minor, given the combination of a simple problem (just 1D) and a small number of processors.

Figure 10 presents the results that we got with 64 processors. Figure 10(a) is similar to the cases with one and eight processors. Figure 10(b), however, now shows a more significant increase of the balancing ratio b [see Eq. (16)], although the small number of levels contributes to such a ratio not increas much.

4.2 Sum of 10D Gaussian Distributions

Let us define $N = 10$, $D = [0.5, 3.5]^N = [0.5, 3.5] \times \ldots \times [0.5, 3.5]$, and the three distributions $\pi_{\text{prior}} : D \to \mathbb{R}_+$, $f_1 : \mathbb{R}^N \to \mathbb{R}_+$ and $f_2 : \mathbb{R}^N \to \mathbb{R}_+$ by

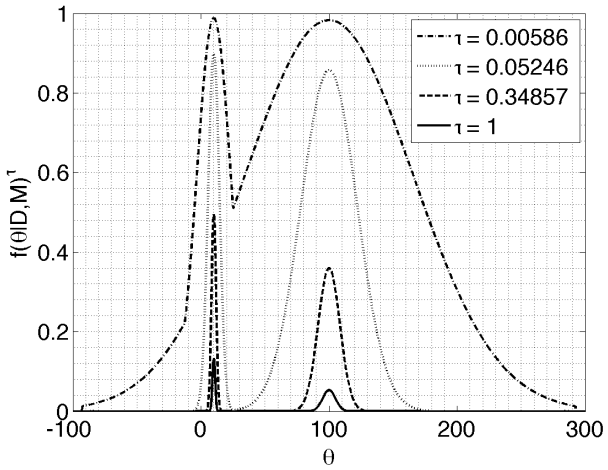
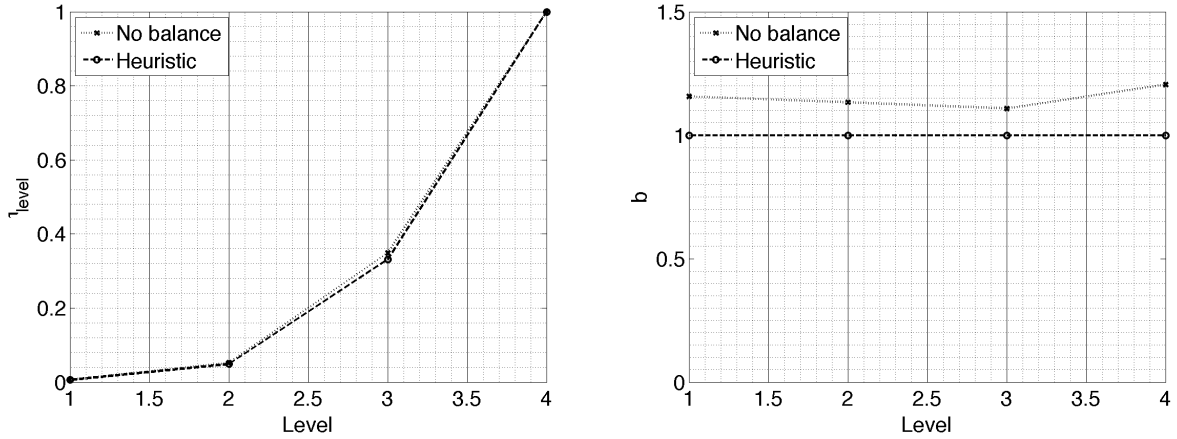


FIG. 8: Intermediary likelihood functions $f(\boldsymbol{\theta}|D,M)^{\tau_\ell}$, obtained during the solution of the 1D test problem, where τ_ℓ is the exponent computed at the ℓ-th level of Algorithm (9), $\ell \geqslant 1$. The four exponent values are the same ones appearing in Fig. 7(a). The continuous curve (exponent $\tau = 1$) is the same curve as in Fig. 6.



(a) (b)

FIG. 9: Behavior of Algorithm (9), without and with heuristic balancing Eq. (18), on the 1D problem with $N_p = 8$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16).

$$\pi_{\text{prior}}(\boldsymbol{\theta}) = \frac{1}{|D|} = \frac{1}{3^{10}}, \quad \forall\, \boldsymbol{\theta} \in D,$$

$$f_1(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{N/2}\sqrt{|\mathbf{V}_1|}}\, e^{-(1/2)(\boldsymbol{\theta}-\boldsymbol{\mu}_1)^T\, \mathbf{V}_1^{-1}\, (\boldsymbol{\theta}-\boldsymbol{\mu}_1)}, \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^N,$$

and

$$f_2(\boldsymbol{\theta}) = \frac{1}{(2\pi)^{N/2}\sqrt{|\mathbf{V}_2|}}\, e^{-(1/2)(\boldsymbol{\theta}-\boldsymbol{\mu}_2)^T\, \mathbf{V}_2^{-1}\, (\boldsymbol{\theta}-\boldsymbol{\mu}_2)}, \quad \forall\, \boldsymbol{\theta} \in \mathbb{R}^N,$$

where

$$\boldsymbol{\mu}_1 = (1,\ldots,1),\ \mathbf{V}_{1,ij} = (0.1)^2 \cdot \exp[(i-j)^2/16],$$

and

$$\boldsymbol{\mu}_2 = (3,\ldots,3),\ \mathbf{V}_{2,ij} = (0.1)^2 \cdot \exp[(i-j)^2/16].$$

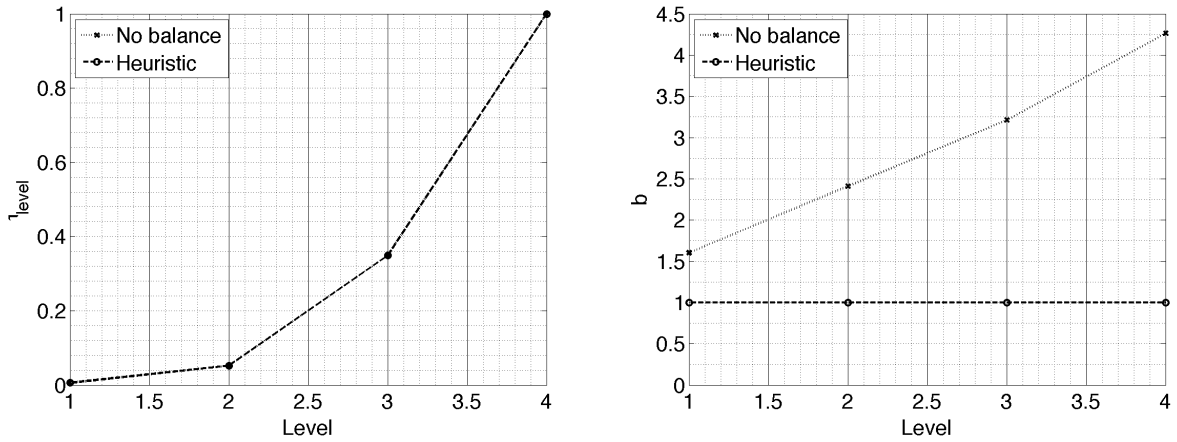


FIG. 10: Behavior of Algorithm (9), without and with heuristic balancing Eq. (18), on the 1D problem with $N_p = 64$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16).

In this example we are interested in the posterior PDF given by

$$\pi_{\text{posterior}}(\boldsymbol{\theta}) \propto \left[\frac{1}{2} f_1(\boldsymbol{\theta}) + \frac{1}{2} f_2(\boldsymbol{\theta})\right] \cdot \pi_{\text{prior}}(\boldsymbol{\theta}).$$

Figures 11 and 12 present the results obtained with eight and 64 processors, respectively. Figures 11(a) and 12(a) are similar to each other, but they present many more levels than in the 1D problem case. Also, the first 15 levels ended up resulting in very small exponents, which motivated us to present them in the logarithmic scale.

Regarding the behavior of the balancing ratio b, we observed that at least one processor easily got chains of size 0 (zero) already at the initial levels if load balancing was not pursued, resulting in $b = +\infty$. We, therefore, decided to present in Figs. 11(b) and 12(b), the tendency of b to increase at every level during the run of balanced Algorithm (9). That is, we present the value of b before and after step 08 of Algorithm (9). The strong tendency to unbalancing is clear in both Figs. 11(b) and 12(b), due to the problem complexity (10 dimensions, with two isolated modes). The

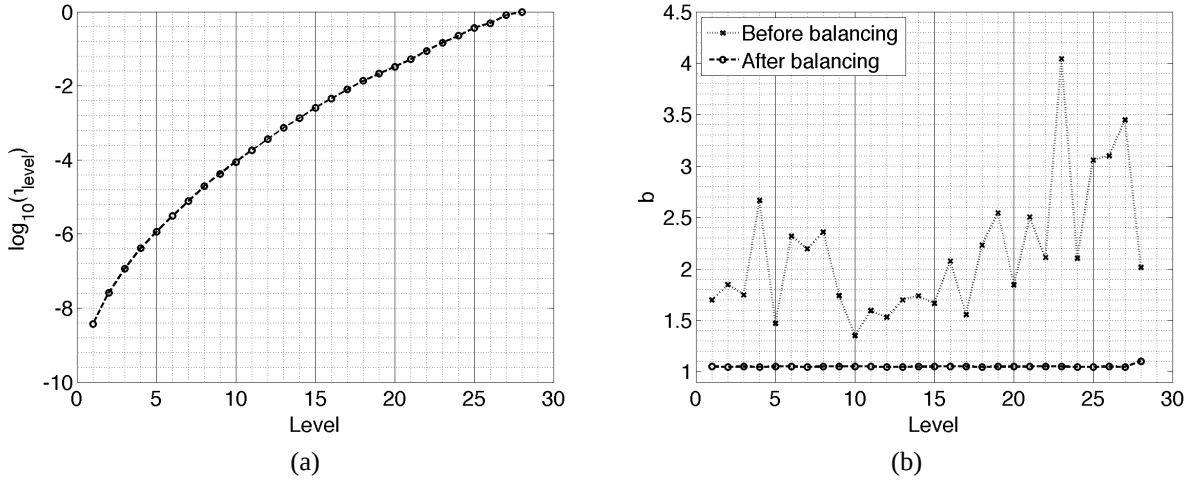
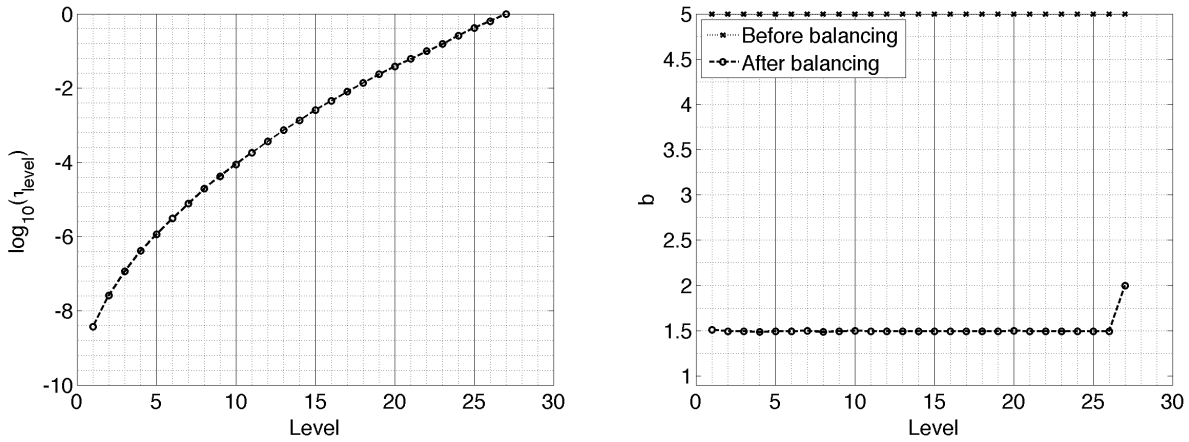


FIG. 11: Behavior of the Algorithm (9), with heuristic balancing Eq. (18), on the 10D problem with $N_p = 8$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16).



(a) (b)

FIG. 12: Behavior of the Algorithm (9), with heuristic balancing Eq. (18), on the 10D problem with $N_p = 64$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16). The value "5" in (b) represents $+\infty$, meaning that at least one processor was assigned zero chain positions before reassignment step 08 in Algorithm (9).

tendency is stronger for the case of more processors, since the bigger the number of processors, the less samples each processor has at every level when one fixes the total number of samples per level (fixed to 8,192 in our tests, (as reported in the beginning of Section 4). Also, the heuristic algorithm performs very well in the case of eight processors [Fig. 11(a)], always bringing b to values close to the ideal value of 1. The case of 64 processors [Fig. 11(b)] is more challenging, where the heuristic algorithm guarantees $1.5 \leqslant b \leqslant 2$, and even so, at every step at least one processor got zero samples to handle before reassignment step 08, causing $b = +\infty$ [represented by the value "5" in Fig. 11(b)]. Although not equal to the ideal value of 1, these b values between 1.5 and 2 are still much better than the case where (potentially many) processors get completely idle, with no samples to handle.

4.3 A Hysteretic Model Class

In this example we consider the nonlinear seismic response of a four-story building. We model such a response with a shear building model with some linear viscous damping and hysteretic bilinear interstory restoring forces. More specifically, let $t \geqslant 0$ denote time, let $a_g(t)$ be a given total acceleration at the base (Fig. 13), and for the ith floor [degree of freedom (dof)], $1 \leqslant i \leqslant N_o \equiv 4$, let us denote:

$$m_i = \text{mass},$$

$$q_i(t) = \text{horizontal displacement},$$

$$F_i(t) = \text{hysteretic restoring force, illustrated in Fig. 14.}$$

We also define the mass matrix

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 \\ 0 & 0 & m_3 & 0 \\ 0 & 0 & 0 & m_4 \end{bmatrix},$$

the stiffness matrix

$$\mathbf{K} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 & 0 \\ -k_2 & k_2 + k_3 & -k_3 & 0 \\ 0 & -k_3 & k_3 + k_4 & -k_4 \\ 0 & 0 & -k_4 & k_4 \end{bmatrix},$$

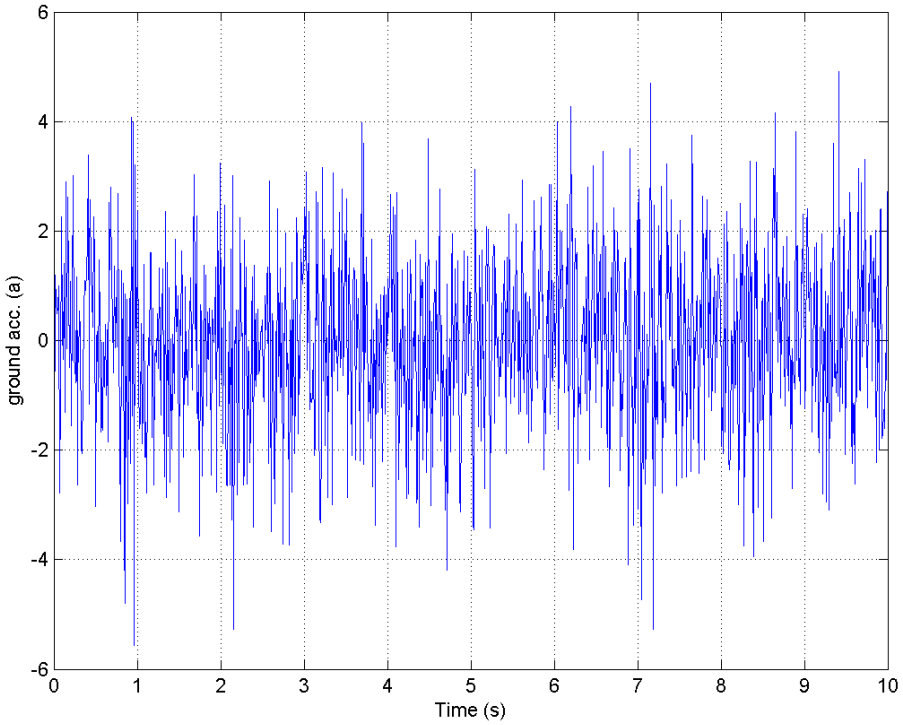


FIG. 13: Horizontal base acceleration (input data) used in our hysteretic test problem.

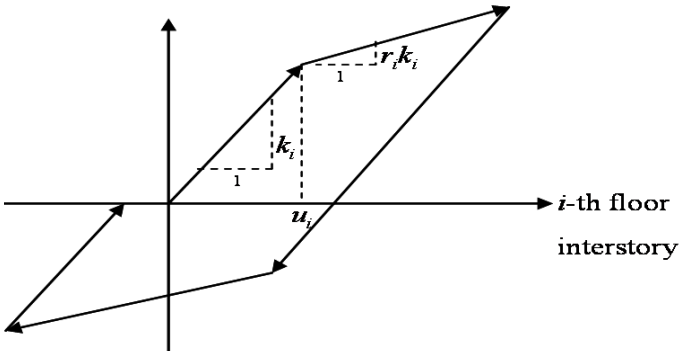


FIG. 14: Illustration of the hysteretic restoring force [see Eq. (21)] used in our hysteretic test problem. The terms r_i, k_i, and u_i denote model parameters.

and the Rayleigh damping matrix

$$\mathbf{C} = \rho\mathbf{M} + \gamma\mathbf{K},$$

for given positive scalar parameters ρ and γ. We model the response $\mathbf{q}(t) \equiv [q_1(t), q_2(t), q_3(t), q_4(t)]$ as satisfying the equation of motion

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{C}\dot{\mathbf{q}}(t) + \mathbf{F}(t) = -\mathbf{M} \cdot \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{4\times 1} \cdot a_g(t), \tag{21}$$

where $\mathbf{F}(t) \equiv [F_1(t), F_2(t), F_3(t), F_4(t)]$. In this model, the hysteretic restoring force $F(t)$ depends on the whole time history $[\mathbf{q}(t), \dot{\mathbf{q}}(t)]$ of responses from the initial instant until time t.

The (noisy) measured data $\mathbf{y} = (y_1, y_2, y_3, y_4)$ available for model calibration consists of 10 s of accelerometer data at each floor (Fig. 15), with a sample interval Δt of 0.01 s. It was obtained by adding white noise to the output simulation of the hysteretic model with the following input values:

$$m_1 = m_2 = m_3 = m_4 = 2 \times 10^4 \text{ kg},$$

$$k_1 = 2.2 \times 10^7 \text{ Nm}^{-1}, \quad k_2 = 2.0 \times 10^7 \text{ Nm}^{-1}, \quad k_3 = 1.7 \times 10^7 \text{ Nm}^{-1}, \quad k_4 = 1.45 \times 10^7 \text{ Nm}^{-1},$$

$$r_1 = r_2 = r_3 = r_4 = 0.1,$$

$$u_1 = u_2 = 8 \times 10^{-3} \text{ m}, \quad u_3 = u_4 = 7 \times 10^{-3} \text{ m},$$

$$\rho = 7.959 \times 10^{-1},$$

and

$$\gamma = 2.5 \times 10^{-3}.$$

These input values were chosen deliberately so that the excitation a_g did not cause some of the upper floors to enter the nonlinear regime; that is, so that our test inverse problem did not become globally identifiable. In order to diminish the computational cost of this test problem, while still being able to show the benefit of load balancing, we used just 250 time steps; i.e., the data were available at instants $t_n = (n-1) \times 0.04$, $1 \leqslant n \leqslant N_T \equiv 251$. Also, we assumed an additive noise in the measurements; i.e.,

$$y_i = q_i + \varepsilon_i, \ 1 \leqslant i \leqslant N_o.$$

We consider a total of 15 unknown parameters $\theta = (\theta_1, \ldots, \theta_{15})$. We model the variables ε_i as independently and identically distributed Gaussian variables with mean zero and some unknown prediction-error variance σ^2. The

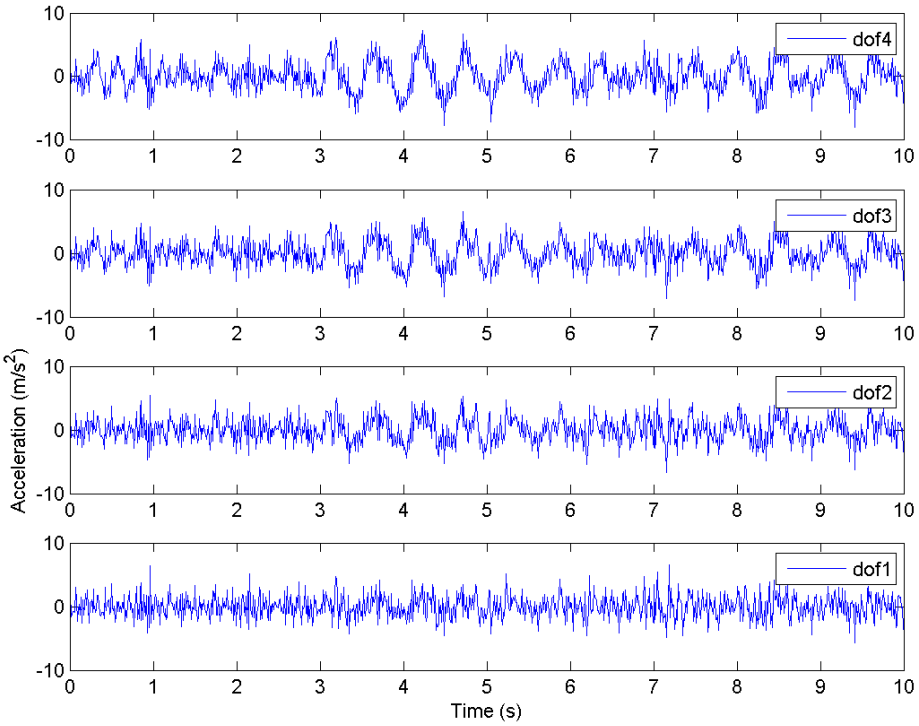


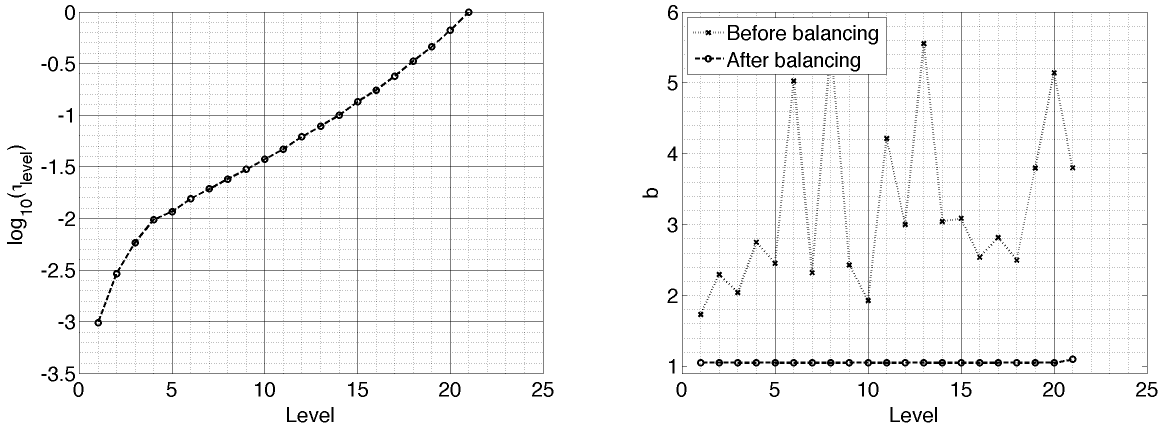
FIG. 15: Horizontal acceleration of each of the four floors (measured data aimed for calibration) used in our hysteretic test problem.

variance σ^2 is assumed to be the same for all $N_o = 4$ floors. The other 14 parameters are related to the four triples (k_i, r_i, u_i), $1 \leqslant i \leqslant N_o$ (see Fig. 14), to ρ, and to γ. The likelihood function for this model class M is given by

$$f(\mathbf{y}|\boldsymbol{\theta}, M) = \frac{1}{(2\pi\sigma^2)^{N_o N_T/2}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N_o}\sum_{n=1}^{N_T}[y_i(t_n) - \ddot{q}_i(t_n;\theta_2,...,\theta_{15})]^2\right).$$

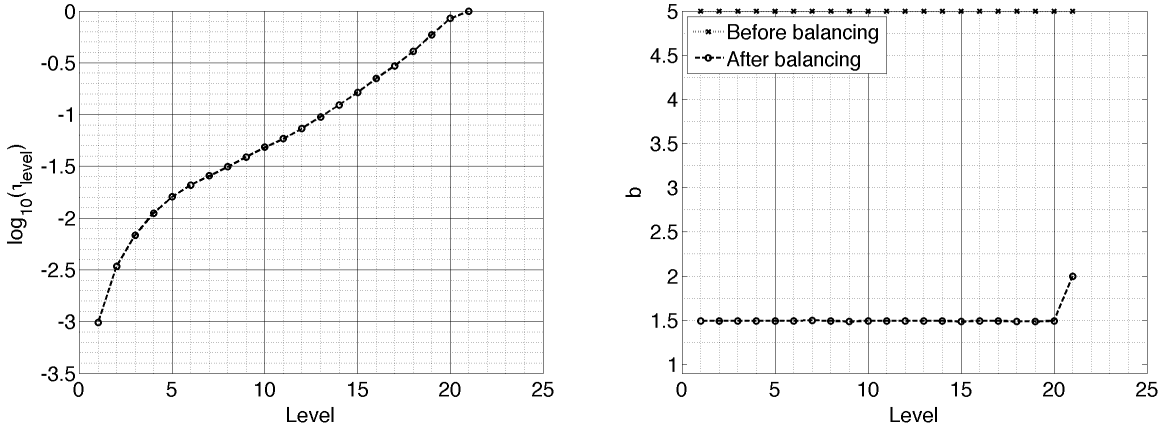
We use an inverse gamma prior for σ, and a 14-dimensional Gaussian prior for $\theta_2, ..., \theta_{15}$ with zero mean and diagonal covariance matrix equal to a scaled identity matrix.

Figures 16 and 17 present the results obtained with eight and 64 processors, respectively. All the comments of Section 4.2 apply here as well. The only difference now is that the tendency for unbalancing is stronger in Fig. 16(b) than in Fig. 11(b).



(a) (b)

FIG. 16: Behavior of Algorithm (9), with heuristic balancing Eq. (18), on the hysteretic problem with $N_p = 8$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16).



(a) (b)

FIG. 17: Behavior of Algorithm (9), with heuristic balancing Eq. (18), on the hysteretic problem with $N_p = 64$ processors: τ_{level} is the exponent at each level, while b is the balancing ratio defined in Eq. (16). The value "5" in (b) represents $+\infty$, meaning that at least one processor was assigned zero chain positions before reassignment step 08 in Algorithm (9).

5. CONCLUSIONS

As the application of Bayesian analysis gradually evolves to more realistic mathematical models, parallel computing will become a natural component of the analysis. In this paper, we have analyzed the important problem of balancing the load of parallel processors when applying the AMSSA on parallel HPC environments. We have discussed two ways to balance the computational load: by solving a BIP problem, and by using a heuristic algorithm. We have performed experiments with the heuristic approach using combinations of three different posterior PDFs (a sum of 1D Gaussians, a sum of 10D Gaussians, and a PDF related to a stochastic hysteretic model class) and three different number of processors (one, eight, and 64). The heuristic algorithm has performed very well in all cases.

Parallel MCMC algorithms involve interdisciplinary challenges in applied mathematics, computer science, and computational science; e.g., robust statistical exploration of parameter spaces, load balancing, algorithm resilience with respect to computational node failures, software library design, to name a few. We intend to further investigate the BIP approach and other heuristic algorithms in our future work, especially in the context of heterogeneous parallel HPC platforms.

ACKNOWLEDGMENTS

This research was supported by the Department of Energy (National Nuclear Security Administration), under the Predictive Science Academic Alliance Program (PSAAP), Award Number DE-FC52-08NA28615, and by the King Abdullah University of Science and Technology (KAUST), under the Academic Excellence Alliance (AEA) program. E.E.P. was also partially supported by Sandia National Laboratories under Contract Numbers 1017123 and 1086312. All calculations were performed on the RANGER high-performance computer at the Texas Advanced Computing Center (TACC) [45]. The authors are also thankful to two anonymous referees, whose comments and questions helped them to improve their paper.

REFERENCES

1. Cheung, S. H. and Beck, J. L., New Bayesian updating methodology for model validation and robust predictions of a target system based on hierarchical subsystem tests, *Comput. Methods Appl. Mech. Eng.*, accepted, 2012.
2. Cheung, S. H. and Prudencio, E. E., Adaptive multi-level stochastic simulation algorithm for Bayesian model class updating, assessment, averaging and validation, in preparation, 2012.
3. Calvetti, D. and Somersalo, E., *Introduction to Bayesian Scientific Computing*, vol. 2, Springer, New York, 2007.
4. Haario, H., Laine, M., Mira, A., and Saksman, E., DRAM: Efficient adaptive MCMC, *Stat. Comput.*, 16:339–354, 2006.
5. Kaipio, J. and Somersalo, E., *Statistical and Computational Inverse Problems*, vol. 160, Springer, New York, 2005.
6. Biros, G. and Ghattas, O., Parallel Lagrange–Newton–Krylov–Schur methods for PDE–constrained optimization, Part I: The Krylov–Schur solver, *SIAM J. Sci. Comput.*, 27(2):687–713, 2005.
7. Biros, G. and Ghattas, O., Parallel Lagrange–Newton–Krylov–Schur methods for PDE–constrained optimization. Part II: The Lagrange–Newton solver and its application to optimal control of steady viscous flows, *SIAM J. Sci. Comput.*, 27(2):714–739, 2005.
8. Prudencio, E. E., Byrd, R., and Cai, X. C., Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems, *SIAM J. Sci. Comput.*, 27:1305–1328, 2006.
9. Prudencio, E. E. and Cai, X. C., Robust multilevel restricted Schwarz preconditioners and applications, *Domain Decomposition Methods in Science and Engineering XVI*, Widlund, O. B. and Keyes, D. E., eds., Springer, New York, pp. 155–162, 2006.
10. Prudencio, E. E. and Cai, X. C., Parallel multilevel restricted Schwarz preconditioners with pollution removing for PDE-constrained optimization, *SIAM J. Sci. Comp.*, 29:964–985, 2007.
11. Yang, H., Prudencio, E. E., and Cai, X. C., Fully implicit Lagrange-Newton-Krylov-Schwarz algorithms for boundary control of unsteady incompressible flows, *Int. J. Num. Methods Eng.*, accepted, 2012.
12. Ghanem, R. and Spanos, P. D., *Stochastic Finite Elements: A Spectral Approach*, Springer, New York, 1991.

13. Xiu, D. and Karniadakis, G., The Wiener-Askey polynomial chaos for stochastic differential equations, *SIAM J. Sci. Comput.*, 24(2):619–644, OCT 31 2002.
14. Debusschere, B., Najm, H., Pébay, P., Knio, O., Ghanem, R., and Le Maître, O., Numerical challenges in the use of polynomial chaos representations for stochastic processes, *SIAM J. Sci. Comput.*, 26(2):698–719, 2004.
15. Babuška, I., Tempone, R., and Zouraris, G., Solving elliptic boundary value problems with uncertain coefficients by the finite element method: the stochastic formulation, *Comput. Methods Appl. Mech. Eng.*, 194:1251–1294, 2005.
16. Babuska, I., Nobile, F., and Tempone, R., A stochastic collocation method for elliptic partial differential equations with random input data, *SIAM J. Numer. Anal.*, 45(3):1005–1034, 2007.
17. Ma, X. and Zabaras, N., An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, *J. Comput. Phys.*, 228(8):3084–3113, 2009.
18. Le Maître, O. and Knio, O., *Spectral Methods for Uncertainty Quantification*, Springer, New York, 2010.
19. Cox, R. T., *The Algebra of Probable Inference*, Johns Hopkins University Press, Baltimore, MD, 1961.
20. Jaynes, E. T., *Probability Theory: The Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.
21. Beck, J. L. and Cheung, S. H., Probability logic, model uncertainty and robust predictive system analysis, *Proc. of International Conference on Structural Safety and Reliability*, Osaka, Japan, September 2009, 2009.
22. Kolmogorov, A. N., *Grundbegriffe der Wahrscheinlichkeitsrechnung*, Springer, Berlin, 1933.
23. Kolmogorov, A. N., *Foundations of the Theory of Probability*, Chelsea Publishing Company, New York, 1956.
24. Beck, J. L. and Katafygiotis, L. S., Updating of a model and its uncertainties utilizing dynamic test data, *Proc. of 1st International Conference on Computational Stochastic Mechanics*, 125–136, 1991.
25. Beck, J. L. and Katafygiotis, L. S., Updating models and their uncertainties, I: Bayesian statistical framework, *ASCE J. Eng. Mech.*, 124:455–461, 1998.
26. Cheung, S. H. and Beck, J. L., Calculation of posterior probabilities for Bayesian model class assessment and averaging from posterior samples based on dynamic system data, *Comput. Aided Civ. Infrastruct. Eng.*, 25(5):304–321, 2010.
27. Cheung, S. H. and Beck, J. L., Comparison of different model classes for Bayesian updating and robust predictions using stochastic state-space system models., *Proc. of 10th International Conference on Structural Safety and Reliability, (ICOSSAR09)*, Osaka, Japan, Sept. 13–17, 2009.
28. Beck, J. L. and Yuen, K. V., Model selection using response measurements: A Bayesian probabilistic approach, *ASCE J. Eng. Mech.*, 130:192–203, 2004.
29. Ching, J., Muto, M., and Beck, J. L., Bayesian linear structural model updating using Gibbs sampler with modal data, *Proc. of International Conference on Structural Safety and Reliability*, Rome, Italy, June, 2005, 2005.
30. Muto, M. and Beck, J. L., Bayesian updating of hysteretic structural models using stochastic simulation, *J. Vib. Control*, 14:7–34, 2008.
31. Kullback, S. and Leibler, R., On information and sufficiency, *Ann. Math. Stat.*, 22(1):79–86, 1951.
32. Gull, S. F., Bayesian inductive inference and maximum entropy, *Maximum Entropy and Bayesian Methods*, Erickson, G. J. and Smith, C. R., eds., Kluwer Academic, Dordrecht, The Netherlands, pp. 53–74, 1998.
33. Cheung, S. H., Oliver, T. A., Prudencio, E. E., Prudhomme, S., and Moser, R. D., Bayesian uncertainty analysis with applications to turbulence modeling, *Reliab. Eng. Syst. Safety*, 96(9):1137–1149, 2011.
34. Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T., Bayesian model averaging: A tutorial (with discussion), *Stat. Sci.*, 14:382–417, 1999.
35. Raftery, A. E., Madigan, D., and Hoeting, J. A., Bayesian model averaging for linear regression models, *J. Am. Stat. Assoc.*, 92:179–191, 1997.
36. Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., PETSc Web page, http://www.mcs.anl.gov/petsc, 2011.
37. Adams, B. M., Dalbey, K. R., Eldred, M. S., Gay, D. M., Swiler, L. P., Bohnhoff, W. J., Eddy, J. P., and Hough, P. D., DAKOTA, A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis, Sandia Technical Report, http://www.cs.sandia.gov/DAKOTA/, 1994–2011.
38. Heroux, M., Bartlett, R., Hoekstra, V. H. R., Hu, J., Kolda, T., Lehoucq, R., Long, K., Pawlowski, R., Phipps, E., Salinger, A.,

Thornquist, H., Tuminaro, R., Willenbring, J., and Williams, A., An overview of trilinos, Sandia Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

39. Prudencio, E. E. and Schulz, K., The parallel C++ statistical library 'QUESO': Quantification of uncertainty for estimation, simulation and optimization, *Proc. of the Worshops of the Euro-Par 2011 Conference*, Lecture Notes in Computer Science series, Vol. 7155, Springer, 2011.
40. Beck, J. L. and Au, S. K., Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation, *J. Eng. Mech.*, 128:380–391, 2002.
41. Ching, J. and Chen, Y., Transitional markov chain monte carlo method for Bayesian model updating, model class selection, and model averaging, *J. Eng. Mech.*, 133(7):816–832, 2007.
42. Chen, D.-S., Batson, R. G., and Dang, Y., *Applied Integer Programming: Modeling and Solution*, Wiley, New York, 2010.
43. Makhorin, A., GNU linear programming kit, http://www.gnu.org/software/glpk/, 2000–2011.
44. Boman, E., Devine, K., Fisk, L. A., Heaphy, R., Hendrickson, B., Leung, V., Vaughan, C., Catalyurek, U., Bozdag, D., and Mitchell, W., Zoltan home page, http://www.cs.sandia.gov/Zoltan, 1999.
45. TACC, Texas advanced computing center, http://www.tacc.utexas.edu/, 2008–2011.

6. THE BINARY INTEGER PROGRAMMING FORMULATION

In this appendix we give a detailed derivation of formulation (15). Let $x_{ij} \in \{0,1\}$ take value 1 (or 0) depending if the ith task is handled (or not) by the jth processor, $1 \leqslant i \leqslant N_t$ and $1 \leqslant j \leqslant N_p$, and let $\mathbf{X}$ be the $N_t \times N_p$ matrix formed by values $x_{ij} \in \{0,1\}$. Problem (14) then can be formulated as a min-max problem, as follows:

$$\begin{cases} \min\limits_{\mathbf{X} \in \{0,1\}^{N_t \times N_p}} \quad \max\limits_{j \in \{1,\ldots,N_p\}} \quad n_j \;=\; \sum\limits_{i=1}^{N_t} a_i \cdot x_{ij}, \\ \qquad\qquad \text{subject to} \quad \sum\limits_{j=1}^{N_p} x_{ij} = 1, \quad \forall\, i = 1, \ldots, N_t, \end{cases} \tag{22}$$

where $\mathbf{X}$ is a $N_t \times N_p$ matrix formed by 0 and 1, and the equality constraint guarantees that each ith task is assigned to only one processor.

Now, let $\mathbb{Z}_+$ be the set of positive integers and let us denote

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_{N_t} \end{bmatrix}_{N_t \times 1} \in {\mathbb{Z}_+}^{N_t},$$

let $\mathbf{r}_i$ be the vector of size N_p formed by the components of the ith row of $\mathbf{X}$,

$$\mathbf{r}_i \equiv \begin{bmatrix} x_{i1} \\ \vdots \\ x_{iN_p} \end{bmatrix}_{N_p \times 1} \in \{0,1\}^{N_p},\ i = 1, \ldots, N_t,$$

let $\mathbf{c}_j$ be the vector of size N_t formed by the components of the jth column of $\mathbf{X}$,

$$\mathbf{c}_j \equiv \begin{bmatrix} x_{1j} \\ \vdots \\ x_{N_t j} \end{bmatrix}_{N_t \times 1} \in \{0,1\}^{N_t},\ j = 1, \ldots, N_p,$$

and let $\mathbf{r}$ and $\mathbf{c}$ be the vectors defined by

$$\mathbf{r} \equiv \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{N_t} \end{bmatrix}_{N_t N_p \times 1}, \quad \mathbf{c} \equiv \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_{N_p} \end{bmatrix}_{N_p N_t \times 1}.$$

Also, let us denote

$$\mathbf{m}_1 \equiv \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N_p \times 1}, \quad \mathbf{m}_0 \equiv \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{N_p \times 1}, \quad \mathbf{1}_{N_t \times 1} \equiv \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}_{N_t \times 1}.$$

and let $\mathbf{M}$ be the $N_t \times N_t N_p$ matrix defined by

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^T & \mathbf{m}_0^T & \dots & \mathbf{m}_0^T \\ \mathbf{m}_0^T & \mathbf{m}_1^T & \dots & \mathbf{m}_0^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{m}_0^T & \mathbf{m}_0^T & \dots & \mathbf{m}_1^T \end{bmatrix}_{N_t \times N_t N_p}.$$

Problem (22) then can be written as

$$\begin{cases} \min\limits_{\mathbf{X} \in \{0,1\}^{N_t \times N_p}} \quad \max\limits_{j \in \{1,\dots,N_p\}} \quad n_j \;=\; \mathbf{a}^T \cdot \mathbf{c}_j, \\ \\ \qquad\qquad\qquad\qquad \text{subject to} \quad \mathbf{M} \cdot \mathbf{r} = \mathbf{1}_{N_t \times 1}. \end{cases}$$

Another possible equivalent formulation of problem (22) is

$$\begin{cases} \min\limits_{\mathbf{X} \in \{0,1\}^{N_t \times N_p}} \quad n_1 \;=\; \mathbf{a}^T \cdot \mathbf{c}_1, \\ \\ \text{subject to} \quad \begin{cases} n_1 \;\geqslant\; n_2, \\ n_2 \;\geqslant\; n_3, \\ \qquad \vdots \\ n_{N_p-1} \;\geqslant\; n_{N_p}, \\ \mathbf{M} \cdot \mathbf{r} \;=\; \mathbf{1}_{N_t \times 1}, \end{cases} \end{cases}$$

that is,

$$\begin{cases} \min\limits_{\mathbf{X} \in \{0,1\}^{N_t \times N_p}} \quad n_1 \;=\; \mathbf{a}^T \cdot \mathbf{c}_1, \\ \\ \text{subject to} \quad \begin{cases} \mathbf{a}^T \cdot (\mathbf{c}_2 - \mathbf{c}_1) \;\leqslant\; 0, \\ \qquad \vdots \\ \mathbf{a}^T \cdot (\mathbf{c}_{N_p} - \mathbf{c}_{N_p-1}) \;\leqslant\; 0, \\ \mathbf{M} \cdot \mathbf{r} \;=\; \mathbf{1}_{N_t \times 1}. \end{cases} \end{cases} \tag{23}$$

Defining the vectors $\mathbf{g}_i$, $i = 1, 2, \dots, N_t$, each of size $N_t N_p$, by

$$\mathbf{g}_i^T \cdot \mathbf{r} = \mathbf{a}^T \cdot \mathbf{c}_i \quad \forall\, \mathbf{X} \in \{0,1\}^{N_t \times N_p},$$

and the matrix $\mathbf{G}$ and vector $\mathbf{0}_{(N_p-1)\times 1}$ by

$$\mathbf{G} \equiv \begin{bmatrix} \mathbf{g}_2^T - \mathbf{g}_1^T \\ \mathbf{g}_3^T - \mathbf{g}_2^T \\ \vdots \\ \mathbf{g}_{N_p}^T - \mathbf{g}_{N_p-1}^T \end{bmatrix}_{(N_p-1)\times N_t N_p}, \mathbf{0}_{(N_p-1)\times 1} \equiv \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{(N_p-1)\times 1},$$

then problem (23) can be written as the following BIP problem:

$$\begin{cases} \min\limits_{\mathbf{r} \in \{0,1\}^{N_t N_p \times 1}} & n_1 = \mathbf{g}_1^T \cdot \mathbf{r}, \\ \text{subject to} & \begin{cases} \mathbf{G} \cdot \mathbf{r} & \leqslant & \mathbf{0}_{(N_p-1)\times 1}, \\ \mathbf{M} \cdot \mathbf{r} & = & \mathbf{1}_{N_t \times 1}, \end{cases} \end{cases} \tag{24}$$

which is problem (15).