Computational Geosciences
https://doi.org/10.1007/s10596-020-09947-4

ORIGINAL PAPER

Reversible and non-reversible Markov chain Monte Carlo algorithms for reservoir simulation problems

P. Dobson[1] · I. Fursov[2] · G. Lord[1] · M. Ottobre[1]

Received: 4 April 2019 / Accepted: 11 February 2020

Abstract

We compare numerically the performance of reversible and non-reversible Markov Chain Monte Carlo algorithms for high-dimensional oil reservoir problems; because of the nature of the problem at hand, the target measures from which we sample are supported on bounded domains. We compare two strategies to deal with bounded domains, namely reflecting proposals off the boundary and rejecting them when they fall outside of the domain. We observe that for complex high-dimensional problems, reflection mechanisms outperform rejection approaches and that the advantage of introducing non-reversibility in the Markov Chain employed for sampling is more and more visible as the dimension of the parameter space increases.

Keywords Markov chain Monte Carlo methods · Non-reversible Markov chains · Subsurface reservoir simulation · High-dimensional sampling

1 Introduction

Markov Chain Monte Carlo (MCMC) methods are popular algorithms which allow one to sample from a given target measure π on $\mathbb{R}^N$. In combination with the Bayesian inference approach, MCMC methods have been very successfully implemented in a vast range of problems in the applied sciences, and the literature about MCMC is extensive. The purpose of MCMC algorithms is to build a Markov chain $\{x_k\}_{k\in\mathbb{N}}$ which has the measure π as invariant measure. Traditionally this is obtained by ensuring that the chain satisfies the *detailed balance* condition with respect to the measure π, so that the resulting chains are *reversible* with respect to π. In recent years, *non-reversible* MCMC algorithms have attracted a lot of attention, because of their favourable convergence and mixing properties; the literature on the matter has rapidly become large, here we refer the reader to e.g. [2, 3, 7, 9, 17, 18] and references therein; however, to the best of our knowledge, most of the papers on non-reversible MCMC so far have tested this new class of algorithms only on relatively simple target measures. Furthermore, the performance of non-reversible algorithms has been discussed almost exclusively in the case in which the measure is supported on the whole of $\mathbb{R}^N$. However, in many applications, it is very important to be able to sample from measures supported on bounded domains. This is the case, for example, in applications to reservoir modelling and petroleum engineering, which we treat in this paper. The purpose of this paper is twofold: on the one hand, we want to test the performance of non-reversible algorithms for complex, high-dimensional problems, which are completely out of reach for a full analytical treatment; on the other hand, we want to employ them for situations in which the target measure is supported in a bounded domain. The non-reversible algorithms that we consider in this paper are the so-called Horowitz algorithm, see [12], and the second-order Langevin-Hamiltonian Monte Carlo (SOL-HMC) algorithm, introduced in [8]. Both of them are non-reversible modifications of the well-known Hamiltonian Monte Carlo (HMC) [16], which is

✉ M. Ottobre
m.ottobre@hw.ac.uk

P. Dobson
pd14@hw.ac.uk

I. Fursov
i.fursov@hw.ac.uk

G. Lord
g.j.lord@hw.ac.uk

[1] Maxwell Institute for Mathematical Sciences, Department of Mathematics, Heriot-Watt University, Edinburgh, EH14 4AS, UK

[2] Institute of Petroleum Engineering, Heriot-Watt University, Edinburgh, EH14 4AS, UK

Published online: 13 March 2020

reversible. More precisely, the Horowitz algorithm is a non-reversible version of HMC and the SOL-HMC algorithm is a modification of the Horowitz algorithm, well-posed in infinite dimensions and therefore well-adapted to sample from the high-dimensional target measures that we will treat here.

All the algorithms we discuss in this paper need in principle no modification in order to sample from measures supported on bounded domains. However, if they are not suitably modified, they will employ proposal moves which fall outside of the support of the target measure. For the problem we consider, this seems to give two major drawbacks, namely (i) proposal moves that fall outside of the box are immediately rejected, so the algorithm wastes time rejecting moves which one knows a priori should not be made;[1] (ii) the likelihood function is calculated through the use of a simulator, which, further to being time-consuming to run, it will mostly fail to calculate correctly values that fall outside the support of the target. For this reason, we will consider two modifications of each one of the mentioned algorithms in which proposal moves that fall outside of the support of the target measures are either rejected or bounced off (or better, reflected off) the boundary of the support (see Section 2), so that the proposal is not automatically rejected as it will fall within the support. With these observations in mind, let us come to summarise the main conclusions of the paper:

- We compare rejection and reflection strategies and test them on both low- and high-dimensional targets and conclude that, for the problems at hand, the two strategies perform similarly when implemented in low dimensions; however, in high dimensions (and for more complex problems where a proxy is employed for the likelyhood function), reflections seem more advantageous.
- We compare the performance of HMC, Horowitz and SOL-HMC and conclude that, in high dimensions, the SOL-HMC algorithm is substantially outperforming the other two.

Performance of all the algorithms is compared by using the normalised effective sample size (nESS) as a criterion for efficiency, see Section 4. We emphasise that one of the main purposes of this paper is to demonstrate how the SOL-HMC algorithm, while being a simple modification of the HMC algorithm, which requires truly minimal code adjustment with respect to HMC, can bring noticeable improvements with respect to the latter method; furthermore, such improvements are more noticeable when tackling high-dimensional complex target measures.

[1] Admittedly, this observation only applies when the size of the domain is known a priori. See also [4].

The paper is organised as follows: in Section 2, we recall the HMC, SOL-HMC and Horowitz algorithms, present the numerical integrators that we use in order to implement such algorithms and introduce the versions of such methods which are adapted to sampling from measures with bounded support. In Section 3, we give details about the types of target measures used to compare the efficiency of these various algorithms and how such measures arise from reservoir simulation problems. This section explains mostly the mathematical structure of such target measures. Further details regarding the simulator and some basic background material about the reservoir model are deferred to Appendix 2. In Section 4, we present numerical experiments. For completeness, we include Appendix 1, containing some simple theoretical results regarding the modified algorithms. In particular, in Appendix 1, we show that our modification of SOL-HMC (i.e. SOL-HMC with reflections) is still non-reversible and it leaves the target measure invariant.

2 Description of the algorithms

In this section, we present the three main algorithms that we would like to compare, namely the Hamiltonian Monte Carlo (HMC) algorithm, the SOL-HMC algorithm and the Horowitz algorithm. With abuse of notation, throughout we will denote a probability measure and its density with the same letter, i.e. $\pi(dx) = \pi(x)dx$.

Suppose we wish to sample from a probability measure π defined on $\mathbb{R}^N$ which has a density of the form

$$\pi(x) \propto e^{-V(x)} e^{-\langle x, C^{-1}x\rangle},$$

i.e. the target density π is absolutely continuous with respect to a Gaussian measure with covariance matrix C (as customary, we assume that such a matrix is symmetric and positive definite). All three of our algorithms make use of the common trick of introducing an auxiliary variable $p \in \mathbb{R}^N$ and sampling from the density $\tilde{\pi}$ defined on the extended state space $\mathbb{R}^N \times \mathbb{R}^N$ as follows

$$\tilde{\pi}(x, p) \propto e^{-V(x)} e^{-\langle x, C^{-1}x\rangle} e^{-\frac{1}{2}p^2}. \tag{2.1}$$

The original target measure π is therefore the marginal of $\tilde{\pi}$ with respect to the variable x. More precisely, the algorithms we will consider generate chains $\{(x^k, p^k)\}_k \subset \mathbb{R}^N \times \mathbb{R}^N$ which sample from the measure (2.1); because (2.1) is a product measure of our desired target times a standard Gaussian, if we consider just the first component of the chain $\{(x^k, p^k)\}_k$, i.e. the chain $\{x^k\}_k$, then, for k large enough, such a chain will be sampling from the measure π. We now focus on explaining how the chain $\{(x^k, p^k)\} \subset \mathbb{R}^N \times \mathbb{R}^N$ is generated by the three algorithms we consider.

Let us introduce the Hamiltonian function

$$H(x, p) = V(x) + \langle x, C^{-1}x\rangle + \frac{1}{2}p^2\,; \tag{2.2}$$

then the associated Hamiltonian flow can be written as

$$\begin{cases} \dot{x} = p \\ \dot{p} = -x - C\nabla V(x). \end{cases} \tag{2.3}$$

Let χ^t denote a numerical integrator for the system (2.3) up to time t (we will comment below on our choices of integrator). The **HMC algorithm** then proceeds as follows: suppose that at time k the first component of the chain is x^k and

(1) Pick $p^k \sim N(0, I)$
(2) Compute

$$(\tilde{x}^{k+1}, \tilde{p}^{k+1}) = \chi^\delta(x^k, p^k)$$

and propose $\tilde{x}^{k+1}$ as the next move
(3) Calculate the acceptance probability α_k, according to

$$\alpha_k = \min(1, e^{-(H(\tilde{x}^{k+1}, \tilde{p}^{k+1}) - H(x^k, p^k))})$$

(4) Set $q^{k+1} = \tilde{q}^{k+1}$ with probability α_k, otherwise set $q^{k+1} = q^k$

In principle, any numerical integrator can be used in an HMC algorithm (see [16, 20] for more detailed comments on this). In this paper, we will consider two numerical integrators χ, which are two popular splitting schemes. The first is given by splitting the "momentum" and "position" equations, see e.g. [20] and references therein. That is, let $\mathbf{M}^t$ denote the solution map at time t of the system

$$\begin{cases} \dot{x} = 0, \\ \dot{p} = -x - C\nabla V(x) \end{cases} \tag{2.4}$$

and $\mathbf{P}^t$ denote the solution map at time t of the system

$$\begin{cases} \dot{x} = p \\ \dot{p} = 0. \end{cases} \tag{2.5}$$

For HMC, we shall always use the numerical integrator

$$\chi_H^\delta = \mathbf{M}^{\delta/2}\mathbf{P}^\delta\mathbf{M}^{\delta/2}\,. \tag{2.6}$$

Note that we can always write the maps $\mathbf{M}^t$ and $\mathbf{P}^t$ explicitly; indeed,

$$\mathbf{M}^{\delta/2}(x, p) = \left(x, p - \frac{\delta}{2}x - C\nabla V(x)\right) \tag{2.7}$$

$$\mathbf{P}^\delta(x, p) = (x + \delta p, p)\,. \tag{2.8}$$

The other splitting scheme that we will consider splits the Hamiltonian system (2.3) into its linear and nonlinear part. More precisely, let R^t and Θ^t be the flows associated with the following ODEs:

$$R^t : \begin{cases} \dot{x} = p, \\ \dot{p} = -x, \end{cases} \quad \Theta^t : \begin{cases} \dot{x} = 0, \\ \dot{p} = -C\nabla V(x). \end{cases} \tag{2.9}$$

The resulting integrator is given by

$$\chi_S^\delta = \Theta^{\delta/2}R^\delta\Theta^{\delta/2}. \tag{2.10}$$

This is the integrator that we will use in the SOL-HMC algorithm (see step (1) of the SOL-HMC algorithm below); the use of this splitting scheme for high-dimensional problems has been studied in [10]. SOL-HMC is motivated as a time discretisation of the SDE

$$\begin{cases} dx &= pdt, \\ dp &= [-x - C\nabla V(x)]dt - pdt + \sqrt{2C}dW_t, \end{cases} \tag{2.11}$$

where $\{W_t\}_{t\geq 0}$ is a standard N-dimensional Brownian motion. Such an equation can be seen as a Hamiltonian dynamics perturbed by an Ornstein-Uhlenbeck process in the momentum variable. As is well known, the SDE (2.11) admits the measure (2.1) as unique invariant measure, see e.g. [21]. With these observations in mind, define $\mathcal{O}^\varepsilon$ to be the map which gives the solution at time ε of the system

$$\begin{cases} dx &= 0, \\ dp &= -pdt + \sqrt{2}dW_t. \end{cases} \tag{2.12}$$

Note that we may solve this system explicitly, indeed

$$\mathcal{O}^\varepsilon(x, p) = (x, pe^{-\varepsilon} + (1 - e^{-2\varepsilon})^{\frac{1}{2}}\xi) \tag{2.13}$$

where ξ is a standard normal random variable. In Section 4, we will set

$$e^{-2\varepsilon} = 1 - i^2, \tag{2.14}$$

where i is a parameter we can tune; in which case we have

$$\mathcal{O}^\varepsilon(x, p) = (x, pe^{-\varepsilon} + i\xi).$$

The **SOL-HMC algorithm** is as follows:

(1) Given (x^k, p^k), let

$$(\hat{x}^k, \hat{p}^k) = \mathcal{O}^\varepsilon(x^k, p^k)$$

and propose

$$(\tilde{x}^{k+1}, \tilde{p}^{k+1}) = \chi_S^\delta(\hat{x}^k, \hat{p}^k),$$

where we recall that χ_S^δ is the integrator introduced in Eq. 2.10
(2) Calculate the acceptance probability α_k according to

$$\alpha_k = \min(1, e^{-(H(\tilde{x}^{k+1}, \tilde{p}^{k+1}) - H(\hat{x}^k, \hat{p}^k))}) \tag{2.15}$$

(3) Set

$$(x^{k+1}, p^{k+1}) = \begin{cases} (\tilde{x}^{k+1}, \tilde{p}^{k+1}) & \text{with probability } \alpha_k, \\ (\hat{x}^k, -\hat{p}^k) & \text{with probability } 1 - \alpha_k \end{cases}$$

Note the momentum flip in case of rejection. This momentum flip is in principle needed in any HMC algorithm. However, in HMC, because the momentum is re-sampled independently at every step (and the Hamiltonian

is an even function of p) in practice, one does not need to change the sign of the momentum upon rejection.

Finally, the algorithm that we will refer to as the **Horowitz algorithm** is just the SOL-HMC algorithm when in step one, instead of using the integrator χ_S, we use the integrator χ_H (defined in Eq. 2.6).

Remark 2.1 We do not give many details about HMC, SOL-HMC and the Horowitz algorithm here, and refer to the already cited literature. However, we would like to stress the two following facts:

- The chain $\{x^k\}_k$ produced by the HMC algorithm is reversible with respect to the measure π, in the sense that it satisfies detailed balance with respect to π [16]—more precisely, the chain $\{(x^k, p^k)\}_k$ generated by HMC satisfies a generalised detailed balance condition with respect to $\tilde{\pi}$, see e.g. [20, Lemma 1] or [5]. In contrast, the chains generated by the Horowitz algorithm and the SOL-HMC do not satisfy any form of detailed balance with respect to $\tilde{\pi}$ and they are therefore non-reversible, see [7, 8]. In Appendix 1, we will show that adding reflections to the algorithm does not alter this property. That is, SOL-HMC with reflections is still non-reversible. Note moreover that the fact of adding reflections does not alter the invariant measure (see Appendix 1).
- The difference between the Horowitz algorithm and HMC may seem small, but in reality, this small difference is crucial. Indeed, thanks to this choice of integrator, SOL-HMC is well-posed in infinite dimensions, while the Horowitz algorithm is not. For a discussion around this matter, see [7, 10].

As mentioned in the introduction, in this paper, we will be interested in sampling from measures which are not necessarily supported on the whole space $\mathbb{R}^N$, but just on some box $B = [-a, a]^N$. If this is the case, then one may still use any one of the above algorithms and reject proposal moves that fall outside the box. We will briefly numerically analyse this possibility (see Section 4). Alternatively, one may want to simply make sure that all the proposed moves belong to the box B, so that the algorithm does not waste too much time rejecting the moves that fall outside the box. We therefore consider modified versions of the introduced algorithms by introducing appropriate reflections to ensure that all of the proposals belong to the box B. Because the proposal is defined through numerical integration of the Hamiltonian dynamics, we will need to modify the integrators χ_H and χ_S.

First, consider the map $\mathbf{P}^\delta$ defined in Eq. 2.8; then, we define map $\mathbf{P}^\delta_{\text{bounce}}$ recursively as follows:

(1) If $\mathbf{P}^\delta(x, p) \in B$, then set $\mathbf{P}^\delta_{\text{bounce}}(x, p) = \mathbf{P}^\delta(x, p)$.

(2) Otherwise, define

$$\alpha = \inf\{\beta \in [0, 1] : \mathbf{P}^{\beta\delta}(x, p) \notin B\}. \tag{2.16}$$

In which case $\mathbf{P}^{\alpha\delta}(x, p)$ lies on the boundary of the box, so there exists some[2] $j \in \{1, \dots, N\}$ such that the jth component of $\mathbf{P}^{\alpha\delta}(x, p)$ is either a or $-a$. Then we define

$$\mathbf{P}^\delta_{\text{bounce}}(x, p) = \mathbf{P}^{(1-\alpha)\delta}_{\text{bounce}}(S_j(\mathbf{P}^{\alpha\delta}(x, p)). \tag{2.17}$$

Here S_j is the reflection map $S_j(x, p) = (x, p_1, \dots, p_{j-1}, -p_j, p_{j+1}, \dots, p_N)$.

Similarly, we define R^δ_{bounce} by

(1) If $R^\delta(x, p) \in B$, then set $R^\delta_{\text{bounce}}(x, p) = R^\delta(x, p)$.

(2) Otherwise, define

$$\alpha = \inf\{\beta \in [0, 1] : R^{\beta\delta}(x, p) \notin B\}. \tag{2.18}$$

In which case $R^{\alpha\delta}(x, p)$ lies on the boundary of the box, so there exists some $j \in \{1, \dots, N\}$ such that the jth component of $R^{\alpha\delta}(x, p)$ is either a or $-a$. Then we define

$$R^\delta_{\text{bounce}}(x, p) = R^{(1-\alpha)\delta}_{\text{bounce}}(S_j(R^{\alpha\delta}(x, p))). \tag{2.19}$$

Note that it may occur that $R^\delta(x, p) \in B$; however, there is some point $\alpha \in [0, 1]$ such that $R^{\alpha\delta}(x, p) \notin B$, in this case, we still set $R^\delta_{\text{bounce}}(x, p) = R^\delta(x, p)$. Therefore, the algorithm HMC-bounce (Horowitz-bounce, respectively) is defined like HMC (Horowitz, respectively), but the numerical $\chi^\delta_{H,Bounce} = \mathbf{M}^{\delta/2}\mathbf{P}^\delta_{\text{bounce}}\mathbf{M}^{\delta/2}$ is employed in place of the integrator χ_H; analogously, the algorithm ***SOL-HMC-bounce*** is defined as the algorithm SOL-HMC with numerical integrator $\chi^\delta_{S,Bounce} = \Theta^{\delta/2}R^\delta_{\text{bounce}}\Theta^{\delta/2}$ in place of χ_S (recall the definition of Θ^t has been given in Eq. 2.9).

To conclude this section, we point out that alternative approaches to handling constraints in the framework of HMC-type algorithms have been also discussed in [16, Section "Handling Constraints"]; there "forbidden values" of the variables to sample from (i.e. values outside the box, in our context) are ruled out by setting theoretically the potential energy to infinity for such values. In [3], the issue of handling constraints has been looked at in the general context of Piecewise Deterministic Markov Processes.

3 Target measures

In this section, we describe the three target measures that will be the object of our simulations. The first measure we consider, π_{Ros}, is a change of measure from the popular 5D Rosenbrock, see Eq. 3.2. This is the most artificial

[2] It could occur that there is more than one j such that the jth component of $\mathbf{P}^{\alpha\delta}(x, p)$ is $\pm a$, in which case apply the operator S_j for all such j.

example we consider. The other two target measures are posterior measures for parameter values in reservoir simulation models. Roughly speaking, the second target measure we describe, π_{full}, is a posterior measure on the full set of parameters of interest in a reservoir model; for our reservoir model, which is quite realistic, we will be sampling from 338 parameters; hence, this measure will be a measure supported on $\mathbb{R}^{338}$. The third measure, π_{light}, is a measure on $\mathbb{R}^{21}$, which derives from considering a considerably simplified reservoir model. We will refer to the former as *full reservoir model* and to the latter as *lightweight parametrisation*. In this section, we explain the mathematical structure of π_{full} and π_{light}, without giving many details regarding the inverse problem related to the reservoir model. More details about the reservoir model and the simulator used to produce the likelihood function have been included in Appendix 2 for completeness. In the following, we let I_N denote the $N \times N$ identity matrix. Let us now come to describe our targets.

- **Change of measure from 5D Rosenbrock (i.e. π_{Ros}).** The first target measure we consider is a measure on $\mathbb{R}^5$ and it is a change of measure from the 5D Rosenbrock measure; namely, the density of 5D Rosenbrock is given by

$$f(x) = \sum_{i=1}^{4} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2, \quad x = (x_1, \ldots, x_5). \tag{3.1}$$

  The target we consider is given by

$$\pi_{Ros}(x) \propto e^{-\frac{1}{2}f(x)} e^{-\frac{1}{2}\langle x, C^{-1}x\rangle}, \tag{3.2}$$

  where C is the prior covariance matrix. In all the numerical experiments (see Section 4) regarding π_{Ros}, we take $C = 0.3 \cdot I_5$.
- **Full reservoir simulation**. We study a Bayesian inverse problem for reservoir simulation. We consider a synthetic reservoir with 7 layers, and 40 producing/injecting wells. A diagrammatic representation of the reservoir is shown in Fig. 1.[3] Each layer is divided in blocks and, while the well goes through all the layers, it will not necessarily communicate through perforations with all the blocks it goes through (in Fig. 1 we highlight in yellow the boxes containing a perforation of a well). We also assume that, in each layer, the well goes through at most one block. In total, our subsurface reservoir is made of 124 blocks: 38 blocks on the boundaries to represent the active aquifers, one block per layer per well, plus some additional blocks (which are neither aquifer blocks nor crossed by the wells).

The reservoir properties (i.e. the parameters that we will be interested in sampling from) are described by the *pore volumes* V_ℓ of the blocks, $\ell \in \{1, \ldots, 124\}$, the *transmissibilities* $T_{\ell j}$ between the interconnected blocks ℓ and j and the *perforation productivity coefficients* $J_{w\ell}$ for the well-block connections. We do not explain here the practical significance of such parameters and for more details on reservoir simulation, we refer the reader to [1]. Altogether, the parameter space for this example is 338-dimensional. For the sake of clarity, all (non-zero) $T_{\ell j}$ are re-indexed with a single index as T_p, $p \in \{1, \ldots, 139\}$; and similarly, $J_{w\ell}$ are re-indexed as J_k, $k \in \{1 \ldots 75\}$ and we denote by $x \in \mathbb{R}^{338}$ the full vector of parameters, i.e. $x = (V_1, \ldots, V_{124}, T_1, \ldots, T_{139}, J_1, \ldots J_{75})^T$. There are 86 non-aquifer blocks in total, and we always assume an ordering of the parameters V_ℓ such that the first 86 of them correspond to the non-aquifer blocks. In our Bayesian inverse problem for the parameters x, the likelihood function is defined from the reservoir simulation output, and the prior is a Gaussian with covariance matrix C. The observed block pressure and the well bottom hole pressure (BHP) data are known for certain wells and time steps; we arrange such data into the vector d_0. The likelihood $L(d_0|x)$, see Eq. 3.3 below, is defined using the simulator-modelled data $d(x)$, the observed data d_0 and the covariance matrix for data errors C_d. The function $d(x)$ is found by numerical solution of a system of ordinary differential equations, which we report in Appendix 2, see Eqs. B.1 – B.4; such a system describes the relation between the vector of reservoir properties x and the simulated pressures. The important thing for the time being is that such a system is high dimensional and the resulting posterior is analytically intractable.[4] Finally, we seek to sample from the measure

$$\pi_{full}(x|d_0) \propto L(d_0|x) \cdot e^{-\langle x, C^{-1}x\rangle},$$

where the likelihood function is of the form

$$L(d_0|x) = \exp\left(-\frac{1}{2}(d(x) - d_0)^T C_d^{-1}(d(x) - d_0)\right). \tag{3.3}$$

In our numerical experiments, we will always take the matrix C_d to be diagonal, with the entries equal to either $\sigma^2_{BHP} = 20^2$ or $\sigma_b^2 = 9$. We will give more details about this choice in Appendix 2. The full parameterisation is further

[3] We emphasise that the reservoir we consider here is a multilayer reservoir. The reason for considering such a reservoir is that there exist (quite often) multilayer petroleum reservoirs, where well inflow from the different layers is generally unknown, i.e. the total inflow is known, but its allocation is uncertain. Resolving this uncertainty is an important issue for appropriate management of the reservoir and a challenge in uncertainty quantification. See [13–15, 22, 23].

[4] The simulator we use also allows for fast calculation of the gradients of the log likelihood by the adjoint procedure [19], so that HMC-type samplers can be run.

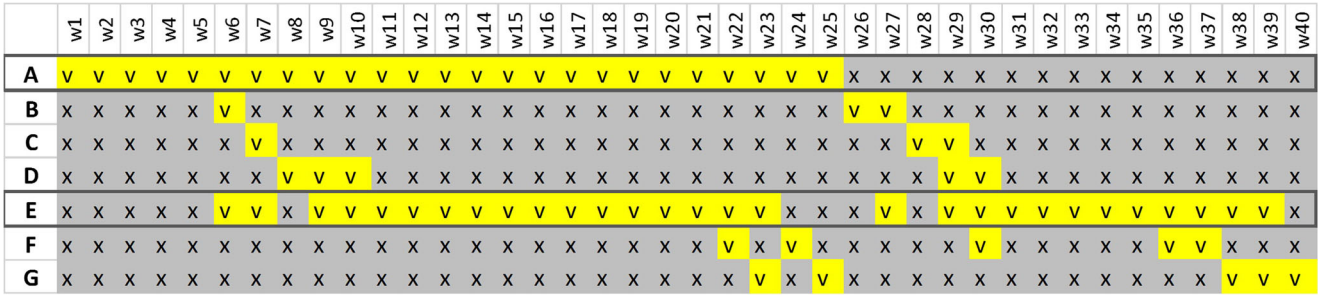

Fig. 1 Perforations of the 40 wells (columns) in the seven layers (rows). Yellow "v" stands for the block containing a perforation of a well. That is, the well goes through all the layers, but there is a hole for well-block communication only in correspondence of the yellow boxes. This figure does not show all the blocks—but only those perforated by the wells. In particular, it does not show the aquifer blocks located on the boundary

divided into three subcases denoted here as *full-a*, *full-b* and *full-c*, which have different min/max bounds for the parameters of interest or prior covariances. For the *full-a* case, we define the minimum L^i and maximum U^i bounds of each parameter $x_i \in \{V_\ell, T_p, J_k\}$ as follows: let $\bar{x}_i$ be the maximum likelihood value of the parameter x_i, found approximately, by running an optimisation algorithm on all the parameters;[5] we then take $L^i = 0.1\bar{x}_i$, $U^i = 10\bar{x}_i$, $i = 1, ..., 338$. Since the values of physical parameters x_i may differ by several orders of magnitude, it makes sense to apply a transform to get a similar magnitude for all the parameters. Such a transform was done by function $\log_{10}$ and a constant shift, mapping the parameters x_i from the original range $[L^i, U^i]$ to $[-1, 1]$. The prior covariance is taken as $C_{\text{full-a}} = 0.25 \cdot I_{338}$. So, all the parameters in the transformed representation vary within the box $[-1, 1]$ and have standard deviation 0.5. For the *full-b* case, wider parameter bounds are taken: $L^i = 0.001\bar{x}_i$, $U^i = 1000\bar{x}_i$, $i = 1, ..., 338$. The parameters are transformed by $\log_{10}$ function, and then mapped to the interval $[-3, 3]$. The prior covariance is the same as in the *full-a* case, so all the parameters have standard deviation 0.5 in the transformed representation. Case *full-c* uses the same parameter bounds and the same transform as case *full-b*, but a wider prior covariance $C_{\text{full-c}} = 9 \cdot C_{\text{full-a}}$, which means the prior standard deviation is 1.5 in the transformed representation.

- **Lightweight parameterisation** Here we consider a reduced, 21-dimensional, parameter space. Here we just fix the values of $V_1, \ldots, V_{86}$ (non-aquifer blocks), and we find the remaining $V_{87}, \ldots, V_{124}$ (aquifer blocks), $T_1, \ldots, T_{139}$ (all blocks), $J_1, \ldots J_{75}$ (all perforations), which are required by the simulator, using 21 new parameters. Such parameters essentially act as multipliers; namely, for each one of the seven layers $n \in \{A, \ldots G\}$, we introduce one *pore volume multiplier for the aquifer blocks* $\tilde{V}_n$, one *transmissibility multiplier* $\tilde{T}_n$ and one perforation productivity multiplier $\tilde{J}_n$. These parameters, collectively denoted by $y \in \mathbb{R}^{21}$, are those that we are interested in sampling from, by using the posterior measure

$$\pi_{\text{light}}(y) \propto L(d_0|X(y)) \cdot \rho(y),$$

where $\rho(y)$ is a zero mean Gaussian with covariance matrix denoted by C_{21}, described below. Because we are using the same simulator as for the full reservoir simulation, the likelihood function L is still the one defined in Eq. 3.3; hence, necessarily we must have $X(y) \in \mathbb{R}^{338}$. To define the function $X : \mathbb{R}^{21} \to \mathbb{R}^{338}$, we need to introduce some notation first. Denote by A_n the number of aquifer blocks in layer n, P_n the number of transmissibility coefficients in layer n and K_n the number of well perforations in this layer. Let $\bar{V}_\ell$ be the maximum likelihood value of the parameter V_ℓ (similarly for $\bar{T}_p$ and $\bar{J}_k$), again found by running a maximum likelihood algorithm, and let the corresponding full vector denoted by $\bar{x}$. The first 86 components of $X(y)$ (corresponding to non-aquifer V_ℓ) are taken equal to $\bar{V}_\ell$, $\ell = 1, \ldots 86$, irrespective of the input y. The remaining $338 - 86 = 252$ components of $X(y)$ are found by a linear mapping $M \cdot y$, using a 252×21 sparse matrix M. The first column of M contains the vector

$$(\bar{V}_{86+1}, \ldots, \bar{V}_{86+A_1}, 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 0)^T,$$

the second column contains the vector

$$(\underbrace{0, \ldots\ldots\ldots, 0}_{L_1}, \bar{V}_{86+L_1+1}, \ldots, \bar{V}_{86+A_1+A_2}, 0 \ldots 0)^T,$$

and so forth until the 7th column. The columns from 8 to 14 are built similarly, such that column $n + 7$ corresponds to layer n and has P_n non-zero values equal to $\bar{T}_p$ (for appropriate indices p). The last seven columns are built in the same way, this time using the values $\bar{J}_k$.

For the *lightweight* parameterisation, the following minimum and maximum bounds were employed: [0.15, 15]

[5] The optimisation algorithm used here is BFGS [11], but in principle, any other could be used.

for all $\tilde{V}_n$, [0.07, 7] for all $\tilde{T}_n$ and [0.11, 11] for all $\tilde{J}_n$. As before, the physical parameters (multipliers) y_i are mapped to the interval $[-1, 1]$. The prior covariance C_{21}, which acts in the transformed variables, was taken as essentially a diagonal matrix with the main diagonal equal to 0.25; however, additional non-zero covariances equal to 0.1 were also specified between the transmissibility multiplier $\tilde{T}_n$ and perforation productivity multiplier $\tilde{J}_n$ for each layer n. A brief summary of the four discussed cases of the model parameterisation is presented in Table 1.

4 Numerics: sampling from measures supported on bounded domains

To compare efficiency of the algorithms, we compute a normalised effective sample size (nESS), where the normalisation is by the number of samples N.

Following [6], we define the effective sample size $ESS = N/\tau_{int}$ where N is the number of steps of the chain (after appropriate burn-in) and τ_{int} is the integrated autocorrelation time, $\tau_{int} := 1+\sum_k \gamma(k)$, where $\gamma(k)$ is the $lag-k$ autocorrelation. Consistently, the normalised ESS, nESS, is just $nESS := ESS/N$. Notice that $nESS$ can be bigger than one (when the samples are negatively correlated), and this is something that will appear in our simulations. As an estimator for τ_{int}, we will take the Bartlett window estimator (see for example [6, Section 6], and references therein) rather than the initial monotone sequence estimator (see again [6, Section 6]), as the former is more suited to include non-reversible chains. Since the nESS is itself a random quantity, we performed 10 runs of each case using different seeds, and our plots below show P10, P50, P90 percentiles of the nESS from these runs (P50 being the middle circle point).

For reversible chains, criteria based on monitoring variance or on exponentially fast convergence are well known and it is known how such criteria are related to each other. For non-reversible chains, it is not instead clear how these criteria relate; that is, while some chains may perform well by the point of view of variance reduction, they will not if assessed with spectral methods (i.e. rate of convergence), see [3, 17, 18] and references therein. For this reason, we chose a simpler, broadly applicable criterion, the reliability of which is not altered by the non-reversibility. In our results below, each chain provides samples for all the unknown parameters; however, for clarity, we sometimes only display a representative subset—when this is the case, the choice of the paramenters is pointed out in the caption.

4.1 Bounces vs rejection

First we consider the performance of the two proposed methods for sampling from the box B. We illustrate these by comparing SOL-HMC-bounce and SOL-HMC-rej.

In Fig. 2, we compare performance of SOL-HMC-bounce and SOL-HMC-rej for sampling from the 5D Rosenbrock target π_{Ros}; each one of the five parameters is taken to vary in the interval $[-a, a]$, and Fig. 2 shows how the performance varies as the size a of the box varies, $a = 0.1, 0.2, \ldots, 1.4$. The target acceptance rate for both samplers was set to 0.9, and parameter $i = 0.6$ (defined in Eq. 2.14). For each value of a, each chain produced by any of the two algorithms will create samples for all five coordinates $x_1, \ldots, x_5$ of the target; in Fig. 2, we just display results for the first and the third coordinate (see caption).

As a "sanity test", the plots indicate that for the larger boxes ($a \geq 0.8$), the two implementations SOL-HMC-bounce and SOL-HMC-rej are almost identical (in terms of nESS), which is natural as for large box sizes, these two algorithms coincide. For small box sizes, the performance of the two samplers depends really on which coordinate is being sampled, so the performance of the two algorithms is substantially indistinguishable for this low-dimensional problem.

It is important to note the following practical drawback of SOL-HMC-rej (or indeed any other sampler which handles boundaries by the rejection mechanism) with respect to SOL-HMC-bounce: during the proposal step, a trajectory may leave the box, and then return back inside the box.

Table 1 Summary of the subcases for the reservoir simulation model

Case	Dim	Parameters notation	*For phys. par.*		*For transformed parameters*		
			l_i	u_i	Params range	Prior cov	Prior std
Lightweight	21	$\tilde{V}_n$, $\tilde{T}_n$, $\tilde{J}_n$, or y_i	0.1	10	$[-1, 1]$	$C_{21} \approx$ diag	≈ 0.5
Full-a	338	V_ℓ, T_p, J_k, or x_i	0.1	10	$[-1, 1]$	$C_{\text{full-a}} =$ diag	0.5
Full-b	338	V_ℓ, T_p, J_k, or x_i	0.001	1000	$[-3, 3]$	$C_{\text{full-b}} = C_{\text{full-a}}$	0.5
Full-c	338	V_ℓ, T_p, J_k, or x_i	0.001	1000	$[-3, 3]$	$C_{\text{full-c}} = 9C_{\text{full-a}}$	1.5

In physical representation, the lower bounds are $L^i = l_i b_i$, the upper bounds are $U^i = u_i b_i$, where l_i, u_i are reported in the table, and b_i are some base case parameter values (e.g. for all *full* parameterisations $b_i = \bar{x}_i$)

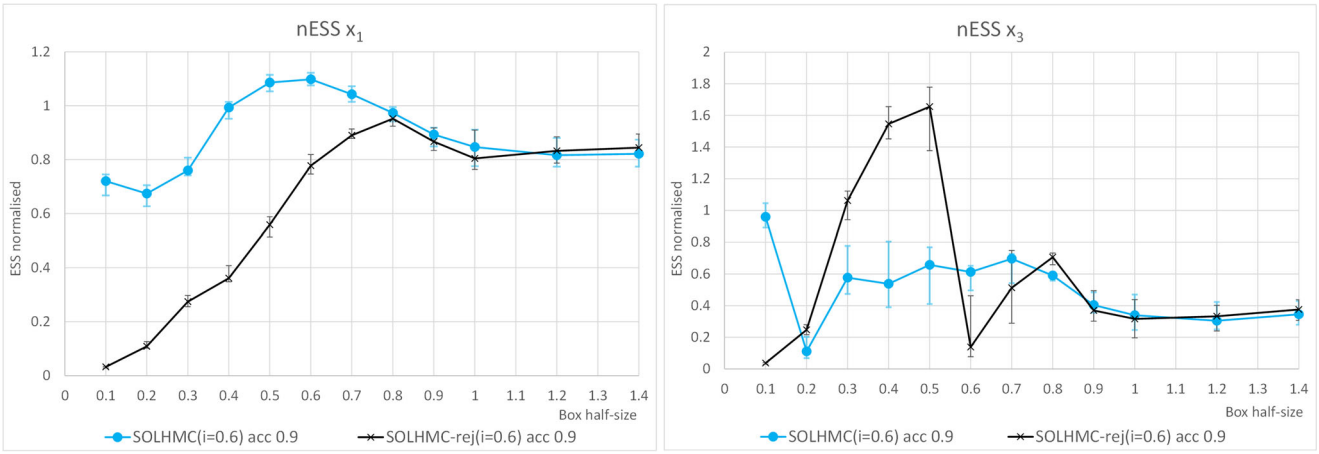

Fig. 2 Normalised ESS for SOL-HMC-bounce (blue) and SOL-HMC-rej (black) for different sizes of the box bounding the parameter space (X axis). The two plots correspond to coordinates x_1, x_3 only. The other three coordinates have nESS plots similar to x_3

By construction of the algorithm, such a trajectory is not rejected just because it escaped from the domain for a short while. The accept/reject decision is made only for the final point of the proposal trajectory, and thus every trajectory needs to be calculated till the end. However, if the trajectory is allowed to leave the box for the intermediate calculations, it may go to the extreme regions of the parameter space, where the simulator may suffer from severe numerical errors and abnormal behaviour. We illustrate this phenomenon by comparing SOL-HMC-rej against HMC-bounce in Fig. 3 for *full-a* in (A) and *full-b* in (B). (Here we think of HMC-bounce as sort of gold standard and for this reason, we compare SOL-HMC-rej with HMC-bounce). We examine the ratio of nESS of SOL-HMC-rej and HMC-bounce and plot a histogram for the parameters. When the nESS ratio is bigger than one, then SOL-HMC-rej is performing better than HMC-bounce. This is the case in (B) for *full-b*. However, in (A) for *full-a*, the boundary of B is encountered far more frequently, just because the size of the box for this target measure is smaller, see Table 1. Moreover, a comparison of the histograms in Fig. 3a with the one in Fig. 9a shows better performance of SOL-HMC-bounce with respect to SOL-HMC Rejections. From now on, we consider SOL-HMC-bounce only.

4.2 Comparison for 5D Rosenbrock

We consider the 5D Rosenbrock target π_{Ros} where the minimum-to-maximum range for each one of the five parameters was taken as $[-a, a]$, where $a = 0.1, 0.2, \ldots, 1.4$. The plots in Fig. 4 compare the performance of the HMC-bounce, SOL-HMC-bounce and Horowitz-bounce algorithms. The target acceptance rate is 0.9, and the parameter $i = 0.6$ for SOL-HMC-bounce and Horowitz-bounce. For this small dimensional problem, we observe that SOL-HMC-bounce and Horowitz-bounce have similar nESS across the range of sizes for the box B. For smaller boxes B (e.g. $a \leq 0.5$), all three algorithms have similar nESS. For larger box sizes, we see for parameter x_1 an advantage in using SOL-HMC-bounce/Horowitz-bounce over the HMC-bounce; however, for x_2, there is a slight advantage to HMC-bounce. This corroborates the idea that in low dimension, the advantage of introducing irreversibility in the sampler is hardly noticeable.

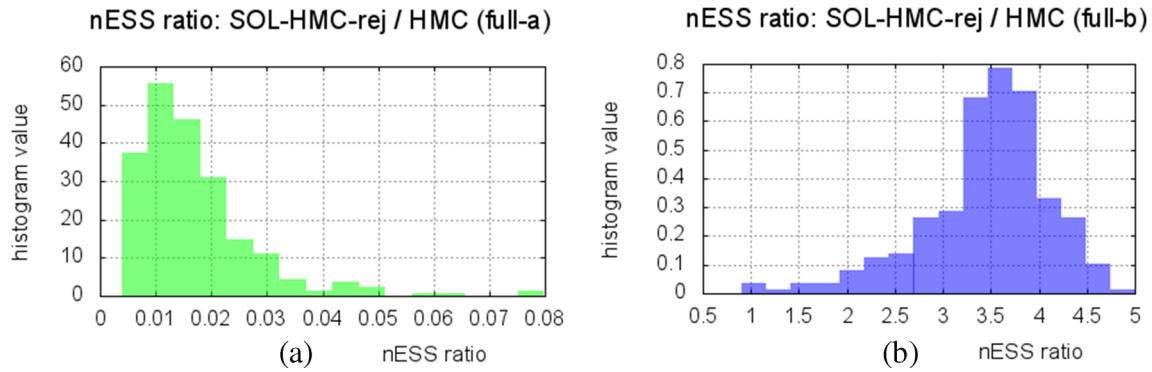

Fig. 3 Ratio of nESS (SOL-HMC-rej divided by HMC-bounce). Parameterisation *full-a* (**a**) is shown in green and *full-b* (**b**) in blue

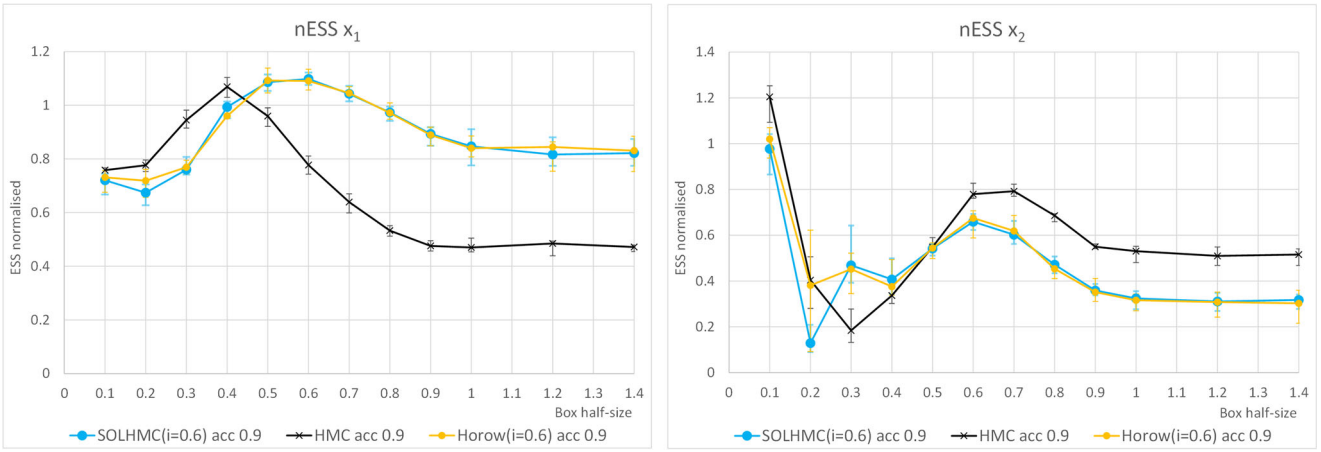

Fig. 4 Normalised ESS for SOL-HMC-bounce (blue), HMC (black) and Horowitz (orange), for different sizes of the box bounding the parameter space (X axis). The two plots correspond to coordinates x_1, x_2 only. The other three coordinates show a similar picture

4.3 Increasing parameter space and effectiveness of non-reversibility

We now consider our more realistic targets and increase the parameter space to 21 and then to 338. We now clearly see the advantage of the non-reversible algorithms SOL-HMC-bounce and Horowitz-bounce over HMC-bounce.

Figure 5 reports the nESS for the lightweight parameterisation of the reservoir simulation problem for the following four cases: HMC-bounce with an acceptance rate of 0.82 (target 0.8), SOL-HMC-bounce with acceptance rate of 0.77 (target 0.9), SOL-HMC-bounce with an acceptance rate of 0.69 (target 0.8) and Horowitz-bounce with an acceptance rate of 0.68 (target 0.8). All SOL-HMC-bounce and Horowitz-bounce algorithms took the parameter $i = 0.5$ and here we give results from a single MCMC run in each case. The plot clearly shows that the non-reversible algorithms outperform HMC for the majority of the parameters. We also observe the variability due to acceptance rate: for SOL-HMC-bounce, a better nESS is achieved for the higher acceptance rate.

As we further increase the dimension and complexity, the advantage of the non-reversible algorithm becomes further apparent. In Fig. 6, we compare for *full-a* SOL-HMC-bounce and HMC-bounce and observe a clear improved nESS for SOL-HMC-bounce across the whole parameter space.

Finally, we compare SOL-HMC-bounce and Horowitz against the benchmark of HMC-bounce by examining the ratio of nESS. Recall that when the ratio is bigger than one, then SOL-HMC-bounce (or Horowitz) has a larger nESS than HMC. We consider the targets *full-a*, *full-b* and *full-c*. In Fig. 7, we compare for *full-a* SOL-HMC-bounce against Horowitz-bounce. First, note that in both cases, the nESS ratio is > 1 for most parameters showing a clear improvement in the non-reversible algorithms over HMC. To aid comparison between SOL-HMC-bounce against Horowitz-bounce, we plot on (A) and (B) a fit of the histogram

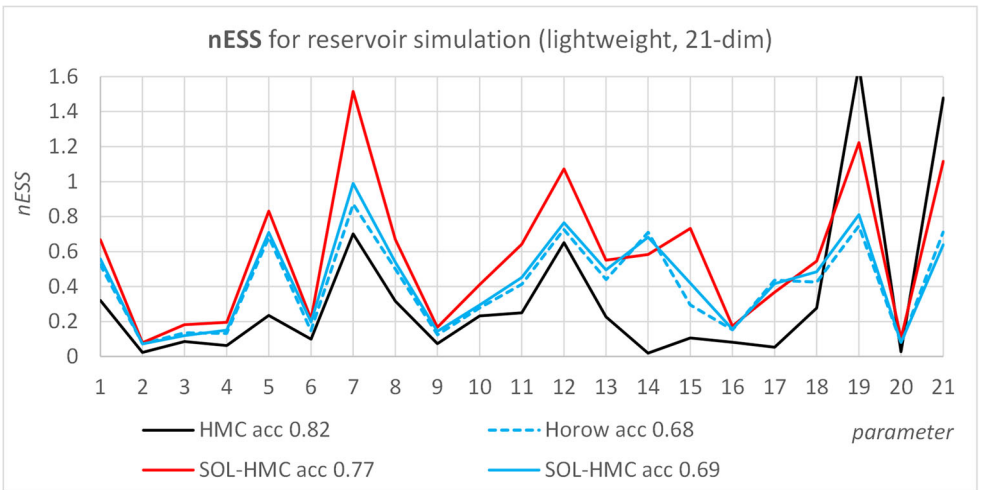

Fig. 5 Normalised ESS (Y axis) for the reservoir simulation MCMC, lightweight parameterisation. X axis shows the 21 parameters. In the legend, the real acceptance rates are indicated

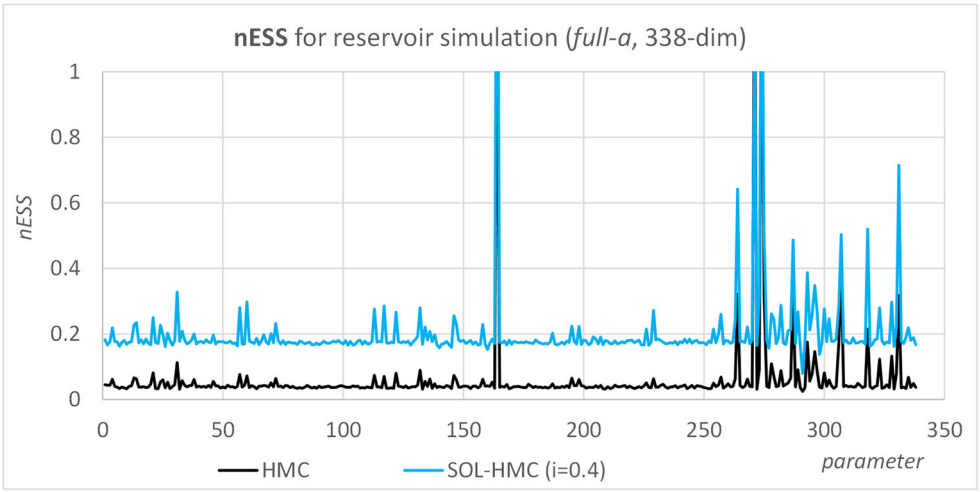

Fig. 6 Normalised ESS (Y axis) for the reservoir simulation MCMC, *full-a* parameterisation. X axis shows the 338 parameters. The samplers are HMC and SOL-HMC with $i = 0.4$

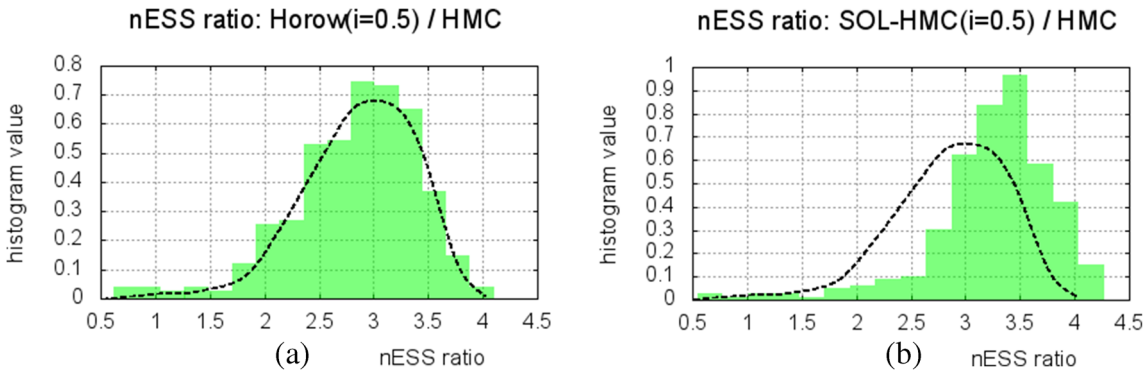

Fig. 7 Ratio of nESS for Horowitz-bounce by nESS for SOL-HMC-bounce. The target measure here is the 338-dimensional *full-a*. Parameter $i = 0.5$

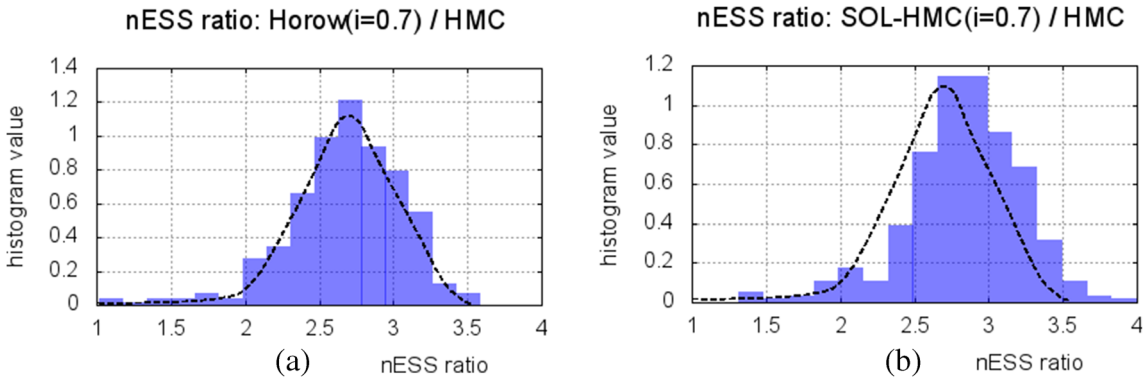

Fig. 8 Ratio of nESS for Horowtiz-bounce and SOL-HMC-bounce for target *full-b* ($i = 0.7$)

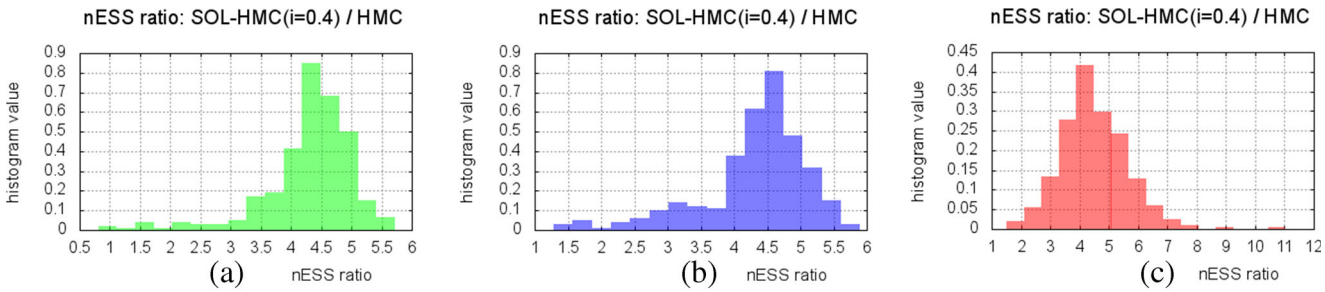

Fig. 9 Ratio of nESS for SOL-HMC-bounce for targets *full-a* (**a**), *full-b* (**b**) and *full-c* (**c**) and in each case $i = 0.4$

from (A), this is the black dotted line. We see that the nESS for SOL-HMC-bounce over the parameters is larger than that for Horowitz-bounce and that there is an improvement using SOL-HMC-bounce. Here we took $i = 0.5$. Figure 8 examines the target *full-b*. For both SOL-HMC-bounce and Horowitz-bounce, we see an improvement over the reversible HMC algorithm as the ratios are > 1 for all parameters. We also observe a shift to larger values and hence improvement in the nESS for SOL-HMC-bounce (B) compared with Horowitz-bounce (A). In this figure, we took $i = 0.7$. This can be compared with Fig. 9b where $i = 0.4$.

Finally, in Fig. 9, we examine SOL-HMC-bounce for *full-a* (A), *full-b* (B) and *full-c* (C) for the same value of $i = 0.4$. We see a clear improvement of the non-reversible SOL-HMC-bounce over HMC in each case. We compare here to the SOL-HMC-bounce for *full-b* for the same value of $i = 0.4$ in (B). We observe a similar improvement for SOL-HMC-bounce over HMC in both cases.

5 Conclusion

We have investigated two different ways to deal with sampling measures on a bounded box B: rejection and bounces. This is crucial in many practical applications, for example to respect physical laws (such as porosity for reservoir modelling or pixel values in image reconstruction). We have explained and demonstrated why, for complex problems involving the use of a proxy, reflection algorithms should be preferred to rejection strategies. We have furthermore shown that when sampling from complex realistic target measures, such as those that arise in reservoir simulation, non-reversible algorithms such as SOL-HMC and Horowitz outperform standard reversible algorithms such as HMC. In addition, we see that as the problem size grows SOL-HMC is superior to Horowitz having larger nESS.

Funding information The work of I. Fursov and G. J. Lord was supported by the EPSRC EQUIP grant (EP/K034154/1). P. Dobson was supported by the Maxwell Institute Graduate School in Analysis and its Applications (MIGSAA), a Centre for Doctoral Training funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016508/01), the Scottish Funding Council, Heriot-Watt University and the University of Edinburgh.

Appendix 1

This Appendix gathers some basic results about the SOL-HMC-bounce algorithm, presented in Section 2. Throughout we use the notation introduced in Section 2.

Proposition A.1 *The SOL-HMC-bounce algorithm with reflections preserves the target measure.*

Proof It is easy to see that the operator $\mathcal{O}^\varepsilon$ preserves the target measure $\tilde{\pi}$. Indeed $\mathcal{O}^\varepsilon$ leaves the x-variable untouched so, because $\tilde{\pi}$ is the product of $\pi(x)$ and a standard Gaussian in the p variable, looking at the definition (2.12)–(2.13) of $\mathcal{O}^\varepsilon$, all one needs to show is that if p is drawn from a standard Gaussian then $\hat{p} := pe^{-\varepsilon} + i\xi$ is also a Gaussian random variable—here ξ is a standard Gaussian independent of p. This is readily seen as, by definition, $\hat{p}$ has expectation 0 and variance 1, since $e^{-2\varepsilon} + i^2 = 1$. Therefore, if (x, p) are drawn from $\tilde{\pi}$, then $\mathcal{O}^\varepsilon(x, p) = (x, \hat{p})$ is also distributed according to $\tilde{\pi}$.

Let $\chi = \chi^\delta_{S,bounce}$ denote the integrator described in the SOL-HMC-bounce algorithm. Since $\mathcal{O}^\varepsilon$ preserves the target measure $\tilde{\pi}$, it remains to show that the combination of the integrator χ and the accept–reject mechanism preserves the target measure.

It is well known, for instance see [20, Theorem 9], that if the integrator $\chi^\delta_{S,bounce}$ is *reversible under momentum flip* (that is, $\chi^\delta_{S,bounce} \circ S = S \circ (\chi^\delta_{S,bounce})^{-1}$ where $S(x, p) = (x, -p)$) and volume preserving, then the composition of $\chi^\delta_{S,bounce}$ and of the accept–reject move satisfies the detailed balance equation. In particular, this step also preserves the target measure $\tilde{\pi}$.

Therefore, it is sufficient to show that $\chi^\delta_{S,bounce} = \Theta^{\delta/2} \circ R_{\text{bounce}} \circ \Theta^{\delta/2}$ is reversible under momentum flip and volume preserving. Note that both Θ^δ and R^δ are flows corresponding to a Hamiltonian system so they must be reversible and volume preserving, see [20, Section 8.2.2 and 8.2.3]. The composition of these operators also has these two properties and including reflection preserves these two properties, therefore R_{bounce} is volume preserving and reversible, and hence so is $\chi^\delta_{S,bounce}$. □

Proposition A.2 *The SOL-HMC-bounce algorithm defined in* Section 2 *is non-reversible.*

Proof of Proposition A.2 For simplicity, we will only consider the case when $N = 1, C = 1$ and $V(x) = 0$. That is, we consider the target measure to be the "truncation of a standard two dimensional Gaussian", namely

$$\hat{\pi}(x, p) = \frac{1}{Z_a} e^{-\frac{1}{2}(x^2+y^2)} \mathbb{1}_{[-a,a]}(x),$$

where Z_a is a normalising constant. In this case, $\Theta^{\delta/2}$ is the identity map, and R^δ can be written as

$$R^\delta(x,\, p) = (x\cos(\delta) + p\sin(\delta),\, p\cos(\delta) - x\sin(\delta)).$$

With these observations, if at time k the chain is $(x^k,\, p^k)$, then we can write the proposed move $(\tilde{x}^{k+1},\, \tilde{p}^{k+1})$ in the $k+1$-th step of SOL-HMC-bounce as

$$(\tilde{x}^{k+1},\, \tilde{p}^{k+1}) = R^\delta_{\text{bounce}}(x^k,\, p^k e^{-\varepsilon} + i\xi)$$

where ξ is drawn from a standard normal distribution. In this case, the acceptance probability is given by

$$\alpha = \min(1,\, e^{-\frac{1}{2}((\tilde{x}^{k+1})^2 + (\tilde{p}^{k+1})^2 - (x^k)^2 - (p^k e^{-\varepsilon} + i\xi)^2)}).$$

Now we wish to calculate the transition kernel, $K((x,\, p),\, (y,\, q))$, for this Markov chain, i.e. find the probability density corresponding to the move from $(x,\, p)$ to $(y,\, q)$.

Observe that R^δ is a rotation about the origin and hence preserves radial distance, that is if $(\tilde{x},\, \tilde{p}) := R^\delta(x,\, p)$, then

$$\tilde{x}^2 + \tilde{p}^2 = x^2 + p^2.$$

Flipping momentum sign, i.e. applying reflections S, also preserves radial distance; therefore, the operator R^δ_{bounce} preserves radial distance. In particular, if $\tilde{x}^2 + \tilde{p}^2 < a^2$ (or equivalently $x^2 + p^2 < a^2$), then $R^\delta(x,\, p)$ must remain in the strip $[-a,\, a] \times \mathbb{R}$, so in this situation $R^\delta_{\text{bounce}}(x,\, p) = R^\delta(x,\, p)$.

Suppose that $y^2 + q^2 \le a^2$. Fix some $x \in [-a,\, a]$, $p \in \mathbb{R}$, then let $(\hat{x},\, \hat{p}) = \mathcal{O}^\varepsilon(x,\, p) = (x,\, pe^{-\varepsilon} + i\xi)$, where ξ is a standard normal random variable. In which case we have that $\hat{p}$ is normally distributed with mean $pe^{-\varepsilon}$ and variance i^2. Set $(y,\, q) = R^\delta(\hat{x},\, \hat{p})$, then

$$(y,\, q) = (\hat{x}\cos(\delta) + \hat{p}\sin(\delta),\, \hat{p}\cos(\delta) - \hat{x}\sin(\delta)).$$

Therefore, y is normally distributed with mean $x\cos(\delta) + pe^{-\varepsilon}\sin(\delta)$ and variance $i^2\sin(\delta)^2$. Once y has been determined, we may solve for q and find

$$q = \frac{y\cos(\delta) - x}{\sin(\delta)}. \tag{A.1}$$

In which case the transition kernel is given by

$$\begin{aligned} K((x,\, p),\, (y,\, q)) \;=\; & \frac{1}{\sqrt{2\pi i^2\sin(\delta)^2}} e^{-\frac{(y - x\cos(\delta) - pe^{-\varepsilon}\sin(\delta))^2}{2i^2\sin(\delta)^2}} \\ & \times \alpha\delta_{\frac{y\cos(\delta) - x}{\sin(\delta)}}(q) \\ & + (1-\alpha)\delta_x(y)\frac{1}{\sqrt{2\pi i^2}} e^{-\frac{(q + pe^{-\varepsilon})^2}{2i^2}} \end{aligned}$$

where α is the acceptance probability and is given by

$$\alpha = \min(1,\, e^{\frac{1}{2}(x^2 + p^2 - y^2 - q^2)}).$$

Now the algorithm is reversible if and only if the detailed balance condition holds, that is

$$\begin{aligned} \hat{\pi}(x,\, p)K((x,\, p),\, (y,\, p)) \;=\; & \hat{\pi}(y,\, q)K((y,\, q),\, (x,\, p)), \\ & \forall x,\, y \in [-a,\, a],\, p,\, q \in \mathbb{R}. \end{aligned} \tag{A.2}$$

To see that this does not hold consider the point $(x,\, p) = (0,\, 0)$ and let $(y,\, q)$ be some point in the ball of radius a. Then by (A.1), we must have $y = q\tan(\delta)$, and the left hand side of (A.2) becomes

$$\begin{aligned} & \hat{\pi}(0,\, 0)K((0,\, 0),\, (q\tan(\delta),\, q)) \\ = \; & \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{2\pi i^2\sin(\delta)^2}} e^{-\frac{(q\tan(\delta))^2}{2i^2\sin(\delta)^2}} \\ & \times \min(1,\, e^{-\frac{1}{2}(q^2 + q^2\tan(\delta)^2)}) > 0. \end{aligned}$$

On the other hand, if we suppose $0 < \delta < \pi/4$, then to move from $(q\tan(\delta),\, q)$ to $(0,\, 0)$ is not possible unless $q = 0$, since (A.1) in this case becomes $q\tan(\delta) = 0$. Therefore, for any $q \neq 0$, the right hand side of (A.2) must be zero, in particular we have that the algorithm is not reversible. □

Appendix 2. Description of reservoir model and simulator

The simulator we use is an *in-house* single-phase simulator working on an unstructured grid with finite volumes spatial discretisation and backward Euler time discretisation, calculating the dynamics of pressures and fluid flows in the subsurface porous media.

To obtain the observed pressure data, a fine grid three-phase model was run in the first place, using Schlumberger Eclipse black oil simulator [1]. The resulting output Eclipse pressures were perturbed by the uncorrelated Gaussian noise, with standard deviation $\sigma_{BHP} = 20$ bar for the well BHP data, and $\sigma_b = 3$ bar for the reservoir (block) pressure data. Altogether 380 measurement points were considered (365 for the BHP, 15 for the reservoir pressure), taken with time step of 6 months. The data errors' covariance matrix C_d is diagonal, with the entries equal to either σ^2_{BHP} or σ^2_b.

In the forward simulation mode, the reservoir properties are fixed, the producing and injecting wells (indexed by w) are controlled by the volumetric flow rates q_w and the output modelled data are the time-dependent pressures at the blocks P_ℓ and the bottom hole pressures at the wells P^{BHP}_w. The equations describing the fluid flow are as follows. First, the volumetric flow rate $Q_{\ell j}$ between the pair of connected blocks $\ell,\, j$ is proportional to the pressure difference between them, which can be regarded as Darcy's law:

$$Q_{\ell j} = T_{\ell j}(P_\ell - P_j - \rho\, g\, h_{\ell j}), \tag{B.1}$$

where ρ is the known liquid density, $h_{\ell j}$ is the known depth difference between the block centers and g is the acceleration due to gravity.

The inflow $q_{w\ell}$ into the perforation of well w in block ℓ is proportional to the difference of the bottom hole pressure (BHP) and the block pressure:

$$q_{w\ell} = J_{w\ell}(P_\ell - P_w^{\mathrm{BHP}} - \rho\, g\, h_w^\ell), \tag{B.2}$$

where h_w^ℓ is the depth difference between the block center and the BHP reference depth. The total inflow into well w is obtained by summing up contributions related to this well; that is,

$$q_w = \sum_\ell q_{w\ell}. \tag{B.3}$$

Finally, the volumetric inflows and outflows for block ℓ are balanced, with the excessive/deficient fluid volume leading to the block pressure change via the following compressibility equation:

$$c_\ell\, V_\ell\, \frac{\partial P_\ell}{\partial t} = \sum_j Q_{j\ell} - \sum_w q_{w\ell}, \tag{B.4}$$

where t denotes time, and the first (second, respectively) summation on the right hand side is taken over all the blocks j connected to the block ℓ (all the wells w perforated in block ℓ, respectively). The compressibility c_ℓ of the block is supposed to be known. The simulated reservoir time spans 12 years.

References

1. Eclipse – Industry Reference Reservoir Simulator, Reference Manual. Version 2015.1
2. Bouchard-Côte, A., Doucet, A., Vollmer, S.J.: The bouncy particle sampler: a non-reversible rejection-free Markov Chain Monte Carlo method submitted (2015)
3. Duncan, A.B., Lelievre, T., Pavliotis, G.A.: Variance reduction using nonreversible Langevin samplers. J. Stat. Phys. **163**(3), 457–491 (2016)
4. Bierkens, J., Bouchard-Cote, A., Doucet, A., Duncan, A.B., Fearnhead, P., Lienart, T., Roberts, G., Vollmer, S.J.: Piecewise deterministic Markov processes for scalable Monte Carlo on restricted domains. arxiv preprint (2018)
5. Lelievre, T., Rousset, M., Stoltze, G.: Free Energy Computations: a Mathematical Perspective. Imperial College Press, London (2010)
6. Ma, Y.-A., Fox, E.B., Chen, T., Wu, L.: Irreversible samplers from jump and continuous Markov processes. Stat Comput, 1–26 (2018)
7. Ottobre, M.: Markov chain Monte Carlo and irreversibility. Reports on Math Phys (2016)
8. Ottobre, M., Pillai, N., Pinski, F., Stuart, A.M.: A function space HMC algorithm with second order Langevin diffusion limit Bernoulli (2016)
9. Ottobre, M., Pillai, N., Spiliopoulos, K.: Optimal scaling of the MALA algorithm with irreversible proposals for Gaussian targets. arXiv:1702.01777
10. Beskos, A., Pinski, F., Sanz-Serna, J.M., Stuart, A.M.: Hybrid Monte Carlo on Hilbert spaces. Stoch. Proc Appl. (2011)
11. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley-Interscience Publication (2000)
12. Horowitz, A.M.: A generalized guided Monte Carlo algorithm. Phys. Lett. B **268**(2), 247–252 (1991)
13. Lolon, E.P., Archer, R.A., Ilk, D., Blasingame, T.A.: New semi-analytical solutions for multilayer reservoirs. SPE 114946
14. Poe, B.D. Jr.., Atwood, W.K., Kohring, J., Brook, K.: Characterization of multilayer reservoir properties using production logs. SPE 101721
15. Popa, C., Popa, A., Cover, A.: Zonal allocation and increased production opportunities using data mining in Kern River . SPE 90266
16. Neal, R.M.: MCMC using Hamiltonian dynamics handbook of Markov chain Monte Carlo (2010)
17. Rey-Bellet, L., Spiliopoulos, K.: Irreversible Langevin samplers and variance reduction: A large deviations approach. Nonlinearity **28**(7), 2081–2103 (2015)
18. Rey-Bellet, L., Spiliopoulos, K.: Variance reduction for irreversible Langevin samplers and diffusion on graphs. Electron. Commun. Probab., 20 (2015)
19. Rodrigues, J.R.P.: Calculating derivatives for automatic history matching. Computational Geosciences (2006)
20. Sanz-Serna, J.M.: Markov Chain Monte Carlo and Numerical Differential Equations. Current Challenges in Stability Issues for Numerical Differential Equations, pp. 39–88. Springer, Cham (2014)
21. Villani, C.: Hypocoercivity. Mem. Amer. Math. Soc., **202** (950) (2009)
22. Yudin, E., Lubnin, A.: Simulation of Multilayer Wells Operating. SPE 149924
23. Zangl, G., Hermann, R.: Waterflood pattern optimization using genetic algorithms with multi-tank material balance. SPE 90259