SIAM Review
Vol. 65, No. 4, pp. 1074–1105

A Benchmark for the Bayesian Inversion of Coefficients in Partial Differential Equations*

David Aristoff†
Wolfgang Bangerth‡

Abstract. Bayesian methods have been widely used in the last two decades to infer statistical properties of spatially variable coefficients in partial differential equations from measurements of the solutions of these equations. Yet, in many cases the number of variables used to parameterize these coefficients is large, and oobtaining meaningful statistics of their probability distributions is difficult using simple sampling methods such as the basic Metropolis–Hastings algorithm—in particular, if the inverse problem is ill-conditioned or ill-posed. As a consequence, many advanced sampling methods have been described in the literature that converge faster than Metropolis–Hastings, for example, by exploiting hierarchies of statistical models or hierarchies of discretizations of the underlying differential equation.

At the same time, it remains difficult for the reader of the literature to quantify the advantages of these algorithms because there is no commonly used benchmark. This paper presents a benchmark Bayesian inverse problem—namely, the determination of a spatially variable coefficient, discretized by 64 values, in a Poisson equation, based on point measurements of the solution—that fills the gap between widely used simple test cases (such as superpositions of Gaussians) and real applications that are difficult to replicate for developers of sampling algorithms. We provide a complete description of the test case and provide an open-source implementation that can serve as the basis for further experiments. We have also computed 2×10^{11} samples, at a cost of some 30 CPU years, of the posterior probability distribution from which we have generated detailed and accurate statistics against which other sampling algorithms can be tested.

Key words. Bayesian inference, inverse problems, partial differential equations, Markov chain Monte Carlo, benchmarking

MSC codes. 65N21, 35R30, 74G75

DOI. 10.1137/21M1399464

Contents

*Received by the editors February 18, 2021; accepted for publication (in revised form) January 9, 2023; published electronically November 7, 2023.
https://doi.org/10.1137/21M1399464

Funding: The work of the second author was partially supported by the National Science Foundation under award OAC-1835673 as part of the Cyberinfrastructure for Sustained Scientific Innovation (CSSI) program; by award DMS-1821210; by award EAR-1925595; and by the Computational Infrastructure in Geodynamics initiative (CIG), through the National Science Foundation under award EAR-1550901 and the University of California at Davis. The work of the first author was supported by the National Science Foundation via awards DMS-1818726 and DMS-2111277.

†Department of Mathematics, Colorado State University, Fort Collins, CO 80523 USA (aristoff@math.colostate.edu).

‡Department of Mathematics, Department of Geosciences, Colorado State University, Fort Collins, CO 80523 USA (bangerth@colostate.edu).

1. Introduction. Inverse problems are parameter estimation problems in which one wants to determine unknown, spatially variable material parameters in a partial differential equation (PDE) based on measurements of the solution. In the deterministic approach, in essence one seeks that set of parameters for which the solution of the PDE would best match the measured values; this approach is widely used in many applications. On the other hand, the Bayesian approach to inverse problems recognizes that all measurements are subject to measurement errors and that models are also inexact; as a consequence, we ought to pose the inverse problem as one that seeks a probability distribution describing how likely it is that parameter values lie in a given interval or set. This generalization of the perspective on inverse problems has long roots, but it first came to the attention of the wider scientific community through a 1987 book by Tarantola [50]. It was later followed by a significantly revised and more accessible version by the same author [51] as well as numerous other books on the subject; we mention [36] as one example, along with [3, 2, 17] for tutorial-style introductions to the topic. The Bayesian approach to inverse problems has been used in a wide variety of inverse applications (too many to mention in detail), including acoustics [12], flow in the Earth's mantle [60], laminar and turbulent flow [13], ice sheet modeling [43], astronomy [15], chemistry [26, 40], and groundwater modeling [34].

From a computational perspective, the primary challenge in Bayesian inverse problems is that after discretizing the spatially variable parameters that one seeks to infer, one generally ends with trying to characterize a finite- but high-dimensional probability distribution $\pi(\theta)$ that describes the relative likelihood of parameters θ. In

particular, we are typically interested in computing the mean and standard deviation of this probability distribution (i.e., which set of parameters $\langle\theta\rangle$ *on average* fits the measured data best, and what we know about its variability given the uncertainty in the measured data). Computing these integral quantities in high-dimensional spaces can only be done through sampling methods such as Markov chain Monte Carlo (MCMC) algorithms. On the other hand, sampling in high-dimensional spaces often suffers from a number of problems: (i) long burn-in times until a chain finally finds the region where the probability distribution $\pi(\theta)$ has values substantially different from zero; (ii) long autocorrelation length scales if $\pi(\theta)$ represents elongated, curved, or "ridged" distributions; (iii) for some inverse problems, multimodality of $\pi(\theta)$ is also a problem that complicates the interpretation of the posterior probability distribution; (iv) in many *ill-posed* inverse problems, parameters have large variances that result in rather slow convergence to reliable and accurate means.

In most high-dimensional applications, the result of these issues is that one needs very large numbers of samples to accurately characterize the desired probability distribution. In the context of inverse problems, this problem is compounded by the fact that the generation of every sample requires the solution of the forward problem—generally, the expensive numerical solution of a PDE. As a consequence, the solution of Bayesian inverse problems is computationally exceptionally expensive.

The community has stepped up to this challenge over the past two decades. Numerous algorithms have been developed to make the sampling process more efficient. Starting from the simplest sampler, the Metropolis–Hastings algorithms with a symmetric proposal distribution [33], ideas to alleviate some of the problems include nonsymmetric proposal distributions [45], delayed rejection [54], nonreversible samplers [21], piecewise deterministic Markov processes [11, 55] including Hamiltonian Monte Carlo [39], adaptive methods [32, 6, 44], randomize-then-optimize methods [9, 8, 7], affine invariant samplers [30, 24, 27], and combinations thereof [16, 31].

Other approaches introduce parallelism (e.g., the differential evolution method and variations [52, 53, 57]), or hierarchies of models (see, for example, the survey by Peherstorfer, Willcox, and Gunzburger [42] and references therein, as well as [46, 23, 14, 58, 25]). Yet other methods exploit the fact that discretizing the underlying PDE gives rise to a natural multilevel hierarchy (see [20] among many others) or that one can use the structure of the discretized PDE for efficient sampling algorithms [56, 37].

Many of these methods are likely vastly faster than the simplest sampling methods that are often used. Yet, the availability of a whole zoo of possible methods and their various possible combinations has also made it difficult to assess which method really should be used if one wants to solve a particular inverse problem, and there is no consensus in the community on this topic. Underlying this lack of consensus is that *there is no widely used benchmark* for Bayesian inverse problems. Most of the papers above demonstrate the qualities of their particular innovation using some small but artificial test cases such as a superposition of Gaussians, and often a more elaborate application that is insufficiently well described and often too complex for others to reproduce. As a consequence, the literature contains few examples of comparisons of algorithms *using test cases that reflect the properties of actual inverse problems.*

Our contribution here seeks to address this lack of widely used benchmarks. In particular:

- We provide a complete description of a benchmark that involves characterizing a posterior probability distribution $\pi(\theta)$ on a 64-dimensional parameter space that results from inverting data for a discretized coefficient in a Poisson equation.

- We explain in detail why this benchmark is at once simple enough to make reproduction by others possible, yet difficult enough to reflect the real challenges one faces when solving Bayesian inverse problems.
- We provide highly accurate statistics for $\pi(\theta)$ that allow others to assess the correctness of their own algorithms and implementations. We also provide a performance profile for a simple Metropolis–Hastings sampler as a baseline against which other methods can be compared.

To make adoption of this benchmark simpler, we also provide an open-source implementation of the benchmark that can be adapted to experimentation using other sampling methods with relative ease.

The remainder of this paper is structured as follows. In section 2, we provide a complete description of the benchmark. In section 3, we then evaluate highly accurate statistics of the probability distribution that solves the benchmark based on 2×10^{11} samples we have computed. Section 4 provides a short discussion of what we hope the benchmark will achieve, along with our conclusions. An appendix presents details of our implementation of the benchmark (Appendix A), discusses a simple one-dimensional benchmark for which one can find solutions in a much cheaper way (Appendix B), and provides some statistical background relevant to section 3 (in Appendix C).

2. The Benchmark for Sampling Algorithms for Inverse Problems.

2.1. Design Criteria. In the design of the benchmark described in this paper, we were guided by the following principle:

> *A good benchmark is neither too simple nor too complicated. It also needs to reflect properties of real-world applications.*

Specifically, we design a benchmark for inferring posterior probability distributions using sampling algorithms that correspond to the Bayesian inversion of coefficients in PDEs—in other words, cases where the relative posterior likelihood is computed by comparing (functionals of) the forward solution of PDEs with (simulations of) measured data.

The literature has many examples of papers that consider sampling algorithms for such problems (see the references in the introduction). However, they are typically tested only on cases that fall into the following two categories:

- Simple posterior probability density functions (PDFs) that are given by explicitly known expressions such as Gaussians or superpositions of Gaussians. There are advantages to such test cases: (i) the probability distributions are cheap to evaluate, and it is consequently possible to create essentially unlimited numbers of samples; (ii) because the PDF is exactly known, exact values for statistics such as the mean, covariances, or maximum likelihood (MAP) points are often computable exactly, facilitating the quantitative assessment of convergence of sampling schemes. On the other hand, these test cases are often so simple that *any* reasonable sampling algorithm converges relatively quickly, making true comparisons between different algorithms difficult. More importantly, however, such simple test cases *do not reflect real-world properties* of inverse problems. Most inverse problems are ill-posed, nonlinear, and high-dimensional. They are often unimodal, but with PDFs that are typically quite insensitive along certain directions in parameter space, reflecting the ill-posedness of the underlying problem. Because real-world problems are so different from simple artificial test cases, it is difficult to draw conclusions

from the performance of a new sampling algorithm when applied to a simple test case.

- Complex applications, such as the determination of the spatially variable oil reservoir permeability from the production history of an oil field, or the determination of seismic wave speeds from travel times of earthquake waves from their source to receivers. Such applications are, of course, the target for applying advanced sampling methods, but they make for poor benchmarks because they are very difficult to replicate by other authors. As a consequence, they are almost exclusively used only in the original paper in which a new sampling algorithm is first described, and it is difficult for others to compare this new sampling algorithm against previous ones, since they have not been tested against the same, realistic benchmark.

We position the benchmark in this paper between these extremes. Specifically, we have set out to achieve the following goals:

- *Reflect real properties of inverse problems:* Our benchmark should reflect properties one would expect from real applications, such as the permeability or seismic wave speed determinations mentioned above. We do not really know what these properties are, but intuition and knowledge of the literature suggest that they include very elongated and nonlinear probability distributions, quite unlike Gaussians or their superpositions. In order for our benchmark to reflect these properties, we base it on a PDE.
- *High-dimensional:* Inverse problems are originally infinite-dimensional, i.e., we seek parameters that are functions of space and/or time. In practice, these need to be discretized, leading to finite- but often high-dimensional problems. It is well understood that the resulting curse of dimensionality leads to practical problems that often make the Bayesian inverse problem extremely expensive to solve. At the same time, we want to reflect these difficulties in our benchmark.
- *Computable with acceptable effort:* A benchmark needs to have a solution that is known to an accuracy that is sufficiently good to compare against. This implies that it can't be so expensive that we can only compute a few thousand or tens of thousands of samples of the posterior probability distribution. This rules out most real applications for which each forward solution, even on parallel computers, may take minutes, hours, or even longer. Rather, we need a problem that can be solved in at most a second on a single processor to allow the generation of a substantial number of samples.
- *Reproducible:* To be usable by anyone, a benchmark needs to be completely specified in all of its details. It also needs to be simple enough so that others can implement it with reasonable effort.
- *Available:* An important component of this paper is that we make the software that implements the benchmark available as open source; see Appendix A. In particular, the code is written in a modular way that allows evaluating the posterior probability density for a given set of parameter values—i.e., the key operation of all sampling methods. The code is also written in such a way that it is easy to use in a multilevel sampling scheme where the forward problem is solved with a hierarchy of successively more accurate approximations.

2.2. Description of the Benchmark. Given the design criteria discussed in the previous subsection, let us now present the details of the benchmark. Specifically, we seek (statistics of) a nonnormalized posterior probability distribution $\pi(\theta|\hat{z})$ on a

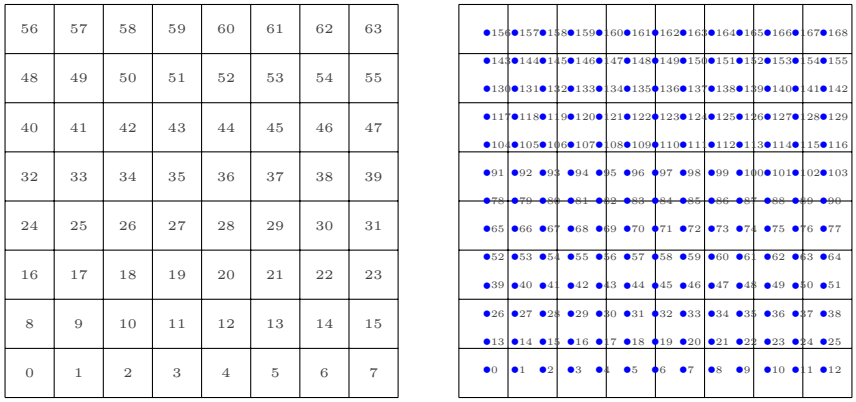


Fig. 1 *Left: Numbering of the* 64 *cells on which the parameters are defined. Right: Numbering and locations of the* $13^2 = 169$ *evaluation points at which the solution is evaluated.*

parameter space $\theta \in \Theta = \mathbb{R}^{64}$ of modestly high dimension 64—large enough to be interesting, while small enough to remain feasible for a benchmark. Here, we think of $\hat{z}$ as a set of measurements made on a physical system that is used to infer information about the internal parameters θ of the system. As is common in Bayesian inverse problems, $\pi(\theta|\hat{z})$ is defined as the product of a likelihood times a prior probability:

$$\pi(\theta|\hat{z}) \propto L(\hat{z}|\theta)\,\pi_{\text{pr}}(\theta). \tag{2.1}$$

Here, $L(z|\theta)$ describes how likely it would be to measure values z if θ were the "true" values of the internal parameters. π_{pr} is a (not necessarily normalized) probability distribution encoding our prior beliefs about the parameters. A complete description of the benchmark then requires us to describe the values of z and ways to evaluate the functions L and π_{pr}. We will split the definition of L into a discussion of the forward model and a statistical model of measurements in the following.

2.2.1. The Forward Model. The setting we want to pursue is as follows: Let us imagine a membrane stretched over a frame that bounds a domain Ω which, for simplicity, we assume to be the unit square $\Omega = (0,1)^2$. The membrane is subject to an external vertical force $f(\mathbf{x})$ which for the purpose of this benchmark we choose as constant $f(\mathbf{x}) = 10$. Furthermore, the membrane has a spatially variable resistance $a(\mathbf{x})$ to deflection (for example, it may have a variable thickness or may be made from different materials). In this benchmark, we assume that $a(\mathbf{x})$ is piecewise constant on a uniform 8×8 grid as shown in Figure 1, with the 64 values that parameterize $a(\mathbf{x})$ given by the elements of the vector $\theta_0, \ldots, \theta_{63}$ as also indicated in the figure. In other words, there is a 1:1 relationship between the vector θ and the piecewise constant coefficient function $a(\mathbf{x}) = a^{\theta}(\mathbf{x})$.

Then, an appropriate model to describe the vertical deflection $u(\mathbf{x})$ of the membrane would express u as the solution of the following PDE that generalizes the Poisson equation:

$$-\nabla \cdot [a(\mathbf{x})\nabla u(\mathbf{x})] = f(\mathbf{x}) \qquad \text{in } \Omega, \tag{2.2}$$

$$u(\mathbf{x}) = 0 \qquad \text{on } \partial\Omega. \tag{2.3}$$

This model is, of course, not exactly solvable, but its solution can be approximated using discretization. The way we define the likelihood L then requires us to specify

exactly how we discretize this model. Concretely, we define $u_h(\mathbf{x})$ as the solution of a finite element discretization of (2.2)–(2.3) using a uniform 32×32 mesh and a Q_1 (bilinear) element. Because $f(\mathbf{x})$ is given, and because there is a 1:1 relationship between θ and $a(\mathbf{x})$, this discretized model then implies that for each θ we can find a $u_h(\mathbf{x}) = u_h^\theta(\mathbf{x})$ that can be thought of as being parameterized using the 1089 degrees of freedom of the Q_1 discretization on the 32×32 mesh. (However, of these 1089 degrees of freedom, 128 are on the boundary and are constrained to zero.) In other words, using the Q_1 shape functions $\varphi_k(\mathbf{x})$, we can express $u_h(\mathbf{x}) = \sum_{k=0}^{1088} U_k \varphi_k(\mathbf{x})$. It is important to stress that the mapping $\theta \mapsto u_h^\theta(\mathbf{x})$ (or, equivalently, $\theta \mapsto U^\theta$) is *nonlinear*.

The function $u_h^\theta(\mathbf{x})$ can be thought of as the predicted displacement at every point $\mathbf{x} \in \Omega$ if θ represented the spatially variable stiffness coefficient of the membrane. In practice, however, we can only measure finitely many things, and consequently we define a measurement operator $\mathcal{M} : u_h \mapsto z \in \mathbb{R}^{169}$ that evaluates u_h on a uniform 13×13 grid of points $\mathbf{x}_k \in \Omega$ so that $\mathbf{x}_k = \left(\frac{i}{13+1}, \frac{j}{13+1}\right), 1 \le i, j \le 13$, with $k = 13(i-1) + (j-1)$. The locations of these points are also indicated in Figure 1. This last step then defines a linear mapping. Because of the equivalence between the function u_h and its nodal vector U, the linearity of the measurement operator implies that we can write $z = MU$ with a matrix $M \in \mathbb{R}^{169 \times 1089}$ that is given by $M_{kl} = \varphi_l(\mathbf{x}_k)$.

In summary, a parameter vector $\theta \in \mathbb{R}^{64}$ then predicts measurements $z^\theta \in \mathbb{R}^{169}$ using the following chain of maps:

$$\theta \mapsto a^\theta(\mathbf{x}) \mapsto U^\theta \mapsto z^\theta. \tag{2.4}$$

The mapping $\theta \mapsto z^\theta$ is commonly called the "forward model" as it *predicts* measurements z^θ if we know the parameter values θ. The "inverse problem" is then, of course, the inverse operation: to *infer* the parameters θ that describe a system based on measurements z of its state u.

All of the steps of the forward model have been precisely defined above and are easily computable with some basic knowledge of finite element methods (or using the code discussed in Appendix A). The expensive step is to solve for the nodal vector U^θ, as this requires the assembly and solution of a linear system of size 1089.

REMARK 2.1. *The 32×32 mesh to define the forward model is chosen sufficiently fine to resolve the exact solution u reasonably well. At the same time, it is coarse enough to allow for the rapid evaluation of the solution—even a rather simple implementation should yield a solution in less than a second, and a highly optimized implementation such as the one discussed in Appendix* A.1 *will be able to do so in less than* 5 *milliseconds on modern hardware. As a consequence, this choice of mesh allows for computing a large number of samples and consequently accurate quantitative comparisons of sampling algorithms.*

We also mention that the 32×32 mesh for $u_h(\mathbf{x})$ is twice more globally refined than the 8×8 mesh used to define $a(\mathbf{x})$ in terms of θ. It is clear to practitioners of finite element discretizations of PDEs that the mesh for u_h must be at the very least as fine as the one for the coefficient a^θ to obtain any kind of accuracy. On the other hand, these choices then leave room for a hierarchy of models in which the forward model uses 8×8, 16×16, and 32×32 meshes; we expect that multilevel sampling methods will use this hierarchy to good effect.

REMARK 2.2. *In our experiments, we will choose the values of θ (and consequently of $a(\mathbf{x})$) clustered around one. With the choice $f = 10$ mentioned above, this leads*

to a solution $u(\mathbf{x})$ with values in the range 0 to 0.95. This then also implies that we should think of the numerical magnitude of our measurements z_k^θ as $\mathcal{O}(1)$.

2.2.2. The Likelihood $L(z|\theta)$. Given the predicted measurements z^θ that correspond to a given set of parameters θ, the likelihood $L(z|\theta)$ can be thought of as expressing the (nonnormalized) probability of actually obtaining z in a measurement if θ is the "correct" set of parameters. This is a statement that encodes the measurement error of our measurement device.

For the purposes of this benchmark, we assume that these measurement errors are independent and identically distributed for all 169 measurement points. More specifically, we define the likelihood as the (nonnormalized) probability function

$$L(z|\theta) = \exp\left(-\frac{\|z - z^\theta\|^2}{2\sigma^2}\right) = \prod_{k=0}^{168} \exp\left(-\frac{(z_k - z_k^\theta)^2}{2\sigma^2}\right), \tag{2.5}$$

where we set $\sigma = 0.05$ and where z^θ is related to θ using the chain (2.4).

REMARK 2.3. *We can think of* (2.5) *as encoding our belief that our measurement system produces a Gaussian-distributed measurement $z_k \sim N(z_k^\theta, \sigma)$. Given that $z_k^\theta = \mathcal{O}(1)$, $\sigma = 0.05$ implies a measurement error of 5%. This is clearly much larger than the accuracy with which one would be able to determine the deflection of a membrane in practice. On the other hand, we have chosen σ this large to ensure that the Bayesian inverse problem does not lead to a probability distribution $\pi(\theta|\hat{z})$ that is so narrowly centered around a value $\bar{\theta}$ that the mapping $\theta \mapsto z^\theta$ can be linearized around $\bar{\theta}$—in which case the likelihood $L(\hat{z}|\theta)$ would become Gaussian, as is also discussed in Appendix* B. *We will demonstrate in section* 3.4 *that $\pi(\theta|\hat{z})$ is indeed not Gaussian and, moreover, it is large along a curved ridge that cannot easily be approximated by a Gaussian either.*

2.2.3. The Prior Probability $\pi_{\text{pr}}(\theta)$. Our next task is to describe our prior beliefs for the values of the parameters. Given that the 64 values of θ describe the stiffness coefficient of a membrane, it is clear that they must be positive. Furthermore, as with many mechanical properties that can have values over vast ranges,[1] reasonable priors are typically posed on the "order of magnitude" (that is, the *logarithm*), not the size of the coefficient itself. We express this through the (nonnormalized) probability distribution

$$\pi_{\text{pr}}(\theta) = \prod_{i=0}^{63} \exp\left(-\frac{(\ln(\theta_i) - \ln(1))^2}{2\sigma_{\text{pr}}^2}\right), \tag{2.6}$$

where we choose $\sigma_{\text{pr}} = 2$. We recognize the prior density of $\ln(\theta_k)$ as a Gaussian with mean σ_{pr}^2 and standard deviation σ_{pr}. The value $\sigma_{\text{pr}} = 2$ was chosen to make values of θ between 10^{-2} and 10^2 reasonably likely, given that the "true" values $\hat{\theta}$ we seek to uncover range between 10^{-1} and 10^1—see section 2.2.4.

Because this prior distribution is posed on the logarithm of the parameters, the prior on the parameters themselves is very heavy-tailed, with mean values $\langle\theta_k\rangle_{\pi_{\text{pr}}}$ for each component much larger than the value at which π_{pr} takes on its maximum (which is at $\theta_k = 1$). Indeed, the mean of each θ_k with respect to π_{pr} is about 403.43.

[1] For example, the Young's modulus that is related to the stiffness of a membrane can range from 0.01 GPa for rubber to 200 GPa for typical steels. Similarly, the permeability of typical oil reservoir rocks can range from 1 to 1000 millidarcys.

We note that this prior probability is quite weak and, in particular, does not assume any (spatial) correlation between parameters as is often the case in inverse problems [56, 49, 36]. The correlations we will observe in our posterior probability (see section 3.3) are therefore a consequence of the likelihood function only.

2.2.4. The "True" Measurements $\hat{z}$. The last piece necessary to describe the complete benchmark is the choice of the "true" measurements $\hat{z}$ that we want to use to infer the statistical properties of the parameters θ. For the purposes of this benchmark, we will use the 169 values for $\hat{z}$ given in Table 1.

In some sense, it does not matter where these values come from—we could have measured them in an actual experiment and used them to infer the coefficients of the system we measured on. On the other hand, for the purposes of a benchmark, it might be interesting to know whether these "true measurements" $\hat{z}$ correspond to a "true set of parameters" $\hat{\theta}$ against which we can compare statistics such as the mean $\langle\theta\rangle$ of the posterior probability $\pi(\theta|\hat{z})$.

Indeed, this is how we generated $\hat{z}$: We chose a set of parameters $\hat{\theta}$ that corresponds to a membrane of uniform stiffness $a(\mathbf{x}) = 1$ except for two inclusions in which $a = 0.1$ and $a = 10$, respectively. This setup is shown in Figure 2.[2] Using $\hat{\theta}$, we then used the series of mappings as shown in (2.4) to compute $\hat{z}$. However, to avoid an inverse crime, we used a 256×256 mesh and a bicubic (Q_3) finite element to compute $\hat{\theta} \mapsto \hat{u}_h \mapsto \hat{z} = \mathcal{M}\hat{u}_h$, rather than the 32×32 mesh and the bilinear (Q_1) element used to define the mapping $\theta \mapsto u_h \mapsto z^\theta = \mathcal{M}u_h$.

As a consequence of this choice of higher accuracy (and higher computational cost), we can in general not expect that there is a set of parameters θ for which the forward model of section 2.2.1 would predict measurements z^θ that are *equal* to $\hat{z}$. Furthermore, the presence of the prior probability π_{pr} in the definition of $\pi(\theta|\hat{z})$ implies that we should not expect that either the mean $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ or the MAP point $\theta_{\text{MAP}} = \arg\max_\theta \pi(\theta|\hat{z})$ is equal or even simply close to the "true" parameters $\hat{\theta}$.

3. Statistical Assessment of $\pi(\theta|\hat{z})$. The previous section provides a concise definition of the nonnormalized posterior probability density $\pi(\theta|\hat{z})$. Given that the mapping $\theta \mapsto z^\theta$ is nonlinear and involves solving a PDE, there is no hope that $\pi(\theta|\hat{z})$ can be expressed as an explicit formula. On the other hand, all statistical properties of $\pi(\theta|\hat{z})$ can of course be obtained by sampling, for example, using algorithms such as the Metropolis–Hastings sampler [33].

In order to provide a useful benchmark, it is necessary that at least some properties of $\pi(\theta|\hat{z})$ are known with sufficient accuracy to allow others to compare the convergence of their sampling algorithms. To this end, we have used a simple Metropolis–Hastings sampler to compute 2×10^{11} samples that characterize $\pi(\theta|\hat{z})$, in the form of $N = 2000$ Markov chains of length $N_L = 10^8$ each. (Details of the sampling algorithm used to obtain these samples are given in Appendix A.2.) Using the program discussed in Appendix A.1, the effort to produce this many samples amounted to

[2] This setup has the accidental downside that both the set of parameters $\hat{\theta}$ and the set of measurement points $\mathbf{x}_k$ at which we evaluate the solution are symmetric about the diagonal of the domain. Since the same is true for our finite element meshes, the exact solution of the benchmark results in a probability distribution that is invariant to permutations of parameters about the diagonal as well, and this is apparent in Figure 5, for example. A better designed benchmark would have avoided this situation, but we only realized the issue after expending several years of CPU time. At the same time, the expected symmetry of values allows for a basic check of the correctness of inversion algorithms: If the inferred mean value $\langle\theta_7\rangle_{\pi(\theta|\hat{z})}$ is not approximately equal to $\langle\theta_{63}\rangle_{\pi(\theta|\hat{z})}$—see the numbering shown in the left panel of Figure 1—then something is wrong.

Table 1 *The "true" measurement values $\hat{z}_k, k = 0, \ldots, 168$, used in the benchmark. The values are also available in the electronic supplemental material and are shown in full double precision accuracy to allow for exact reproduction of the benchmark.*

$\hat{z}_0$	0.060 765 117 622 593 69	$\hat{z}_{60}$	0.623 530 057 415 691 7	$\hat{z}_{120}$	0.514 009 175 452 694 3
	0.096 019 101 208 484 81		0.555 933 270 404 593 5		0.555 933 270 404 596 9
	0.123 885 251 783 858 4		0.467 030 499 447 417 8		0.567 744 369 374 330 4
	0.149 518 411 737 520 1		0.349 980 914 381 1		0.547 825 166 529 545 3
	0.184 159 612 754 978 4		0.196 882 637 462 94		0.489 575 966 490 898 2
	0.217 452 502 826 112 2		0.217 452 502 826 125 3		0.410 964 174 301 917 1
	0.225 099 616 089 869 8		0.412 232 953 784 340 4		0.395 727 260 284 338
	0.219 795 476 900 299 3		0.577 945 241 983 156 6		0.377 894 932 200 473 4
	0.207 469 569 837 092 6		0.685 968 374 925 437 2		0.359 626 827 185 712 4
	0.188 999 647 766 301 6		0.737 310 833 139 606 3		0.219 125 026 894 894 8
$\hat{z}_{10}$	0.163 272 253 215 372 6	$\hat{z}_{70}$	0.745 881 198 317 824 6	$\hat{z}_{130}$	0.163 272 253 215 368 3
	0.127 678 248 003 818 6		0.727 896 802 240 655 9		0.285 039 780 666 332 5
	0.077 118 459 157 893 12		0.690 479 353 535 775 1		0.373 006 008 206 081
	0.096 019 101 208 485 52		0.636 917 645 271 028 8		0.432 532 550 635 420 7
	0.200 058 953 336 798 3		0.567 744 369 374 321 5		0.467 030 499 447 431 5
	0.338 559 259 195 176 6		0.478 473 876 486 586 7		0.478 473 876 486 602 3
	0.393 430 002 464 780 6		0.360 219 063 282 326 2		0.467 712 268 759 904 1
	0.404 022 389 246 154 1		0.203 179 205 473 732 5		0.434 171 688 106 105 5
	0.412 232 953 784 309 2		0.225 099 616 089 881 8		0.388 186 479 011 099
	0.410 048 009 154 555 4		0.410 048 009 154 578 7		0.377 894 932 200 460 2
$\hat{z}_{20}$	0.394 915 163 718 996 8	$\hat{z}_{80}$	0.555 561 595 613 713 7	$\hat{z}_{140}$	0.363 336 256 718 736 4
	0.369 787 326 479 123 2		0.656 123 536 696 093 8		0.346 445 726 190 539 9
	0.334 018 262 359 24		0.711 655 887 807 071 5		0.209 636 232 136 565 5
	0.285 039 780 666 338 2		0.727 896 802 240 657		0.127 678 248 003 814 8
	0.218 426 003 247 867 1		0.712 192 867 867 018 7		0.218 426 003 247 863 4
	0.127 112 115 635 095 7		0.671 218 739 142 872 9		0.282 169 498 339 525 2
	0.123 885 251 783 861 1		0.613 915 777 559 149 2		0.324 831 514 891 553 5
	0.338 559 259 195 181 9		0.547 825 166 529 538 1		0.349 980 914 381 109 7
	0.711 928 516 276 647 5		0.467 712 268 759 903 1		0.360 219 063 282 333 3
	0.817 571 286 175 642 8		0.358 765 491 100 084 8		0.358 765 491 100 079 9
$\hat{z}_{30}$	0.683 625 411 657 810 5	$\hat{z}_{90}$	0.205 073 429 167 591 8	$\hat{z}_{150}$	0.353 438 997 477 926 8
	0.577 945 241 983 115 7		0.219 795 476 900 309 4		0.364 264 009 018 228 3
	0.555 561 595 613 689 7		0.394 915 163 719 015 7		0.359 626 827 185 69
	0.528 518 156 173 671 9		0.528 518 156 173 691 1		0.346 445 726 190 529 5
	0.491 439 702 849 224		0.621 319 720 186 747 1		0.326 072 895 342 464 3
	0.440 936 749 485 328 2		0.674 517 904 909 440 7		0.180 670 595 355 394
	0.373 006 008 206 077 2		0.690 479 353 535 786		0.077 118 459 157 892 44
	0.282 169 498 339 521 4		0.671 218 739 142 878 7		0.127 112 115 635 096 3
	0.161 017 673 385 773 9		0.617 840 828 935 951 4		0.161 017 673 385 775 7
	0.149 518 411 737 525 7		0.545 360 502 723 788 3		0.183 460 041 273 014 4
$\hat{z}_{40}$	0.393 430 002 464 792 9	$\hat{z}_{100}$	0.489 575 966 490 909	$\hat{z}_{160}$	0.196 882 637 462 944 3
	0.817 571 286 175 656 2		0.434 171 688 106 127 8		0.203 179 205 473 735 4
	0.943 915 462 552 765 3		0.353 438 997 477 945 6		0.205 073 429 167 588 5
	0.801 590 411 509 512 8		0.208 322 749 696 134 7		0.208 322 749 696 124 5
	0.685 968 374 925 402 4		0.207 469 569 837 099		0.217 959 990 927 999 8
	0.656 123 536 696 059 9		0.369 787 326 479 136 6		0.219 125 026 894 882 2
	0.621 319 720 186 731 5		0.491 439 702 849 241 2		0.209 636 232 136 555 1
	0.575 361 131 500 004 9		0.575 361 131 500 020 3		0.180 670 595 355 388 7
	0.514 009 175 452 682 3		0.623 530 057 415 701 7	$\hat{z}_{168}$	0.106 796 555 001 001 3
	0.432 532 550 635 416 5		0.636 917 645 271 049 7		
$\hat{z}_{50}$	0.324 831 514 891 548 2	$\hat{z}_{110}$	0.613 915 777 559 157 9		
	0.183 460 041 273 008 6		0.545 360 502 723 793 5		
	0.184 159 612 754 991 7		0.433 660 492 961 285 1		
	0.404 022 389 246 183 2		0.410 964 174 301 931 2		
	0.683 625 411 657 843 9		0.388 186 479 011 124 5		
	0.801 590 411 509 539 6		0.364 264 009 018 259 2		
	0.787 011 956 114 497 7		0.217 959 990 928 014 5		
	0.737 310 833 139 580 8		0.188 999 647 766 301 1		
	0.711 655 887 807 046 3		0.334 018 262 359 246 1		
	0.674 517 904 909 428 3		0.440 936 749 485 338 1		

approximately 30 CPU years on current hardware. On the other hand, we will show below that this many samples are really necessary in order to provide statistics of $\pi(\theta|\hat{z})$ accurately enough to serve as reference values—at least, if one insists on using an algorithm as simple as the Metropolis–Hastings method. In practice, we hope that

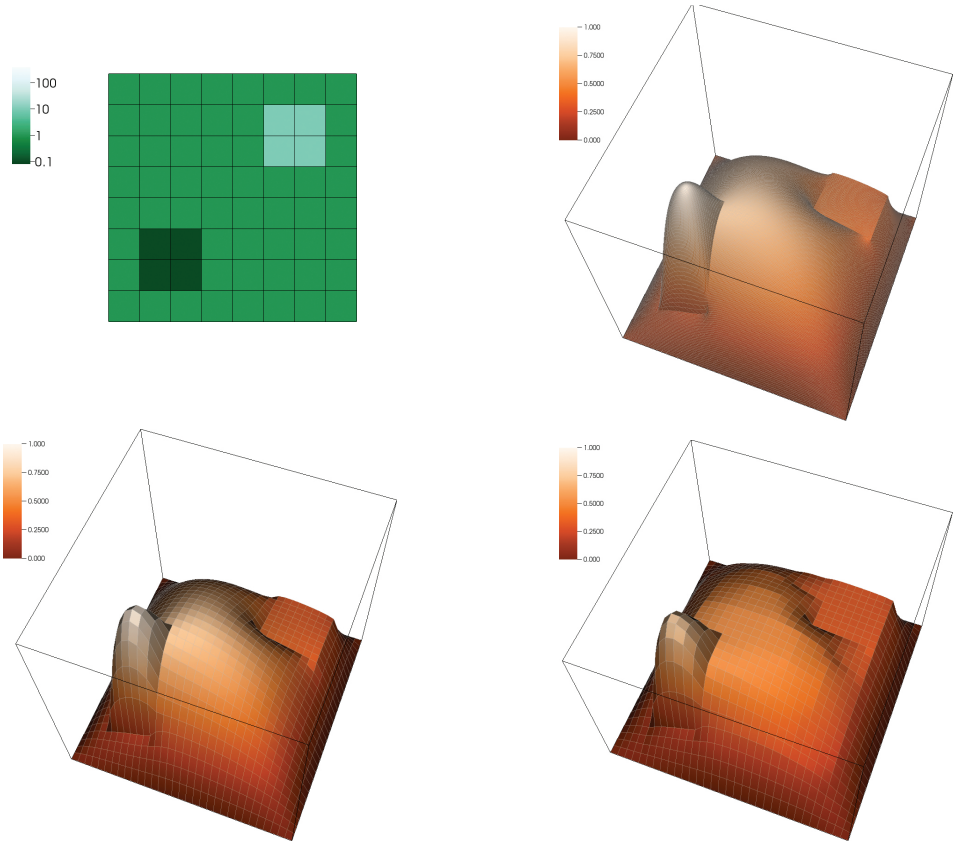


Fig. 2 *Top left: The* 8×8 *grid of values* $\hat{\theta}$ *used in generating the "true measurements"* $\hat{z}$ *via the forward model discussed in section* 2.2.4. *The color scale matches that in Figure* 5. *Top right: The solution of the Poisson equation corresponding to* $\hat{\theta}$ *on the* 256×256 *mesh and using* Q_3 *finite elements. "True" measurements* $\hat{z}$ *are obtained from this solution by evaluation at the points shown in the right panel of Figure* 1. *Bottom left: For comparison, the solution obtained for the same set of parameters* $\hat{\theta}$, *but using the* 32×32 *mesh and the* Q_1 *element that defines the forward model. Bottom right: The solution of this discrete forward model applied to the posterior mean* $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ *that we will compute later; the values of* $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ *are tabulated in Table* 2 *and visualized in Figure* 5.

this benchmark is useful in the development of algorithms that are substantially better than the Metropolis–Hastings method. In addition, when assessing the convergence properties of a sampling algorithm, it is of course not necessary to achieve the same level of accuracy as that obtained here.

In the following we therefore provide a variety of statistics computed from our samples, along with an assessment of the accuracy with which we believe that we can state these results. We will denote by $0 \le L < N = 2000$ the number of the chain, and by $0 \le \ell < N_L = 10^8$ the number of a sample $\theta_{L,\ell}$ on chain L. If we need to indicate one of the 64 components of a sample, we will use a subscript index k as already done in section 2.2.1.

3.1. How Informative Is Our Data Set? While we have $N = 2000$ chains, each with a large number $N_L = 10^8$ of samples per chain, a careful assessment needs to

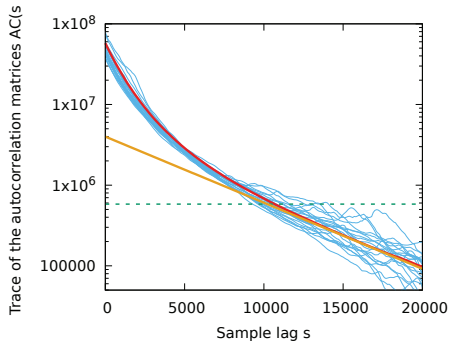


Fig. 3 *Decay of the trace of the autocovariance matrices $AC_L(s)$ with the sample lag s. The light blue curves show the traces of $AC_L(s)$ for twenty of our chains. The red curve shows the trace of the averaged autocovariance matrices, $AC(s) = \frac{1}{N}\sum_{L=0}^{N-1} AC_L(s)$. The thick yellow line corresponds to the function $4 \cdot 10^6 \cdot e^{-s/5300}$ and shows the expected, asymptotically exponential decay of the autocovariance. The dashed green line indicates a reduction of the autocovariance by roughly a factor of 100 compared to its starting values.*

include an evaluation of how informative all of these samples really are. For example, if the samples on each chain have a correlation length of 10^7 because our Metropolis–Hastings sampler converges only very slowly, then each chain really only contains approximately ten statistically independent samples of $\pi(\theta|\hat{z})$. Consequently, we could not expect great accuracy in estimates of the mean value, covariance matrices, and other quantities obtained from each of the chains. Similarly, if the "burn-in" time of the sampler is a substantial fraction of the chain lengths N_L, then we would have to throw away many of the early samples.

To assess these questions, we have computed the autocovariance matrices

$$\begin{aligned} AC_L(s) &= \left\langle [\theta_{L,\ell} - \langle\theta\rangle_L][\theta_{L,\ell-s} - \langle\theta\rangle_L]^T \right\rangle_L \\ &= \frac{1}{N_L - s - 1} \sum_{\ell=s}^{N_L-1} [\theta_{L,\ell} - \langle\theta\rangle_L][\theta_{L,\ell-s} - \langle\theta\rangle_L]^T \end{aligned} \tag{3.1}$$

between samples s apart on chain L. We expect samples with a small lag s to be highly correlated (i.e., $AC_L(s)$ to be a matrix that is large in some sense), whereas for large lags s, samples should be uncorrelated and $AC_L(s)$ should consequently be small. A rule of thumb is that samples at lags s can be considered decorrelated from each other if $AC_L(s) \le 10^{-2} AC_L(0)$ entrywise; see Appendix‘.

Figure 3 shows the trace of these autocovariance matrices for several of our chains. (We only computed the autocovariance at lags $s = 0, 100, 200, 300, \ldots$ up to $s = 20{,}000$ because of the cost of computing $AC_L(s)$.) The curves show that the autocovariances computed from different chains all largely agree, and at least asymptotically they decay roughly exponentially with s, as expected. The data also suggest that the autocorrelation length of our chains is around $N_{AC} = 10^4$—in other words, each of our chains should result in approximately $N_L/N_{AC} = 10^4$ meaningful and statistically independent samples.

To verify this claim, we estimated the *integrated autocovariance* [48] using

$$IAC \approx \frac{1}{N} \sum_{L=0}^{N-1} 100 \sum_{s=-200}^{200} AC_L(|100s|). \tag{3.2}$$

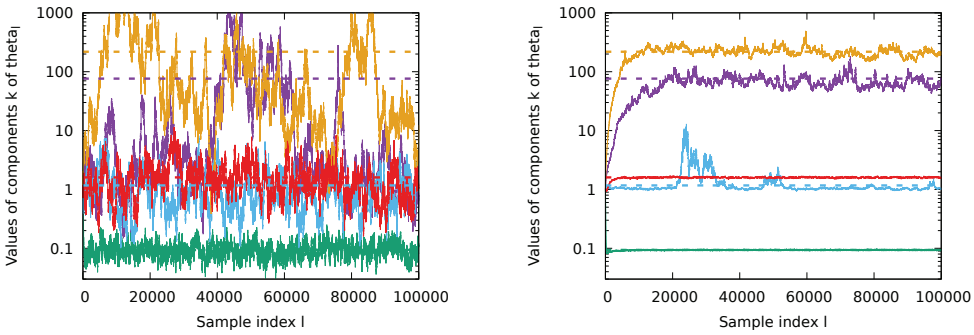


Fig. 4 *Two perspectives on the length of the burn-in phase of our sampling scheme. Left: Values of components $k = 0, 9, 36, 54$, and 63 of samples θ_ℓ for $\ell = 0, \ldots, 100{,}000$, for one randomly chosen chain. (For the geometric locations of the parameters that correspond to these components, see Figure 1.) Right: Across-chain averages $\frac{1}{N}\sum_{L=0}^{N-1} \theta_{L,\ell}$ for the same components k as above. Both images also show the mean values $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ for these five components as dashed lines.*

The integrated autocovariance is obtained by summing up the autocovariance. (The factor of 100 appears because we only computed $AC_L(s)$ at lags that are multiples of 100.) We show in Appendix C that the integrated autocovariance leads to the following estimate of the effective sample size:

$$\text{Effective sample size per chain} \approx \frac{N_L}{\lambda_{\max}(C^{-1} \cdot IAC)} \approx 1.3 \times 10^4, \tag{3.3}$$

where $\lambda_{\max}$ indicates the maximum eigenvalue and $C = \frac{1}{N}\sum_{L=0}^{N-1} AC_L(0)$ is the covariance matrix; see also (3.4). This is in good agreement with Figure 3 and the effective sample size derived from it.

This leaves the question of how long the burn-in period of our sampling scheme is. Figure 4 shows two perspectives on this question. The left panel of the figure shows several components $\theta_{\ell,k}$ of the samples of one of our chains for the first few autocorrelation lengths. The data shows that there is at least no obvious "burn-in" period on this scale that would require us to throw away a substantial part of the chain. At the same time, it also illustrates that the large components of θ are poorly constrained and vary on rather long time scales that make it difficult to assess convergence to the mean. The right panel shows across-chain averages of the ℓth samples, more clearly illustrating that the burn-in period may only last for around 20,000 samples—that is, that only around two of the approximately 10,000 statistically independent samples of each chain are unreliable.

Having thus convinced ourselves that it is safe to use all 10^8 samples from all chains, and that they indeed contain meaningful information, we will next turn our attention toward computing statistical information that characterizes $\pi(\theta|\hat{z})$ and against which other implementations of sampling methods can compare their results.

3.2. The Mean Value of $\pi(\theta|\hat{z})$. The simplest statistic one can compute from samples of a distribution is the mean value. Table 2 shows the 64 values that charac-

Table 2 *Sample means* $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ *for the* 64 *parameters, along with their estimated* 2*-sigma uncertainties.*

$\langle\theta\rangle_0 =$	76.32 ± 0.30	$\langle\theta\rangle_{24} =$	0.881 977 ± 0.000 039	$\langle\theta\rangle_{48} =$	1.085 21 ± 0.000 11
	1.2104 ± 0.0094		0.5828 ± 0.0020		9.386 ± 0.089
	0.977 380 ± 0.000 051		267.72 ± 0.62		12.44 ± 0.12
	0.882 007 ± 0.000 039		369.35 ± 0.64		22.50 ± 0.17
	0.971 859 ± 0.000 048		234.59 ± 0.53		88.57 ± 0.33
	0.947 832 ± 0.000 064		13.29 ± 0.14		283.41 ± 0.57
	1.085 29 ± 0.000 11		22.36 ± 0.16		218.65 ± 0.49
	11.39 ± 0.10		0.988 806 ± 0.000 074		0.933 451 ± 0.000 087
$\langle\theta\rangle_8 =$	1.119 ± 0.011	$\langle\theta\rangle_{32} =$	0.971 900 ± 0.000 049	$\langle\theta\rangle_{56} =$	11.35 ± 0.11
	0.093 721 5 ± 0.000 002 7		0.9509 ± 0.0079		1.081 43 ± 0.000 11
	0.115 799 2 ± 0.000 003 9		30.76 ± 0.19		0.949 869 ± 0.000 074
	0.5815 ± 0.0022		233.93 ± 0.52		0.988 770 ± 0.000 074
	0.9472 ± 0.0079		1.169 ± 0.012		0.987 866 ± 0.000 083
	6.258 ± 0.079		0.8327 ± 0.0057		0.914 247 ± 0.000 077
	9.334 ± 0.090		88.52 ± 0.33		0.933 426 ± 0.000 087
	1.081 51 ± 0.000 11		0.987 809 ± 0.000 079		1.599 84 ± 0.000 30
$\langle\theta\rangle_{16} =$	0.977 449 ± 0.000 052	$\langle\theta\rangle_{40} =$	0.947 816 ± 0.000 065		
	0.115 796 2 ± 0.000 003 8		6.260 ± 0.076		
	0.461 ± 0.020		7.119 ± 0.087		
	267.01 ± 0.55		13.20 ± 0.13		
	30.87 ± 0.19		0.8327 ± 0.0035		
	7.189 ± 0.089		176.73 ± 0.44		
	12.39 ± 0.11		283.38 ± 0.58		
	0.949 863 ± 0.000 073		0.914 212 ± 0.000 077		

terize the mean

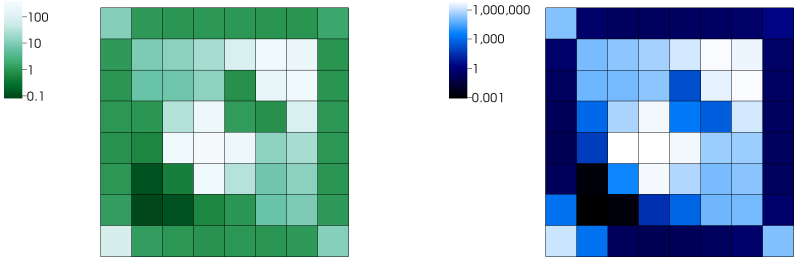
$$\langle\theta_k\rangle_{\pi(\theta|\hat{z})} = \frac{1}{N}\sum_{L=0}^{N-1}\left(\frac{1}{N_L}\sum_{\ell=0}^{N_L-1}\theta_{L,\ell,k}\right).$$

A graphical representation of $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ is shown Figure 5 and can be compared to the "true" values $\hat{\theta}$ shown in Figure 2.[3]

To assess how accurately we know this average, consider that we have $N = 2000$

[3]By comparing Figure 2 and the data of Table 2 and Figure 5, it is clear that for some parameters, the mean $\langle\theta_k\rangle_{\pi(\theta|\hat{z})}$ is far away from the value $\hat{\theta}_k$ used to generate the original data $\hat{z}$—principally those parameters that correspond to large values $\hat{\theta}_k$, but also the "white cross" below and left of center. For the first of these two places, we can first note that the prior probability π_{pr} defined in (2.6) is quite heavy-tailed, with a mean far larger than where its maximum is located. Second, by realizing that a membrane that is locally very stiff is not going to deform in a substantially different way in response to a force from one that is even stiffer in that region—in other words, in areas where the coefficient $a^\theta(\mathbf{x})$ is large, the likelihood function (2.5) is quite insensitive to the exact values of θ, and the posterior probability will be dominated by the prior π_{pr} with its large mean.

For the "white cross," it is plausible that the likelihood is uninformative and that, consequently, mean value and variances are again determined by the prior. To understand why this is so, one could imagine by analogy what would happen if one could measure the solution $u(\mathbf{x})$ of (2.2)–(2.3) exactly and everywhere, instead of only at a discrete set of points. In that case, we would have $u(\mathbf{x}) = z(\mathbf{x})$, and we could infer the coefficient $a(\mathbf{x})$ by solving (2.2)–(2.3) *for the coefficient* instead of for u. This leads to the advection-reaction equation $-\nabla z(\mathbf{x})\cdot\nabla a(\mathbf{x}) - (\Delta z(\mathbf{x}))a(\mathbf{x}) = f(\mathbf{x})$, which is ill-posed and does not provide for a stable solution $a(\mathbf{x})$ at those places where $\nabla z(\mathbf{x}) = \nabla u(\mathbf{x}) \approx 0$. By comparison with Figure 2, we can see that at the location of the white cross, we could not identify the coefficient at one point even if we had measurements available *everywhere*, and not stably so in the vicinity of that point. We can expect that this also holds in the discrete setting of this benchmark—and that consequently, at this location, only the prior provides information.



Inferred mean values $\langle\theta_k\rangle_{\pi(\theta|\hat{z})}$. Variances C_{kk}.

Fig. 5 *Left: Visualization on an 8×8 grid of mean values $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ obtained from our samples. This figure should be compared with the values $\hat{\theta}$ used as input to generate the "true measurements" $\hat{z}$ shown in Figure 2. Right: Variances C_{kk} of the parameters. Dark colors indicate that a parameter is accurately known; light colors that the variance is large. Standard deviations (the square roots of the variances) are larger than the mean values in some cases because of the heavy tails in the distributions of parameters.*

chains of length $N_L = 10^8$ each, and that each of these has its own chain averages

$$\langle\theta_k\rangle_L = \frac{1}{N_L}\sum_{\ell=0}^{N_L-1}\theta_{L,\ell,k}.$$

The ensemble average $\langle\theta_k\rangle_{\pi(\theta|\hat{z})}$ is, of course, the average of the chain averages $\langle\theta_k\rangle_L$ across chains, but the chain averages vary among themselves and we can compute the standard deviation of these chain averages as

$$\operatorname{stddev}\left(\langle\theta_k\rangle_L\right) = \left[\frac{1}{N}\sum_{L=0}^{N-1}\left(\langle\theta_k\rangle_L - \langle\theta_k\rangle_{\pi(\theta|\hat{z})}\right)^2\right]^{1/2}.$$

Under standard assumptions, and assuming that the posterior is Gaussian, we can then estimate that we know the ensemble averages $\langle\theta_k\rangle_{\pi(\theta|\hat{z})}$ to within an accuracy of $\pm\frac{1}{\sqrt{N}}\operatorname{stddev}\left(\langle\theta_k\rangle_L\right)$ with 68% (1-sigma) certainty, and with an accuracy of $\pm\frac{2}{\sqrt{N}}\operatorname{stddev}\left(\langle\theta_k\rangle_L\right)$ with 95% (2-sigma) certainty. In reality, the posterior is *not* Gaussian (see section 3.4), and the argument is not true as stated; however, computing 2-sigma values for all parameters is still a useful metric for how accurately we know each of the parameters.

This 2-sigma accuracy is also provided in Table 2. For all but parameter θ_{18} (for which the relative 2-sigma uncertainty is 4.5%), the relative uncertainty in $\langle\theta\rangle$ is between 0.003% and 1.3%. In other words, the table provides nearly two certain digits for all but one parameter, and four digits for at least half of all parameters.

3.3. The Covariance Matrix of $\pi(\theta|\hat{z})$ and Its Properties. The second statistic we demonstrate is the covariance matrix,

$$C_L = \frac{1}{N_L-1}\sum_{\ell=0}^{N_L-1}\left(\theta_{L,\ell} - \langle\theta\rangle_{\pi(\theta|\hat{z})}\right)\left(\theta_{L,\ell} - \langle\theta\rangle_{\pi(\theta|\hat{z})}\right)^T, \tag{3.4}$$

$$C = \frac{1}{N}\sum_{L=0}^{N-1} C_L.$$

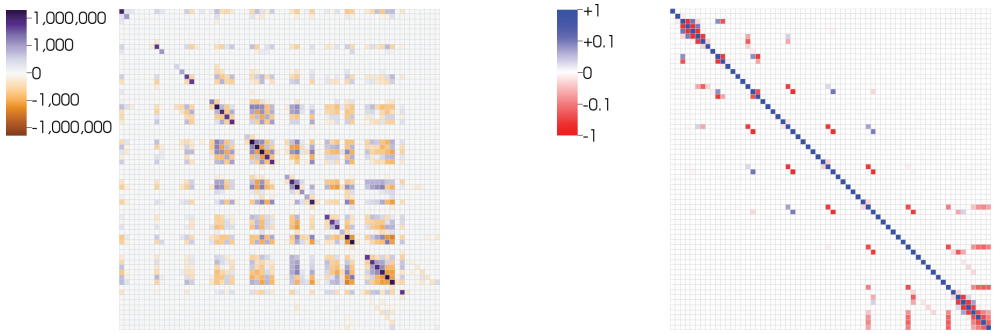


Fig. 6 *Left: A visualization of the covariance matrix C computed from the posterior probability distribution $\pi(\theta|\hat{z})$. The substructure of the matrix in the form of 8×8 tiles represents that geographically neighboring—and consequently correlated—parameters are a distance either ± 1 or ± 8 apart. Right: Correlation matrix $D_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}}\sqrt{C_{jj}}}$.*

While conceptually easy to compute, in practice it is substantially harder to obtain accuracy in C than it is to compute accurate means $\langle\theta\rangle_{\pi(\theta|\hat{z})}$: While we know the latter to two or more digits of accuracy (see Table 2), there is substantial variation between the matrices C_L.[4] The remainder of this section therefore provides only qualitative conclusions we can draw from our estimate of the covariance matrix, rather than quantitative numbers.

First, the diagonal entries of C, C_{kk}, provide the variances of the statistical distribution of θ_k and are shown on the right of Figure 5; the off-diagonal entries $C_{k\ell}$ suggest how *correlated* parameters θ_k and θ_l are and are depicted in Figure 6.

In the context of inverse problems related to PDEs, it is well understood that we expect the parameters to be highly correlated. This can be understood intuitively given that we are thinking of a membrane model: If we increased the stiffness value on one of the 8×8 pixels somewhat, but decreased the stiffness value on a neighboring correspondingly, then we would expect to obtain more or less the same global deformation pattern; maybe there will be small changes at measurement points close to the perturbation, but for measurement points far away the local perturbation will make little difference. As a consequence, we should expect that $L(\hat{z}|\theta) \approx L(\hat{z}|\tilde{\theta})$, where θ and $\tilde{\theta}$ differ only in two nearby components, one component of $\tilde{\theta}$ being slightly larger and the other being slightly smaller than the corresponding component of θ. If the changes are small, then we will also have that $\pi(\theta|\hat{z}) \approx \pi(\tilde{\theta}|\hat{z})$—in other words, we would expect that π is approximately constant in the *secondary diagonal directions* $(0, \ldots, 0, +\varepsilon, 0, \ldots, 0, -\varepsilon, 0, \ldots, 0)$ in θ space.

On the other hand, increasing (or decreasing) the stiffness value in *both* of two adjacent pixels just makes the membrane overall more (or less) stiff and will yield different displacements at *all* measurement locations. Consequently, we expect that the posterior probability distribution $\pi(\theta|\hat{z})$ will vary strongly in the *principal diagonal directions* $(0, \ldots, 0, +\varepsilon, 0, \ldots, 0, +\varepsilon, 0, \ldots, 0)$ in θ space.

[4]For diagonal entries $C_{L,kk}$, the standard deviation of the variation *between* chains is between 0.0024 and 37.7 times the corresponding entry C_{kk} of the average covariance matrix. The variation can be even larger for the many small off-diagonal entries. On the other hand, the average (across chains) difference $\|C_L - C\|_F$ is $0.9\|C\|_F$. This would suggest that we don't know very much about these matrices, but as shown in the rest of the section, *qualitative* measures can be extracted robustly.

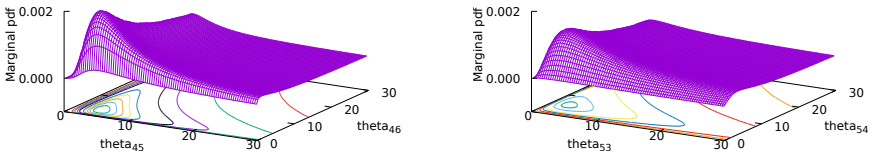


Fig. 7 *Pairwise marginal probabilities for θ_{45} and θ_{46} (left) and for θ_{53} and θ_{54} (right). See Figure* 1 *for the relative locations of these parameters. These marginal distributions illustrate the anticorrelation of parameters: If one is large, the other is most likely small, and vice versa.*

We can illustrate this by computing two-dimensional histograms of the samples for parameters θ_k and θ_l corresponding to neighboring pixels—equivalent to a two-dimensional marginal distribution. We show such histograms in Figure 7. These also indicate that the posterior probability distribution $\pi(\theta|\hat{z})$ is definitely not Gaussian; see also Remark 2.3.

A better way to illustrate correlation is to compute a singular value decomposition of the covariance matrix C. Many inverse problems have only a relatively small number of large singular values of C [56, 22, 12, 60, 43, 13], suggesting that only a finite number of modes is resolvable with the data available—in other words, the problem is ill-posed. Figure 8 shows the singular values of the covariance matrix C for the current case. The data suggests that from the 169 measured pieces of (noisy) data, a deterministic inverse problem could only recover 25–30 modes of the parameter vector $\theta \in \mathbb{R}^{64}$ with reasonable accuracy.[5]

3.4. Higher Moments of $\pi(\theta|\hat{z})$. In some sense, solving Bayesian inverse problems is not very interesting if the posterior distribution $p(\theta|\hat{z})$ for the parameters is Gaussian, or at least approximately so, because it can be done much more efficiently by computing the maximum likelihood estimator through a deterministic inverse problem and then computing the covariance matrix via the Hessian of the deterministic (constrained) optimization problem. For example, [56] provides an excellent overview of the techniques that can be used in this case. Because of these simplifications,

[5]The figure shows the spread of each of the eigenvalues of the within-chain matrices C_L in blue and the eigenvalues of the across-chain matrix C in red. One would expect the latter to be well approximated by the former, and that is true for the largest and smallest eigenvalues, but not for those in the middle. There are two reasons for this. First, each of the C_L is nearly singular, but because each chain is finite, the poorly explored directions are different from one chain to the next. At the same time, it is clear that the sum of (different) singular matrices may actually be "less singular," with fewer small eigenvalues, and this is reflected in the graph. A second reason is that we computed the eigenvalues of each of the C_L and ordered them by size when creating the plot, but without taking into account the associated eigenspaces. As a consequence, if one considers, say, the 32nd largest eigenvalue of C, the figure compares it with the 32nd largest eigenvalues of all of the C_L, when the correct comparison would be with those eigenvalues of the matrices C_L whose eigenspace is most closely aligned; this might be an eigenvalue elsewhere in the order, and the effect will likely be the most pronounced for those eigenvalues whose sizes are the least well constrained.

The conclusions to be drawn from Figure 8 are therefore not the actual sizes of eigenvalues, but the number of "large" eigenvalues. This observation is robust, despite the inaccuracies in our determination of C.

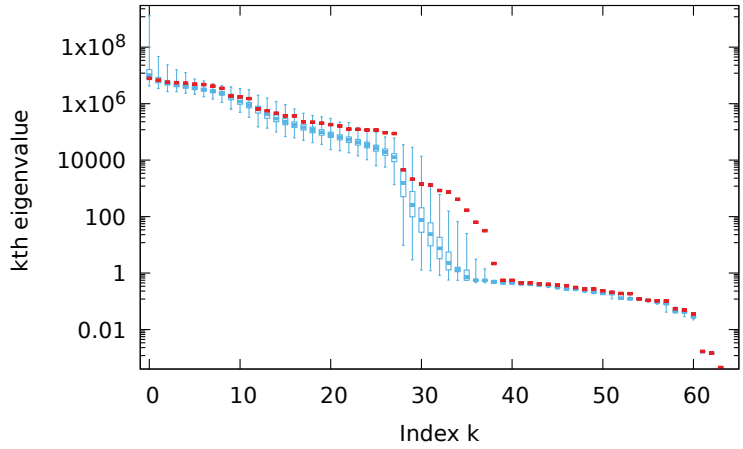


Fig. 8 *Singular values of the covariance matrix C as defined in* (3.4). *Red bars indicate eigenvalues of the across-chain averaged covariance matrix. Blue boxes correspond to the* 25*th and* 75*th percentiles of the corresponding eigenvalues of the covariance matrices C_L of individual chains; vertical bars extend to the minimum and maximum across chains for the kth eigenvalue of the matrices C_L; blue bars in the middle of boxes indicate the median of these eigenvalues.*

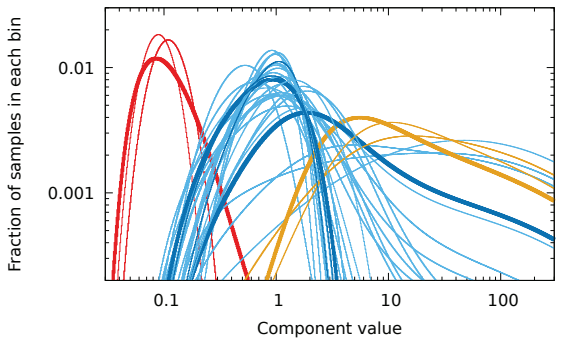


Fig. 9 *Histograms (marginal distributions) for all* 64 *components of θ, accumulated over all 2×10^{11} samples. The histograms for those parameters whose values were* 0.1 *for the purposes of generating the vector $\hat{z}$ are shown in red, those whose values were* 10 *in orange, and all others in blue. The figure highlights histograms for some of those components θ_k whose marginal distributions are clearly neither Gaussian nor log-Gaussian. Note that the histograms were generated with bins whose size increases exponentially from left to right, so that they are depicted with equal size in the figure given the logarithmic θ_k axis. Histograms with equal bin size on a linear scale would be more heavily biased toward the left, but with very long tails to the right. See Figure* 7 *for (pair) histograms using a linear scale.*

it is of interest to know how close the posterior density of this benchmark is to a multidimensional Gaussian.

To evaluate this question, Figure 9 shows histograms of all of the parameters, using 1000 bins that are equally spaced in logarithmic space; i.e., for each component k, we create 1000 bins between -3 and $+3$ and sort samples into these bins based on

$\log_{10}(\theta_k)$. It is clear that many of the parameters have heavy tails and, consequently, cannot be approximated well by Gaussians. On the other hand, given the prior distribution (2.6) that we attached to each of the parameters, it would make sense to conjecture that the *logarithms* $\log(\theta_k)$ might be Gaussian distributed.

If that were so, the double-logarithmic plot shown in the figure would consist of histograms in the form of parabolas open to the bottom, and again a simpler—and presumably cheaper to compute—representation of $\pi(\theta|\hat{z})$ would be possible. However, as the figure shows, this too is clearly not the case: While some parameters seem to be well described by such a parabola, many others have decidedly nonsymmetric histograms or shapes that are simply not parabolic. As a consequence, we conclude that the benchmark is at least not boring in the sense that its posterior distribution could be computed in some comparably much cheaper way.

3.5. Rate of Convergence to the Mean. The data provided in the previous subsections allows for checking whether a separate implementation of this benchmark converges to the same probability distribution $\pi(\theta|\hat{z})$. However, it does not help in assessing whether it does so faster or slower than the simplistic Metropolis–Hastings method used herein. Indeed, as we will outline in the next section, we hope that this work spurs the development and evaluation of methods that can achieve the same results *without* needing more than 10^{11} samples.

To this end, here we provide metrics for how fast our method converges to the mean $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ discussed in section 3.2. More specifically, if we denote by $\langle\theta\rangle_{L,n} = \frac{1}{n}\sum_{\ell=0}^{n-1}\theta_{L,\ell}$ the running mean of samples zero to $n-1$ on chain L, then we are interested in how fast it converges to the mean. We measure this using the following error norm:

$$\begin{aligned} e_L(n) &= \left\| \operatorname{diag}\left(\langle\theta\rangle_{\pi(\theta|\hat{z})}\right)^{-1} \left(\langle\theta\rangle_{L,n} - \langle\theta\rangle_{\pi(\theta|\hat{z})}\right) \right\| \\ &= \left[\left(\langle\theta\rangle_{L,n} - \langle\theta\rangle_{\pi(\theta|\hat{z})}\right)^T \operatorname{diag}\left(\langle\theta\rangle_{\pi(\theta|\hat{z})}\right)^{-2} \left(\langle\theta\rangle_{L,n} - \langle\theta\rangle_{\pi(\theta|\hat{z})}\right) \right]^{1/2}. \end{aligned} \tag{3.5}$$

The weighting by a diagonal matrix containing the inverses of the estimated parameters $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ (given in Table 2 and known to sufficient accuracy) ensures that the large parameters with their large variances do not dominate the value of $e_L(n)$. In other words, $e_L(n)$ corresponds to the "root mean squared relative error."[6]

Figure 10 shows the convergence of a few chains to the ensemble average. While there is substantial variability between chains, it is clear that for each chain, $e_L(n)^2 \to 0$, and, furthermore, that this convergence follows the classic one-over-n convergence of statistical sampling algorithms. Indeed, averaging $e_L(n)^2$ over all chains,

$$e(n)^2 = \frac{1}{N}\sum_{L=0}^{N-1} e_L(n)^2,$$

the behavior of this decay of the "average" square error $e(n)^2$ can be approximated by the following formula that corresponds to the orange line in the figure:

$$e(n)^2 \approx \frac{1.9 \times 10^8}{n}. \tag{3.6}$$

[6] A possibly better choice for the weighting would be to use the inverses of the diagonal entries of the covariance matrix—i.e., the variances of the recovered marginal probability distributions of each parameter. However, these are only approximately known—see the discussion in section 3.3—and consequently do not lend themselves to a concise definition of a reproducible benchmark.

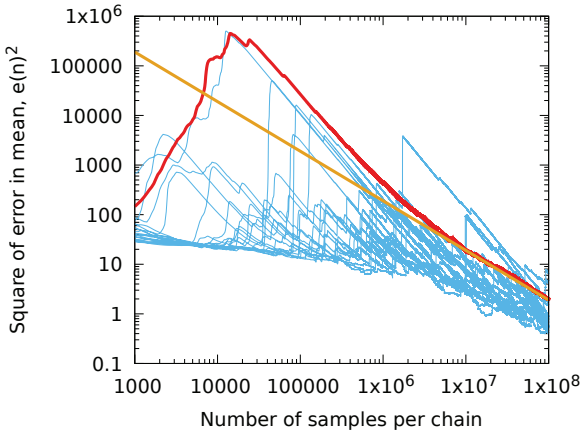


Fig. 10 *Convergence of the square of the relative error $e_L(n)^2$ between the running mean up to sample n and the true mean $\langle\theta\rangle_{\pi(\theta|\hat{z})}$, measured in the weighted norm (3.5), for a randomly chosen subset of chains. The thick red curve corresponds to the average of these squared errors over all chains. This average squared error is dominated by some chains with large errors and so lies above the curves $e_L(n)^2$ of most chains. The thick orange line corresponds to the decay $\frac{1.9\times10^8}{n}$ and represents an approximately average convergence behavior of the chains.*

While we have arrived at the factor 1.9×10^8 by fitting a curve "by eye," it turns out—maybe remarkably—that we can also theoretically support this behavior: using the Markov chain central limit theorem [35] (see Appendix C for details), we can estimate the mean of $ne(n)^2$ as

$$\operatorname{tr}\left(\operatorname{diag}(\langle\theta\rangle_{\pi(\theta|\hat{z})})^{-1} \cdot IAC \cdot \operatorname{diag}(\langle\theta\rangle_{\pi(\theta|\hat{z})})^{-1}\right) \approx 1.9 \times 10^8,$$

where the matrix IAC is defined in (3.2).

If one measures computational effort by how many times an algorithm evaluates the probability distribution $\pi(\theta|\hat{z})$, then n in (3.5) can be interpreted as work units and (3.6) provides an approximate relationship between work and error. Similar relationships can be obtained experimentally for other sampling algorithms that we hope this benchmark will be used for, and (3.6) therefore allows comparison among other algorithms as well as against the one used here.

4. Conclusions and What We Hope This Benchmark Achieves. As the data presented in the previous section illustrates, it is possible to obtain reasonably accurate statistics about the Bayesian solution of the benchmark introduced in section 2, even using a rather simple method: the standard Metropolis–Hastings sampler. At the same time, using this method, it is not at all trivial to compute posterior statistics *accurately*: we had to compute 2×10^{11} samples and expended 30 CPU years on this task (plus another two CPU years on postprocessing the samples).

But all of this also makes for a good benchmark: Simple algorithms, with known performance, *can* solve it to a reasonable accuracy, and more advanced algorithms should be able to do so in a fraction of the time without making the test case trivial. For example, it is not unreasonable to hope that advanced sampling software [19, 1, 41, 38], using multilevel and multifidelity expansions [42, 20, 46, 23] and maybe in conjunction with methods that exploit the structure of the problem to approximate

covariance matrices [56], might be able to reduce the compute time by a factor of 100 to 1000, possibly also running computations in parallel. This would move characterizing the performance of such algorithms for the case at hand to the range of a few hours or days on moderately parallel computers; practical computations might not actually need the same level of accuracy and could be solved even more rapidly.

As a consequence of these considerations, we hope that providing a benchmark that is neither too simple nor too hard, and for which the solution is known to good accuracy, spurs research into the development of better sampling algorithms for Bayesian inverse problems. Many such algorithms of course already exist, but in many cases their performance is not characterized on standardized test cases that allow a fair comparison. In particular, their performance is often characterized using probability distributions whose characteristics have nothing to do with those that result from inverse problems—say, sums of Gaussians. By providing a standardized benchmark that matches what we expect to see in actual inverse problems—along with an open-source implementation of a code that computes the posterior probability function $\pi(\theta|\hat{z})$ (see Appendix A)—we hope that we can contribute to more informed comparisons between newly proposed algorithms; specifically, that their performance can be compared with the relationship shown in (3.6) and Figure 10 to provide a concrete speed-up factor over the method used here.

Appendix A. An Open-Source Code to Sample $\pi(\theta|\hat{z})$. We make a code that implements this benchmark available as part of the "code gallery" for DEAL.II at https://dealii.org/developer/doxygen/deal.II/code_gallery_MCMC_Laplace.html, with the name `MCMC-Laplace` and using the Lesser GNU Public License (LGPL) version 2.1 or later as the license. DEAL.II is a software library that provides the basic tools and building blocks for writing finite element codes that solve PDEs numerically. More information about DEAL.II is available at [4, 5]. The DEAL.II code gallery is a collection of programs based on DEAL.II that were contributed by users as starting points for others' experiments.

The code in question has essentially three parts: (i) The forward solver that, given a set of parameters θ, produces the output z^θ using the map discussed in section 2.2.1; (ii) the statistical model that implements the likelihood $L(z|\theta)$ and the prior probability $\pi_{\text{pr}}(\theta)$ and combines them to obtain the posterior probability $\pi(\theta|\hat{z})$; and (iii) a simple Metropolis–Hastings sampler that draws samples from $\pi(\theta|\hat{z})$. The second of these pieces is quite trivial, encompassing only a couple of functions; we will therefore only comment on the first and third pieces below.

A.1. Details of the Forward Solver. The forward solver is a C++ class whose main function performs the following steps:

1. It takes a 64-dimensional vector θ of parameter values and interprets it as the coefficients that describe a piecewise constant field $a(\mathbf{x})$.
2. It assembles a linear system that corresponds to the finite element discretization of (2.2)–(2.3) using a Q_1 (bilinear) element on a uniformly refined mesh.
3. It solves this linear system to obtain the solution vector U^θ that corresponds to the function $u_h^\theta(\mathbf{x})$.
4. It evaluates the solution u_h^θ at the measurement points $\mathbf{x}_k$ to obtain z^θ.

It then returns z^θ to the caller for evaluation with the statistical model.

Such a code could be written in DEAL.II with barely more than 100 lines of C++ code, and this would have been sufficient for the purpose of evaluating new ideas for sampling methods. However, we wanted to draw on as many samples as possible and consequently decided to see how fast we could make this code.

To this end, we focused on accelerating three of the operations listed above, resulting in a code that can evaluate $\pi(\hat{z}|\theta)$ in 4.5 ms on an Intel Xeon E5-2698 processor with 2.20GHz (on which about half of the samples used in this publication were computed), 3.1 ms on an AMD EPYC 7552 processor with 2.2 GHz (the other half), and 2.7 ms on an Intel Core i7-8850H processor with 2.6 GHz in one of the authors' laptops.

The first part of the code that can be optimized for the current application uses the fact that the linear system that needs to be assembled is the sum of contributions from each of the cells of the mesh. Specifically, the contribution from cell K is

$$A^K = P_K^T A^K_{\text{local}} P_K,$$

where P_K is the restriction from the global set of degrees of freedom to only those degrees of freedom that live on cell K, and A^K_{local} is—for the Q_1 element used here—a 4×4 matrix of the form

$$(A^K_{\text{local}})_{ij} = \int_K a^\theta(\mathbf{x}) \nabla\varphi_i(\mathbf{x}) \cdot \nabla\varphi_j(\mathbf{x}) \,\mathrm{d}x.$$

This suggests that the assembly, including the integration above that is performed via quadrature, has to be repeated every time we consider a new set of parameters θ. However, since we discretize u_h on a mesh that is a strict refinement of that used for the coefficient $a^\theta(\mathbf{x})$, and because $a^\theta(\mathbf{x})$ is piecewise constant, we note that

$$(A^K_{\text{local}})_{ij} = \theta_{k(K)} \underbrace{\int_K \nabla\varphi_i(\mathbf{x}) \cdot \nabla\varphi_j(\mathbf{x}) \,\mathrm{d}x}_{(A_{\text{local}})_{ij}},$$

where $k(K)$ is the index of the element of θ that corresponds to cell K. Here, the matrix A_{local} no longer depends on θ and can, consequently, be computed once and for all at the beginning of the program. Moreover, A_{local} does not actually depend on the cell K as long as all cells have the same shape, as is the case here. We therefore have to store only one such matrix. This approach makes the assembly substantially faster, since we only have to perform the local-to-global operations corresponding to P_K on every cell for every new θ, and no longer any expensive integration/quadrature.

Second, we have experimented with solving the linear systems thus assembled as fast as possible. For the forward solver used for each sample, the size of these linear systems is 1089×1089, with at most 9 entries per row. Following a substantial amount of experimentation, we found that a sparse direct solver is faster than any of the other approaches we tried, and we use the UMFPACK [18] interfaces in DEAL.II for this purpose. In particular, this approach is faster than attempting to use an algebraic multigrid method as a solver or preconditioner for the conjugate gradient (CG) method. We have also tried to use a sparse decomposition via UMFPACK as a preconditioner for the CG method, updating the decomposition only every few samples—based on the assumption that the samples change only relatively slowly and so a decomposition of the matrix for one sample is a good preconditioner for the matrix corresponding to a subsequent sample. However, this turned out to be slower than using a new decomposition for each sample.

The linear solver described above consumes about 90% of the time necessary to evaluate each sample. As a consequence, there is certainly room for further improvements. After all numerical results had been generated for this publication, we

followed up on a suggestion by Martin Kronbichler to replace the linear solver with a CG method preconditioned by an incomplete LU decomposition. This accelerates the computations by about a factor of three, from 2.7 ms to less than 0.9 ms per sample on the Intel Core i7-8850H processor mentioned above. It is this accelerated version that is available at the website mentioned above; we have verified that the new version results in the same results up to at least 11 digits in computing the posterior probability using the techniques mentioned in Appendix A.4.

Finally, evaluating the solution of a finite element field $u_h(\mathbf{x})$ at arbitrary points $\mathbf{x}_k$ is an expensive operation since one has to find which cell K the point belongs to and then transform this point into the reference coordinate system of the cell K. On the other hand, the point evaluation is a linear and bounded operation, and so there must exist a vector m_k such that $u_h(\mathbf{x}_k) = m_k \cdot U$, where U is the vector of coefficients that describe u_h. This vector $m_k = (\varphi_i(\mathbf{x}_k))_{i=0}^{1089-1}$ can be computed once and for all. The computation of $z = (u_h(\mathbf{x}_k))_{k=0}^{169-1}$ can then be facilitated by building a matrix M whose rows are the vectors m_k, and then the evaluation at all measurement points reduces to the operation $z = MU$. M is a sparse matrix with at most four entries per row, making this a very economical approach.

The code with all of these optimizations is not very large—it contains 197 semicolons.[7]

A.2. Details of the Metropolis–Hastings Sampler. The steps described at the start of this appendix yield an algorithm that, given a sample θ, can evaluate $\pi(\theta|\hat{z})$. We use this functionality to drive a Metropolis–Hastings sampler to obtain a large number of samples characterizing the posterior probability distribution.

While the basic algorithm of the Metropolis–Hastings sampler is well known [33, 36], its practical implementation depends crucially on a number of details that we will describe in the following.

First, we start the sampling process with a fixed sample $\theta_0 = (1, \ldots, 1)^T$ corresponding to a coefficient $a^\theta(\mathbf{x}) = 1$.

Second, an important step in the MH algorithm is the generation of a "trial" sample $\tilde{\theta}_\ell$ based on the current sample θ_ℓ. To this end, we use the following strategy. We define the components of $\tilde{\theta}_{\ell,k}$ of $\tilde{\theta}_\ell$ as

$$\tilde{\theta}_{\ell,k} = e^{\ln(\theta_{\ell,k})+\xi_k} = \theta_{\ell,k} e^{\xi_k},$$

where ξ_k, $k = 0, \ldots, 63$, are independent and identically distributed Gaussians with mean 0 and standard deviation σ_{prop}. In other words, the "proposal distribution" for the trial samples is an isotropic Gaussian ball centered at θ_ℓ in log space. This has the effect that all elements of samples always stay positive, as one would expect given that they correspond to material stiffness coefficients. The use of a ball in log space is also consistent with the description of our prior probability distribution in section 2.2.3, which is also defined in log space.

To compute the Metropolis–Hastings acceptance probability, we first need to com-

[7] Counting semicolons is a commonly used metric in C and C++ programs. It roughly coincides with the number of declarations and statements in a program, and is a better metric for *code size* than the number of lines of code, as the latter also includes comments and empty lines used to help the readability of a code.

pute the proposal probability density. By definition

$$\begin{aligned}\mathbb{P}(\tilde{\theta}_{\ell,k} \le \tilde{t}_k | \theta_{\ell,k} = t_k) &= \mathbb{P}(t_k e^{\xi_k} \le \tilde{t}_k) \\ &= \mathbb{P}(\xi_k \le \log \tilde{t}_k - \log t_k) \\ &= \frac{1}{\sqrt{2\pi\sigma_{\text{prop}}^2}} \int_{-\infty}^{\log \tilde{t}_k - \log t_k} \exp\left(\frac{-x^2}{2\sigma_{\text{prop}}^2}\right) dx.\end{aligned}$$

The probability density $p_{\text{prop}}(\tilde{\theta}_{\ell,k}|\theta_{\ell,k})$ of proposing $\tilde{\theta}_{\ell,k}$ given $\theta_{\ell,k}$ is then the derivative of this expression with respect to $\tilde{t}_k$, with $\tilde{\theta}_{\ell,k}$ and $\theta_{\ell,k}$ in place of $\tilde{t}_k$ and t_k:

$$p_{\text{prop}}(\tilde{\theta}_{\ell,k}|\theta_{\ell,k}) = \frac{1}{\tilde{\theta}_{\ell,k}} \frac{1}{\sqrt{2\pi\sigma_{\text{prop}}^2}} \exp\left(\frac{-(\log \tilde{\theta}_{\ell,k} - \log \theta_{\ell,k})^2}{2\sigma_{\text{prop}}^2}\right).$$

By definition, the components $\tilde{\theta}_{\ell,k}$ of the proposal vector $\tilde{\theta}_\ell$ are independent conditional on the current state θ_ℓ. Thus the joint probability density, $p_{\text{prop}}(\tilde{\theta}_\ell|\theta_\ell)$, of proposing vector $\tilde{\theta}_\ell$ given vector θ_ℓ is the product of the probabilities above:

$$p_{\text{prop}}(\tilde{\theta}_\ell|\theta_\ell) = \frac{1}{\prod_{k=0}^{63} \tilde{\theta}_{\ell,k}} \times \frac{1}{(2\pi\sigma_{\text{prop}}^2)^{32}} \exp\left(-\frac{\sum_{k=0}^{63}(\log \tilde{\theta}_{\ell,k} - \log \theta_{\ell,k})^2}{2\sigma_{\text{prop}}^2}\right).$$

The Metropolis–Hastings acceptance probability, $A(\tilde{\theta}_\ell|\theta_\ell)$, to accept proposal $\tilde{\theta}_\ell$ given the current state θ_ℓ, is then

$$\begin{aligned}A(\tilde{\theta}_\ell|\theta_\ell) &= \min\left\{1, \frac{\pi(\tilde{\theta}_\ell|\hat{z})}{\pi(\theta_\ell|\hat{z})} \times \frac{p_{\text{prop}}(\theta_\ell|\tilde{\theta}_\ell)}{p_{\text{prop}}(\tilde{\theta}_\ell|\theta_\ell)}\right\} \\ &= \min\left\{1, \frac{\pi(\tilde{\theta}_\ell|\hat{z})}{\pi(\theta_\ell|\hat{z})} \times \prod_{k=0}^{63} \frac{\tilde{\theta}_{\ell,k}}{\theta_{\ell,k}}\right\}.\end{aligned}$$

As usual, with probability $A(\tilde{\theta}_\ell|\theta_\ell)$ the proposal $\tilde{\theta}_\ell$ is accepted, in which case it becomes the next sample $\theta_{\ell+1} = \tilde{\theta}_\ell$; otherwise the proposal is rejected and we keep the current sample, $\theta_{\ell+1} = \theta_\ell$.

In our experiments, we use $\sigma_{\text{prop}} = 0.0725$, corresponding to changing θ by a factor e^ξ that with 65% probability lies within the range $[e^{-\sigma_{\text{prop}}}, e^{+\sigma_{\text{prop}}}] = [0.93, 1.075]$. This results in an acceptance rate for the Metropolis–Hastings algorithm of just under 24%. This is close to the recommended value of 0.234 for Metropolis–Hastings sampling algorithms that can be derived for specific probability distributions that are generally simpler than the one we are interested in here [28, 47]; the theory guiding the derivation of the 0.234 value may not be applicable here (see [10]), but absent better guidance, we stuck with it.

Finally, all steps in the Metropolis–Hastings algorithms that require a random number use the MT19937 random number generator, as implemented by C++11 compilers in the `std::mt19937` class.

A.3. Implementations of the Benchmark in Alternative Languages. We consider the C++ implementation discussed above as the "reference implementation" of the benchmark. However, we recognize that it is rather heavyweight in the sense that it requires the installation of the DEAL.II library. While this readily facilitates

otherwise nontrivial modifications (e.g., for multilevel sampling schemes that require the solution of the forward problem on coarser meshes), it is clear that for simple experiments, it would be nice to have standalone implementations of the benchmark.

As a consequence, we have also developed MATLAB and Python versions of the benchmark. These are available from the same website where the C++ implementation is available—see the link at the top of the appendix. These alternative implementations provide everything one needs to build a sampler, namely, the functionality to provide an input vector θ and to compute the prior $\pi_{\text{pr}}(\theta)$ and the likelihood $L(\hat{z}|\theta)$ from such an input. Using (2.1), these can then be used to compute the posterior probability $\pi(\theta|\hat{z})$ that forms the basis of most sampling algorithms. The MATLAB version includes a basic Metropolis–Hastings sampler with parallel functionality.

On a recent laptop, the MATLAB version is able to compute the posterior probability for a sample in about 4 ms, not substantially slower than the C++ version we have used for our results. The Python version—although an almost literal transcription of the MATLAB version—requires approximately 25 ms. One imagines that it could be optimized further (for example, a substantial part of the run time is spent in the insertion of the cell-local 4×4 matrices into the global matrix), but we have not attempted to do so.

A.4. Testing of Alternative Implementations. In order to facilitate testing of alternative implementations such as those discussed in the previous section (or testing modifications made to the C++ implementation itself), the website from which the benchmark can be obtained (see the top of this appendix) also contains a set of known input/output pairs. Specifically, it contains files for ten different input vectors θ, along with the corresponding outputs z^θ, likelihood $L(\hat{z}|\theta)$, and prior $\pi_{\text{pr}}(\theta)$ for each input vector. The latter two can be combined via (2.1) into the posterior probability $\pi(\theta|\hat{z})$ associated with the input vector θ.

We have used these known input/output pairs obtained from our reference implementation to verify that the alternative implementations of the benchmark discussed in the previous subsection are correct. For example, the MATLAB implementation provides the vectors z^θ to relative errors on the order of 10^{-13}; log priors and likelihoods are computed to relative errors less than 10^{-11}. The Python version achieves the same level of accuracy.

Appendix B. One-Dimensional Version of the Benchmark. Many of the features of the posterior probability on the 64-dimensional parameter space that we have experimentally observed in section 3 match those that one would expect for inverse problems of the kind discussed herein. In particular, the fact that we have a large spread between the large and small eigenvalues of the covariance matrix, the non-Gaussianity of the probability distribution, and the anticorrelation of parameters defined on neighboring cells did not come as a surprise. Yet, strict proofs for these properties are hard to come by.

At the same time, we can investigate an analogous situation in a one-dimensional model with parameters θ_k, for which one can derive the posterior probability distribution analytically. In this appendix, we work out some details of a one-dimensional version of the benchmark. Consider the generalized Poisson equation

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}(x)\right) = f(x), \qquad 0 < x < 1, \tag{B.1}$$

$$u(x) = 0, \qquad x \in \{0, 1\}. \tag{B.2}$$

Again we assume $a(x) = a^\theta(x)$ is parameterized by $\theta = (\theta_0, \dots, \theta_{N-1})$, where

$$a(x) = \theta_k \qquad \text{if} \quad \frac{k}{N} < x < \frac{k+1}{N} \tag{B.3}$$

and we take $f(x) \equiv 1$. The solution to (B.1) is then of the form

$$u(x) = -\int_0^x \frac{y+c}{a(y)}\, dy, \tag{B.4}$$

where c is a constant determined by requiring $u(1) = 0$. Due to the piecewise constant form of $a = a(x)$, this can be rewritten as

$$u(x) = u_k(x) := -\frac{1}{\theta_k}\left(\frac{x^2}{2} + cx\right) + d_k \qquad \text{if} \quad \frac{k}{N} < x < \frac{k+1}{N}. \tag{B.5}$$

There are $N-1$ continuity constraints and two boundary conditions, namely,

$$u_{k-1}\left(\frac{k}{N}\right) = u_k\left(\frac{k}{N}\right), \qquad k = 1, \dots, N-1, \tag{B.6}$$

$$u_0(0) = 0, \quad u_{N-1}(1) = 0. \tag{B.7}$$

This translates, via (B.5), into $N+1$ linear equations for the $N+1$ coefficients $c, d_0, \dots, d_{N-1}$. The boundary conditions show that

$$d_0 = 0, \qquad c = -\frac{\int_0^1 \frac{y}{a(y)}\, dy}{\int_0^1 \frac{1}{a(y)}\, dy} = -\frac{1}{2N}\frac{\sum_{k=0}^{N-1} \theta_k^{-1}(2k+1)}{\sum_{k=0}^{N-1} \theta_k^{-1}}. \tag{B.8}$$

The remaining $N-1$ equations for $d_1, \dots, d_{N-1}$ can be solved using the continuity constraints, which give the equations

$$d_{k-1} - d_k = \left(\frac{1}{2}\left(\frac{k}{N}\right)^2 + c\frac{k}{N}\right)\left(\theta_{k-1}^{-1} - \theta_k^{-1}\right), \qquad k = 1, \dots, N-1, \tag{B.9}$$

which have the solution

$$d_k = -\sum_{j=1}^{k}\left(\frac{1}{2}\left(\frac{j}{N}\right)^2 + c\frac{j}{N}\right)\left(\theta_{j-1}^{-1} - \theta_j^{-1}\right), \qquad k = 1, \dots, N-1.$$

Let us specifically consider the case with $N = 2$ parameters to understand some qualitative features of the benchmark. In this case, $c = -\frac{1}{4}(3\theta_0 + \theta_1)/(\theta_0 + \theta_1)$ and

$$u_0(x) = -\frac{x^2}{2\theta_0} + \frac{3\theta_0 + \theta_1}{4\theta_0(\theta_0 + \theta_1)}x, \qquad u_1(x) = -\frac{x^2}{2\theta_1} + \frac{3\theta_0 + \theta_1}{4\theta_1(\theta_0 + \theta_1)}x + \frac{\theta_1 - \theta_0}{4\theta_1(\theta_0 + \theta_1)}.$$

This solution is shown in Figure 11.

Let us assume that we have measurements at $x_0 = 0.25$ and $x_1 = 0.75$. Then the "exact" measurements we would get are[8]

$$\hat{z}_0 = -\frac{x_0^2}{2\theta_0} + \frac{3\theta_0 + \theta_1}{4\theta_0(\theta_0 + \theta_1)}x_0, \qquad \hat{z}_1 = -\frac{x_1^2}{2\theta_1} + \frac{3\theta_0 + \theta_1}{4\theta_1(\theta_0 + \theta_1)}x_1 + \frac{\theta_1 - \theta_0}{4\theta_1(\theta_0 + \theta_1)}.$$

[8] Unlike in section 2, these values are computed using the exact solution instead of a finite element approximation.

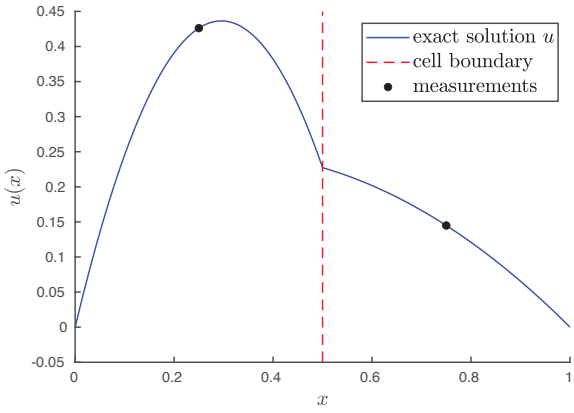


Fig. 11 *Exact solution to* (B.1)–(B.2) *when* $a(x) = 0.1$ *for* $0 < x < 1/2$ *and* $a(x) = 1$ *for* $1/2 < x < 1$, *corresponding to "true" parameters* $\hat{\theta}_0 = 0.1$ *and* $\hat{\theta}_1 = 1$, *with measurements at* $x_0 = 0.25$ *and* $x_1 = 0.75$.

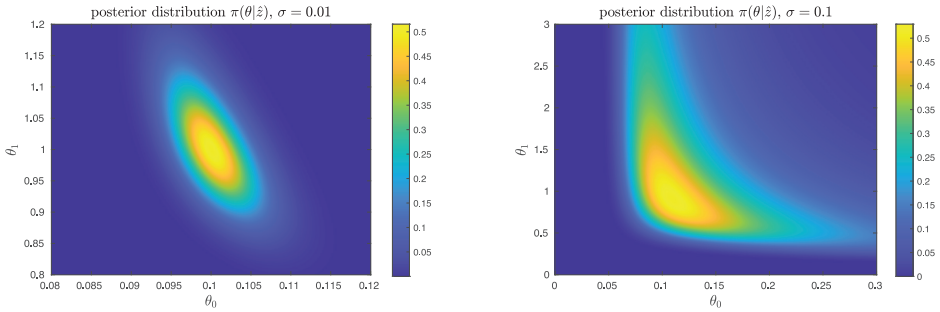


Fig. 12 *Left: Nonnormalized posterior distribution* $\pi(\theta|\hat{z})$ *when* $\sigma = 0.01$ *in the* 1*D model. Notice the distribution is approximately Gaussian for this very small* σ. *Right: Posterior distribution when using* $\sigma = 0.1$ *in the* 1*D model. For this larger value of* σ, *the posterior distribution extends to larger values of* θ_0 *and* θ_1 *and has a non-Gaussian, "banana" shape, reminiscent of the pair histograms in Figure* 7.

We can use these values to exactly and cheaply evaluate the posterior probability (without sampling), and Figure 12 shows $\pi(\theta|\hat{z})$ in this simple case, using "true" parameters $\hat{\theta}_0 = 0.1$ and $\hat{\theta}_1 = 1$ to define $\hat{z}$. We use the prior and likelihoods defined before, with the prior standard deviation $\sigma_{\text{pr}} = 2$ and two different values for the likelihood standard deviations used in (2.5): $\sigma = 0.01$ and $\sigma = 0.1$.

The figure illustrates more concisely the strong correlation between parameters that we experimentally observed in Figure 7. It also illustrates that if σ in the likelihood (2.5) is chosen small (i.e., if the measurement error is small), then the posterior is approximately Gaussian. The observation here therefore validates our choice of a relatively large σ; see also Remark 2.3.

Appendix C. Estimating the Essential Sample Size. Section 3.1 assessed how much information is actually present in the many samples we have computed, and we have also used the results shown there in providing theoretical support for the

key cost-accuracy estimate (3.6). Many of the statistical errors described there can be estimated from the autocovariance $AC_L(s)$ defined in (3.1). The basis for this is the Markov chain central limit theorem. Informally, the Markov chain central limit theorem says that, for large n and N_L, the running means $\langle\theta\rangle_{L,n}$ are approximately normally distributed with mean $\langle\theta\rangle_{\pi(\theta|\hat{z})}$ and covariance

$$\text{Cov}\left(\langle\theta\rangle_{L,n}\right) \approx \frac{IAC}{n}, \tag{C.1}$$

where IAC is the integrated autocovariance obtained by summing up the autocovariance $AC_L(s)$; we estimate IAC from the data in (3.2). Equation (C.1) then justifies the formula for the scaling of $e(n)^2$ below (3.6).

To establish the effective sample size formula (3.3), we cite the "Delta method" in statistics [59]. Informally, the Delta method states that, for large n and N_L, a continuously differentiable function f of the running means is nearly normally distributed with variance

$$\text{Var}(f(\langle\theta\rangle_{L,n})) \approx \frac{1}{n}\nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})^T \cdot IAC \cdot \nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})}), \tag{C.2}$$

provided $\nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})$ is nonzero. The formula (C.2) is obtained by Taylor expanding f and applying the Markov chain central limit theorem. For a Markov chain in which successive samples are all independent, IAC is the covariance matrix, which we estimate from the data in (3.4) and denote by C. Using a standard result on generalized Rayleigh quotients,

$$\frac{\nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})^T \cdot IAC \cdot \nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})}{\nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})^T \cdot C \cdot \nabla f(\langle\theta\rangle_{\pi(\theta|\hat{z})})} \leq \max_{x\neq 0} \frac{x^T \cdot IAC \cdot x}{x^T \cdot C \cdot x} = \lambda_{\max}(C^{-1} \cdot IAC),$$

where $\lambda_{\max}$ denotes the largest eigenvalue. This means that the variance (C.2) is at most $\lambda_{\max}(C^{-1} \cdot IAC)$ times the value it would take if all of the samples on chain L were independent. In other words, the minimum number of "effectively independent samples" is approximately $N_L/\lambda_{\max}(C^{-1} \cdot IAC)$ for each of our chains of length N_L.

It is quite standard to quantify statistical error in MCMC simulations by using the integrated autocovariance [48, 29]. In the literature, it is also common to determine an effective sample size by checking to see where autocovariances cross a certain threshold—such as a small percentage of its initial value—as we did in Figure 3, where we used a threshold of 1%.

Appendix D. Extensions of the Benchmark. A benchmark is only useful if it is concisely described and if the description is sufficiently easy to follow such that others can implement it as well. As a consequence, we have refrained from discussing variations of the benchmark or extending it to a whole family of cases. One can, however, ask what variations would be possible, and how that would affect the difficulty of the benchmark.

First, one could ask how the parameters that define the benchmark's probability distributions affect the difficulty of the benchmark. For example, if one were to substantially reduce the size of σ_{pr} in the definition of the prior probability (2.6), then only a substantially smaller range of of θ values is likely; one would expect that this would increase the size of the misfit $z_k - z_k^\theta$ in (2.5) and that one might want to increase the allowable data uncertainty as described by σ. In any case, a substantially reduced range of likely θ values results in a problem in which the predicted measurements

z^θ become an affine function of θ, and the inverse problem then becomes essentially linear—and the posterior probability distribution becomes essentially Gaussian. Such problems are substantially simpler to solve, because sampling from Gaussian distributions is much easier than sampling from the long-tailed, strongly non-Gaussian distributions we have here (see section 3.4; see also [56]). As a consequence, we have purposely chosen σ_{pr} relatively large in section 2.2.3.

Second, one can of course make the benchmark *much* more difficult to solve by increasing the number of parameters, for example, by using a 16×16 or even finer mesh instead of the 8×8 mesh for $a^\theta(\mathbf{x})$ in section 2.2.1, and concomitantly increasing the number of measurements z^θ from the 169 defined there. Likewise, the computation of z^θ involves the solution of a discretized Poisson equation using a 32×32 finite element mesh using bilinear shape functions; this evaluation of the forward model can of course be made much more expensive by using a finer mesh.

All of these modifications are relatively straightforward to make in the codes discussed in Appendix A. That said, we believe that the choices we have made in the definition of the benchmark provide an appropriate compromise between providing a challenging and realistic test case and allowing for its solution with a reasonable investment of computational resources.

Acknowledgments. W. Bangerth gratefully acknowledges the discussions and early experiments with Kainan Wang many years ago. These early attempts directly led to the ideas that were ultimately encoded in this benchmark. He also appreciates the collaboration with Mantautas Rimkus and Dawson Eliasen on the SampleFlow library that was used for the statistical evaluation of samples. Finally, the feedback given by Noemi Petra, Umberto Villa, and Danny Long is acknowledged with gratitude.

REFERENCES

[1] B. M. Adams et al., *Dakota, a Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version* 6.13 *User's Manual*, Tech. Report SAND2020-12495, Sandia National Laboratories, 2020. (Cited on p. 1093)

[2] O. Aguilar, M. Allmaras, W. Bangerth, and L. Tenorio, *Statistics of parameter estimates: A concrete example*, SIAM Rev., 57 (2015), pp. 131–149, https://doi.org/10.1137/130929230. (Cited on p. 1075)

[3] M. Allmaras, W. Bangerth, J. M. Linhart, J. Polanco, F. Wang, K. Wang, J. Webster, and S. Zedler, *Estimating parameters in physical models through Bayesian inversion: A complete example*, SIAM Rev., 55 (2013), pp. 149–167, https://doi.org/10.1137/100788604. (Cited on p. 1075)

[4] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, *The* `deal.II` *finite element library: Design, features, and insights*, Comput. Math. Appl., 81 (2021), pp. 407–422, https://doi.org/10.1016/j.camwa.2020.02.022, https://arxiv.org/abs/1910.13247. (Cited on p. 1094)

[5] D. Arndt, W. Bangerth, M. Feder, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, S. Sticko, B. Turcksin, and D. Wells, *The* `deal.II` *library, version* 9.4, J. Numer. Math., 30 (2022), pp. 231–246, https://doi.org/10.1515/jnma-2022-0054. (Cited on p. 1094)

[6] Y. F. Atchadé and J. S. Rosenthal, *On adaptive Markov chain Monte Carlo algorithms*, Bernoulli, 11 (2005), pp. 815–828. (Cited on p. 1076)

[7] J. M. Bardsley, *Computational Uncertainty Quantification for Inverse Problems*, SIAM, 2018, https://doi.org/10.1137/1.9781611975383. (Cited on p. 1076)

[8] J. M. Bardsley, A. Seppänen, A. Solonen, H. Haario, and J. Kaipio, *Randomize-then-optimize for sampling and uncertainty quantification in electrical impedance tomography*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 1136–1158, https://doi.org/10.1137/

140978272. (Cited on p. 1076)

[9] J. M. Bardsley, A. Solonen, H. Haario, and M. Laine, *Randomize-then-optimize: A method for sampling from posterior distributions in nonlinear inverse problems*, SIAM J. Sci. Comput., 36 (2014), pp. A1895–A1910, https://doi.org/10.1137/140964023. (Cited on p. 1076)

[10] M. Bédard, *Optimal acceptance rates for Metropolis algorithms: Moving beyond* 0.234, Stochast. Process. Appl., 118 (2008), pp. 2198–2222, https://doi.org/10.1016/j.spa.2007.12.005. (Cited on p. 1097)

[11] J. Bierkens, P. Fearnhead, and G. O. Roberts, *The zig-zag process and super-efficient sampling for Bayesian analysis of big data*, Ann. Statist., 47 (2019), pp. 1288–1320. (Cited on p. 1076)

[12] T. Bui-Thanh and O. Ghattas, *Analysis of the Hessian for inverse scattering problems:* II. *Inverse medium scattering of acoustic waves*, Inverse Problems, 28 (2012), art. 055002, https://doi.org/10.1088/0266-5611/28/5/055002. (Cited on pp. 1075, 1090)

[13] P. Chen, U. Villa, and O. Ghattas, *Taylor approximation and variance reduction for PDE-constrained optimal control under uncertainty*, J. Comput. Phys., 385 (2019), pp. 163–186, https://doi.org/10.1016/j.jcp.2019.01.047. (Cited on pp. 1075, 1090)

[14] J. A. Christen and C. Fox, *Markov chain Monte Carlo using an approximation*, J. Comput. Graphic. Statist., 14 (2005), pp. 795–810, https://doi.org/10.1198/106186005X76983. (Cited on p. 1076)

[15] I. Craig and J. Brown, *Inverse problems in astronomy*, in Bayesian Astrophysics, A. Asensio Ramos and I. Arregui, eds., Cambridge University Press, 1986, pp. 31–61, https://doi.org/10.1017/9781316182406.003. (Cited on p. 1075)

[16] E. Darve, D. Rodríguez-Gómez, and A. Pohorille, *Adaptive biasing force method for scalar and vector free energy calculations*, J. Chem. Phys., 128 (2008), art. 144120. (Cited on p. 1076)

[17] M. Dashti and A. M. Stuart, *The Bayesian Approach to Inverse Problems*, in Handbook of Uncertainty Quantification, R. Ghanem, D. Higdon, and H. Owhadi, eds., Springer, Cham, 2017, pp. 311–428, https://doi.org/10.1007/978-3-319-12385-1_7. (Cited on p. 1075)

[18] T. A. Davis, *Algorithm* 832, ACM Trans. Math. Software, 30 (2004), pp. 196–199, https://doi.org/10.1145/992200.992206. (Cited on p. 1095)

[19] B. Debusschere, K. Sargsyan, C. Safta, and K. Chowdhary, *Uncertainty quantification toolkit (UQTk)*, in Handbook of Uncertainty Quantification, R. Ghanem, D. Higdon, and H. Owhadi, eds., Springer, Cham, 2017, pp. 1807–1827, https://doi.org/10.1007/978-3-319-12385-1_56. (Cited on p. 1093)

[20] T. J. Dodwell, C. Ketelsen, R. Scheichl, and A. L. Teckentrup, *A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 1075–1108, https://doi.org/10.1137/130915005. (Cited on pp. 1076, 1093)

[21] A. B. Duncan, T. Lelievre, and G. A. Pavliotis, *Variance reduction using nonreversible Langevin samplers*, J. Statist. Phys., 163 (2016), pp. 457–491. (Cited on p. 1076)

[22] H. P. Flath, L. C. Wilcox, V. Akçelik, J. Hill, B. van Bloemen Waanders, and O. Ghattas, *Fast algorithms for Bayesian uncertainty quantification in large-scale linear inverse problems based on low-rank partial Hessian approximations*, SIAM J. Sci. Comput., 33 (2011), pp. 407–432, https://doi.org/10.1137/090780717. (Cited on p. 1090)

[23] C. M. Fleeter, G. Geraci, D. E. Schiavazzi, A. M. Kahn, and A. L. Marsden, *Multilevel and multifidelity uncertainty quantification for cardiovascular hemodynamics*, Comput. Methods Appl. Mech. Engrg., 365 (2020), art. 113030, https://doi.org/10.1016/j.cma.2020.113030. (Cited on pp. 1076, 1093)

[24] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, *emcee: The MCMC hammer*, Publ. Astronom. Soc. Pacific, 125 (2013), pp. 306–312. (Cited on p. 1076)

[25] C. Fox and G. Nicholls, *Sampling conductivity images via MCMC*, in Proceedings of the Leeds Annual Statistical Research Workshop (LASR), K. Mardia, C. Gill, and R. Aykroyd, eds., 1997, pp. 91–100. (Cited on p. 1076)

[26] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, *Using Bayesian networks to analyze expression data*, J. Comput. Biol., 7 (2000), pp. 601–620. (Cited on p. 1075)

[27] A. Garbuno-Inigo, N. Nüsken, and S. Reich, *Affine invariant interacting Langevin dynamics for Bayesian inference*, SIAM J. Appl. Dynam. Syst., 19 (2020), pp. 1633–1658, https://doi.org/10.1137/19M1304891. (Cited on p. 1076)

[28] A. Gelman, W. R. Gilks, and G. O. Roberts, *Weak convergence and optimal scaling of random walk Metropolis algorithms*, Ann. Appl. Probab., 7 (1997), pp. 110–120. (Cited on p. 1097)

[29] C. J. Geyer, *Practical Markov chain Monte Carlo*, Statist. Sci., 7 (1992), pp. 473–483. (Cited on p. 1101)
[30] J. Goodman and J. Weare, *Ensemble samplers with affine invariance*, Commun. Appl. Math. Comput. Sci., 5 (2010), pp. 65–80. (Cited on p. 1076)
[31] H. Haario, M. Laine, A. Mira, and E. Saksman, *DRAM: Efficient adaptive MCMC*, Statist. Comput., 16 (2006), pp. 339–354. (Cited on p. 1076)
[32] H. Haario, E. Saksman, and J. Tamminen, *An adaptive Metropolis algorithm*, Bernoulli, 7 (2001), pp. 223–242, https://doi.org/bj/1080222083. (Cited on p. 1076)
[33] W. K. Hastings, *Monte Carlo sampling methods using Markov chains and their applications*, Biometrika, 57 (1970), pp. 97–109. (Cited on pp. 1076, 1082, 1096)
[34] Y. Jiang and A. D. Woodbury, *A full-Bayesian approach to the inverse problem for steady-state groundwater flow and heat transport*, Geophys. J. Internat., 167 (2006), pp. 1501–1512, https://doi.org/10.1111/j.1365-246x.2006.03145.x. (Cited on p. 1075)
[35] G. L. Jones, *On the Markov chain central limit theorem*, Probab. Surv., 1 (2004), pp. 299–320. (Cited on p. 1093)
[36] J. Kaipio and E. Somersalo, *Statistical and Computational Inverse Problems*, Springer-Verlag, 2005, https://doi.org/10.1007/b138659. (Cited on pp. 1075, 1082, 1096)
[37] J. Martin, L. C. Wilcox, C. Burstedde, and O. Ghattas, *A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion*, SIAM J. Sci. Comput., 34 (2012), pp. A1460–A1487, https://doi.org/10.1137/110845598. (Cited on p. 1076)
[38] D. McDougall, N. Malaya, and R. D. Moser, *The parallel C++ statistical library for Bayesian inference: QUESO*, in Handbook of Uncertainty Quantification, R. Ghanem, D. Higdon, and H. Owhadi, eds., Springer, Cham, 2017, pp. 1829–1865, https://doi.org/10.1007/978-3-319-12385-1_57. (Cited on p. 1093)
[39] R. Neal, *MCMC using Hamiltonian dynamics*, in Handbook of Markov Chain Monte Carlo, S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, eds., CRC Press, 2011, pp. 113–162. (Cited on p. 1076)
[40] R. M. Neal, *Bayesian learning via stochastic dynamics*, in Advances in Neural Information Processing Systems, S. Hanson, J. Cowan, and C. Giles, eds., 1992, pp. 475–482. (Cited on p. 1075)
[41] M. Parno, A. Davis, P. Conrad, and Y. M. Marzouk, *MIT Uncertainty Quantification (MUQ) Library*, http://muq.mit.edu/, 2021. (Cited on p. 1093)
[42] B. Peherstorfer, K. Willcox, and M. Gunzburger, *Survey of multifidelity methods in uncertainty propagation, inference, and optimization*, SIAM Rev., 60 (2018), pp. 550–591, https://doi.org/10.1137/16m1082469. (Cited on pp. 1076, 1093)
[43] N. Petra, J. Martin, G. Stadler, and O. Ghattas, *A computational framework for infinite-dimensional Bayesian inverse problems, Part* II*: Stochastic Newton MCMC with application to ice sheet flow inverse problems*, SIAM J. Sci. Comput., 36 (2014), pp. A1525–A1555, https://doi.org/10.1137/130934805. (Cited on pp. 1075, 1090)
[44] G. O. Roberts and J. S. Rosenthal, *Examples of adaptive MCMC*, J. Comput. Graphic. Statist., 18 (2009), pp. 349–367. (Cited on p. 1076)
[45] G. O. Roberts and R. L. Tweedie, *Exponential convergence of Langevin distributions and their discrete approximations*, Bernoulli, 2 (1996), pp. 341–363. (Cited on p. 1076)
[46] J. Seo, C. Fleeter, A. M. Kahn, A. L. Marsden, and D. E. Schiavazzi, *Multi-fidelity estimators for coronary artery circulation models under clinically-informed data uncertainty*, Int. J. Uncertain. Quantif., 10 (2020), pp. 449–466, https://doi.org/10.1615/int.j.uncertaintyquantification.2020033068. (Cited on pp. 1076, 1093)
[47] C. Sherlock and G. O. Roberts, *Optimal scaling of the random walk Metropolis on elliptically symmetric unimodal targets*, Bernoulli, 15 (2009), pp. 774–798, https://doi.org/10.3150/08-bej176. (Cited on p. 1097)
[48] A. Sokal, *Monte Carlo methods in statistical mechanics: Foundations and new algorithms*, in Functional Integration, C. DeWitt-Morette, P. Cartier, and A. Folacci, eds., Springer, 1997, pp. 131–192. (Cited on pp. 1085, 1101)
[49] A. M. Stuart, *Inverse problems: A Bayesian perspective*, Acta Numer., 19 (2010), pp. 451–559, https://doi.org/10.1017/s0962492910000061. (Cited on p. 1082)
[50] A. Tarantola, *Inverse Problem Theory*, Elsevier, Amsterdam, New York, 1987. (Cited on p. 1075)
[51] A. Tarantola, *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, 2005, https://doi.org/10.1137/1.9780898717921. (Cited on p. 1075)
[52] C. J. F. ter Braak, *A Markov chain Monte Carlo version of the genetic algorithm differential evolution: Easy Bayesian computing for real parameter spaces*, Statist. Comput., 16 (2006),

pp. 239–249, https://doi.org/10.1007/s11222-006-8769-1. (Cited on p. 1076)

[53] C. J. F. ter Braak and J. A. Vrugt, *Differential Evolution Markov Chain with snooker updater and fewer chains*, Statist. Comput., 18 (2008), pp. 435–446, https://doi.org/10.1007/s11222-008-9104-9. (Cited on p. 1076)

[54] L. Tierney and A. Mira, *Some adaptive Monte Carlo methods for Bayesian inference*, Statist. Medicine, 18 (1999), pp. 2507–2515. (Cited on p. 1076)

[55] P. Vanetti, *Piecewise-Deterministic Markov Chain Monte Carlo*, Ph.D. thesis, University of Oxford, 2019. (Cited on p. 1076)

[56] U. Villa, N. Petra, and O. Ghattas, *hIPPYlib: An extensible software framework for large-scale inverse problems governed by PDEs. Part* I*: Deterministic inversion and linearized Bayesian inference*, ACM Trans. Math. Softw., 47 (2021), pp. 1–34, https://doi.org/10.1145/3428447. (Cited on pp. 1076, 1082, 1090, 1094, 1102)

[57] J. A. Vrugt, C. J. F. ter Braak, C. G. H. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon, *Accelerating Markov Chain Monte Carlo simulation by differential evolution with self-adaptive randomized subspace sampling*, Internat. J. Nonlinear Sci. Numer. Simul., 10 (2009), pp. 273–290, https://doi.org/10.1515/ijnsns.2009.10.3.273. (Cited on p. 1076)

[58] D. Watzenig and C. Fox, *A review of statistical modelling and inference for electrical capacitance tomography*, Measurement Sci. Technol., 20 (2009), art. 052002, https://doi.org/10.1088/0957-0233/20/5/052002. (Cited on p. 1076)

[59] K. Wolter, *Introduction to Variance Estimation*, Springer Science & Business Media, 2007. (Cited on p. 1101)

[60] J. Worthen, G. Stadler, N. Petra, M. Gurnis, and O. Ghattas, *Towards adjoint-based inversion for rheological parameters in nonlinear viscous mantle flow*, Phys. Earth Planetary Interiors, 234 (2014), pp. 23–34, https://doi.org/10.1016/j.pepi.2014.06.006. (Cited on pp. 1075, 1090)