Simona Bérešová

Numerical realization of the Bayesian inversion accelerated using surrogate models

Persistent URL: http://dml.cz/dmlcz/703185

# NUMERICAL REALIZATION OF THE BAYESIAN INVERSION ACCELERATED USING SURROGATE MODELS

Simona Bérešová[1,2]

[1] Institute of Geonics, Czech Academy of Sciences
Studentská 1768, Ostrava, Czech republic
simona.beresova@ugn.cas.cz
[2] Department of Applied Mathematics, FEECS, VŠB-TUO
17. listopadu 2172, Ostrava, Czech republic

**Abstract:** The Bayesian inversion is a natural approach to the solution of inverse problems based on uncertain observed data. The result of such an inverse problem is the posterior distribution of unknown parameters. This paper deals with the numerical realization of the Bayesian inversion focusing on problems governed by computationally expensive forward models such as numerical solutions of partial differential equations. Samples from the posterior distribution are generated using the Markov chain Monte Carlo (MCMC) methods accelerated with surrogate models. A surrogate model is understood as an approximation of the forward model which should be computationally much cheaper. The target distribution is not fully replaced by its approximation; therefore, samples from the exact posterior distribution are provided. In addition, non-intrusive surrogate models can be updated during the sampling process resulting in an adaptive MCMC method. The use of the surrogate models significantly reduces the number of evaluations of the forward model needed for a reliable description of the posterior distribution. Described sampling procedures are implemented in the form of a Python package.

**Keywords:** Bayesian inversion, delayed-acceptance Metropolis-Hastings, Markov chain Monte Carlo, surrogate model

**MSC:** 65C40, 62F15, 35R30

## 1. Introduction

This contribution focuses on the acceleration of sampling methods in the Bayesian inversion using surrogate models and describes the resulting Python package created within author's PhD studies. The motivation for the development of the package was the solution of inverse problems from the field of geosciences. The underlying mathematical models are usually based on computationally expensive numerical solutions of boundary value problems and the observed data are corrupted with noise.

This contribution provides an analysis of implemented sampling methods. In order to carry out thorough numerical experiments, a computationally cheap model problem is considered. Applications to geoengineering problems can be found in previous publications [2, 4].

Section 2 outlines the principle of the Bayesian inversion that provides the probability distribution of the unknown parameters (called posterior distribution). Section 3 describes methods used to provide samples from the posterior distribution, focusing on the acceleration using surrogate models. Section 4 describes the Python package and its usage. Section 5 discusses the efficiency of the sampling process, the discussion is supported by numerical experiments.

## 2. Problem setting

Consider a mathematical model $G: \mathbb{R}^n \to \mathbb{R}^m$. The aim is to find a probabilistic description of input parameters to the model corresponding to a given vector of noisy outputs $y \in \mathbb{R}^m$.

Further consider a probability space $(\Omega, \mathcal{F}_\Omega, \mathbb{P})$ and measurable spaces $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$, $(\mathbb{R}^m, \mathcal{B}(\mathbb{R}^m))$, $n, m \in \mathbb{N}$. We work with three (multivariate) random variables: unknown parameters $U: \Omega \to \mathbb{R}^n$, observed data $Y: \Omega \to \mathbb{R}^m$, observational noise $Z: \Omega \to \mathbb{R}^m$. Their relationship is described by the noise model $Y = G(U) + Z$. Therefore, $y$ is a realization of $Y$.

The probability distribution of $U$ is called the prior distribution, $f_U$ denotes its probability density function (pdf). It expresses the information about the unknown input parameters known from experience (i.e. without the knowledge of $y$). Similarly, $f_Z$ denotes the pdf of the noise distribution (i.e. probability distribution of $Z$). Now, the aim can be retold in the Bayesian way: We would like to obtain the conditional distribution of $U$ given $Y = y$, called posterior distribution. We also say that we refine the prior distribution using the observed data $y$.

According to the Bayes' theorem, the pdf of the posterior distribution is given by the formula

$$f_{U|Y}(u|y) = \frac{f_Z(y - G(u)) f_U(u)}{\int_{\mathbb{R}^n} f_Z(y - G(v)) f_U(v) \, \mathrm{d}v} \propto \underbrace{f_Z(y - G(u))}_{\text{likelihood}} \underbrace{f_U(u)}_{\text{prior}},$$

where $\propto$ denotes proportionality, $f_Z(y - G(u))$ (as a function of $u$) is called the data likelihood.

The objective is to generate samples from the posterior distribution, see Section 3. Figure 1 illustrates the principle of the Bayesian inversion on a sample observational operator $G: \mathbb{R}^2 \to \mathbb{R}$. This model problem will be also considered in numerical experiments in Section 5.1.

## 3. Markov chain Monte Carlo (MCMC) methods for posterior sampling

MCMC methods serve to generate samples from a target probability distribution. Here, the target distribution is given by the posterior pdf which is understood as
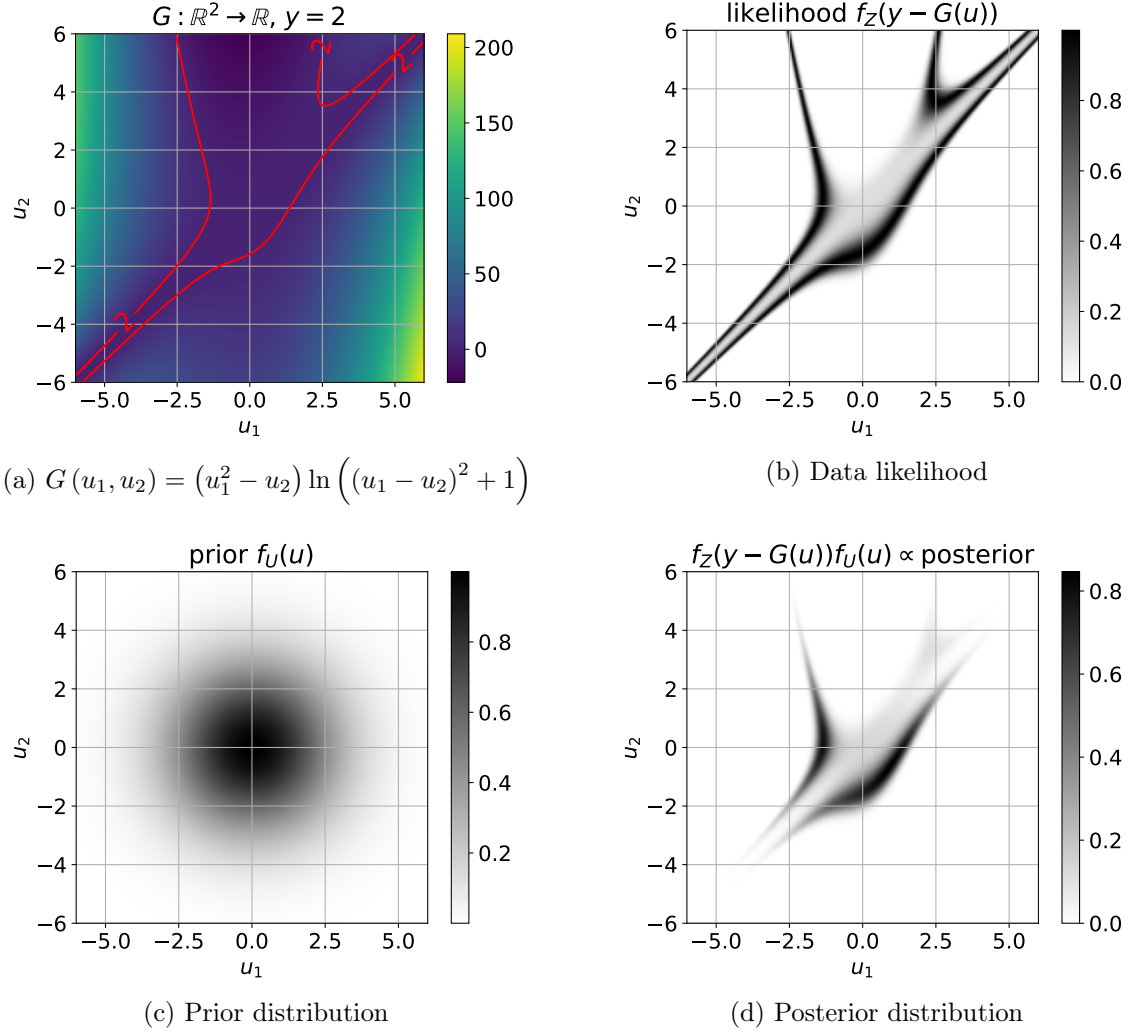
(a) $G\left(u_1, u_2\right) = \left(u_1^2 - u_2\right) \ln\left(\left(u_1 - u_2\right)^2 + 1\right)$

(b) Data likelihood

(c) Prior distribution

(d) Posterior distribution

Figure 1: Illustration of Bayesian inversion

a function of $u$ ($y$ is fixed). The target pdf $f$ can be written as

$$f(u) \propto f_Z(y - G(u)) f_U(u);\tag{1}$$

$\propto$ denotes equality up to a multiplicative constant. We consider the Metropolis-Hastings (MH) algorithm and its modification called the delayed-acceptance MH (DAMH) algorithm, see [6, 7]. In recent years, the DAMH algorithm has been widely used and developed, see e.g. [8, 10].

The MH algorithm (Alg. 1) assumes a symmetric proposal distribution such as the Gaussian random walk. In that case,

$$q(\cdot|u) \text{ is the pdf of } \mathcal{N}_n(u; C),\tag{2}$$

$u \in \mathbb{R}^n$ denotes the mean vector and $C \in \mathbb{R}^{n \times n}$ the covariance matrix.

---
**Algorithm 1** Metropolis-Hastings (MH) algorithm
---
Choose an initial sample $u^{(0)} \in \mathbb{R}^n$ such that $f_Z\left(y - G\left(u^{(0)}\right)\right) f_U\left(u^{(0)}\right) > 0$.
For $k = 0, 1, ...$

1. Propose a sample $v$ from the proposal distribution with pdf $q\left(\cdot | u^{(k)}\right)$.

2. Accept $v$ with probability $a\left(u^{(k)}, v\right) = \min\left\{1, \frac{f_Z(y - G(v)) f_U(v)}{f_Z\left(y - G\left(u^{(k)}\right)\right) f_U\left(u^{(k)}\right)}\right\}$ (i.e., set $u^{(k+1)} = v$). Otherwise reject $v$ (i.e., set $u^{(k+1)} = u^{(k)}$).

---

It can be noticed that the observational operator $G$ is evaluated for each proposed sample $v$. Therefore, if $G$ is computationally expensive, the MH algorithm is not suitable. Higher sampling efficiency can be achieved by the DAMH algorithm in combination with the use of a surrogate model $\widetilde{G}$ that approximates $G$. Evaluations of a suitable surrogate model should be much cheaper compared to the evaluations of $G$. The DAMH algorithm, as introduced in [6], works with the true posterior and also with its approximation. The key property of this algorithm is that it provides samples from the true posterior, the approximation serves only for the acceleration.

Using a surrogate model, the approximation of the posterior pdf can be obtained simply as

$$\widetilde{f}(u) \propto f_Z\left(y - \widetilde{G}(u)\right) f_U(u) \tag{3}$$

The application of the DAMH algorithm to the target distribution in the form of (1) and its approximation (3) leads to Alg. 2.

---
**Algorithm 2** DAMH algorithm using a surrogate model; symmetric proposal pdf
---
Choose an initial sample $u^{(0)} \in \mathbb{R}^n$ such that $f_Z\left(y - G\left(u^{(0)}\right)\right) f_U\left(u^{(0)}\right) > 0$.
For $k = 0, 1, ...$

1. Propose a sample $v$ from the proposal distribution with pdf $q\left(\cdot | u^{(k)}\right)$.

2. Pre-accept $v$ with probability $\widetilde{a}\left(u^{(k)}, v\right) = \min\left\{1, \frac{f_Z\left(y - \widetilde{G}(v)\right) f_U(v)}{f_Z\left(y - \widetilde{G}\left(u^{(k)}\right)\right) f_U\left(u^{(k)}\right)}\right\}$. Otherwise reject $v$.

3. If $v$ is pre-accepted, accept it with probability $a\left(u^{(k)}, v\right) = \min\left\{1, \frac{f_Z\left(y - \widetilde{G}\left(u^{(k)}\right)\right) f_Z(y - G(v))}{f_Z\left(y - \widetilde{G}(v)\right) f_Z\left(y - G\left(u^{(k)}\right)\right)}\right\}$. Otherwise reject $v$.

---

Further improvement can be achieved by increasing the quality of the surrogate model. During the DAMH algorithm, new snapshots $\left(u^{(k)}, G\left(u^{(k)}\right)\right)$ are obtained and it is beneficial to use them for the update of the surrogate model $\widetilde{G}$. The resulting

DAMH algorithm with surrogate model updates (DAMH-SMU) can be written as Alg. 2 with an added step:

4. Optionally, use $\left(u^{(k)}, G\left(u^{(k)}\right)\right)$ to update the surrogate model $\widetilde{G}$.

After $\widetilde{G}$ is modified, it is necessary to recalculate $\widetilde{G}\left(u^{(k)}\right)$; however, the computation cost is assumed to be negligible.

This natural modification of Alg. 2 utilizes all evaluations of $G$ known so far. To implement the surrogate model updates, non-intrusive surrogate models should be used. As shown in numerical experiments in Section 5.1, suitable surrogate models can be constructed for example using a projection to a polynomial basis or using radial basis functions.

MCMC methods are sequential in principle. Therefore, typical utilization of computational resources consists in running several independent sampling processes in parallel. The DAMH-SMU algorithm allows additional acceleration - the parallel processes can share one surrogate model constructed using snapshots obtained by all of the sampling processes. Benefits of this approach are supported by numerical experiments, see Section 5.1.

## 4. Implementation

The sampling methods described in Section 3 are included in the author's Python library for the numerical realization of the Bayesian inversion (available at [5]). The implementation utilizes MPI processes via the mpi4py library, [9]. For the scheme of the parallel processes see Fig. 2. Several sampling processes (SAMPLER 1 to $N$) are running in parallel. These processes share one surrogate model which is refined using data from all of these processes. The COLLECTOR process serves for the construction of the surrogate model and for its distribution to the SAMPLERs. The SAMPLERs also share a pool of SOLVERs, i.e., processes that evaluate the forward model $G$. SOLVERs are typically implemented via a linked numerical library. The SOLVERS POOL assigns the computations required by SAMPLERs to individual SOLVERs. The evaluations of $G$ typically form the majority of the computational time; therefore, for a good utilization of computational resources, the number of SOLVERs is typically lower than the number of SAMPLERs.

The typical sampling process can be divided into several phases:

1. The first phase is the basic MH algorithm, obtained snapshots are used for the construction of a shared surrogate model.

2. The main phase is the DAMH-SMU algorithm during which the surrogate model is updated and used for the acceleration of the sampling process.

3. When the surrogate model is accurate enough, its updates can stop, i.e., the DAMH algorithm is used.

4. Post-processing of obtained samples. This includes the computation of moments, visualization, autocorrelation analysis, etc.
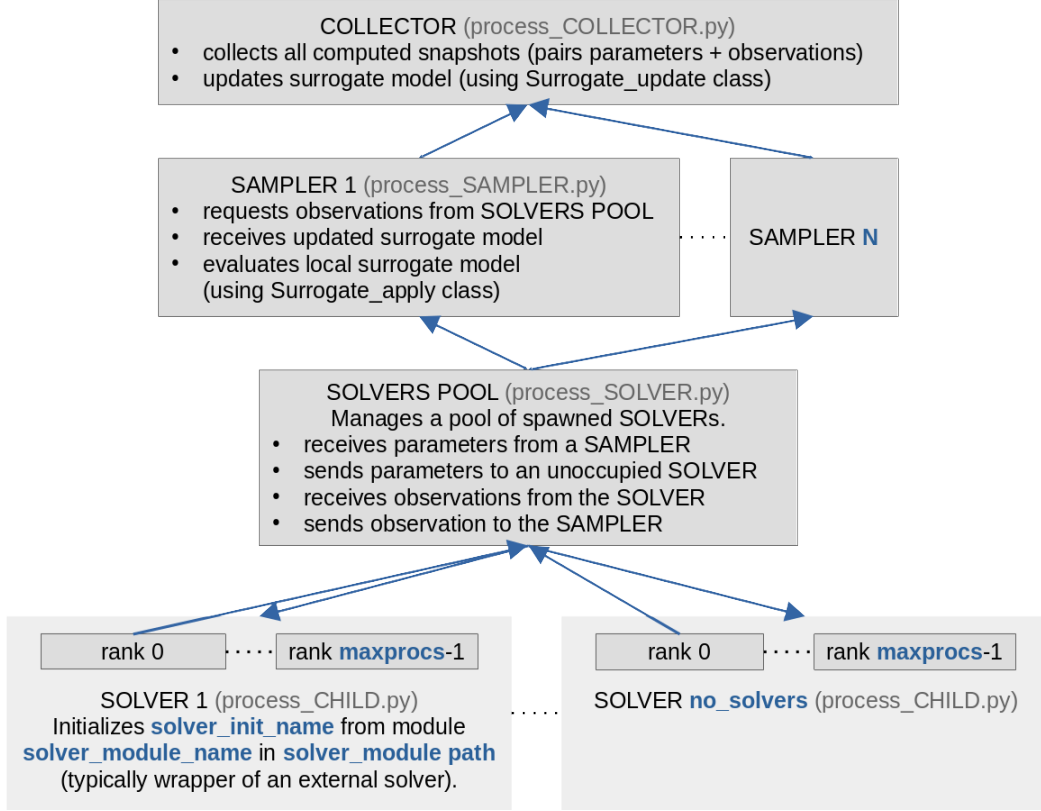
Figure 2: MPI processes

The sampling processes use the Gaussian random walk proposal distribution $\mathcal{N}_n(u, C)$. For each sampling phase, it is possible to specify the covariance matrix $C$, surrogate model type, and a stopping criterion. The stopping criterion can be a pre-specified length of the produced chain, number of $G$ evaluations, or reaching the maximum sampling time. For the surrogate model construction, two methods are implemented: projection to a basis of Hermite polynomials and interpolation using radial basis functions (RBF); for details see [4]. In the case of the polynomial surrogate model, the polynomial degree is chosen based on the number of currently available snapshots up to a pre-specified maximal degree. In the case of the RBF model, it is possible to specify the RBF type (e.g. polyharmonic, Gaussian, etc.) and to set a limit on the number of snapshots used for the construction of the surrogate model.

## 5. Sampling efficiency

MCMC methods are usually understood as methods for the construction of an ergodic Markov chain invariant with respect to (wrt) a target probability distribution. MH and DAMH algorithms are valid (invariant and ergodic) under mild assumptions on supports of $q$, $f$, $\widetilde{f}$.

In the case of the basic MH algorithm, it can be shown that obtained samples form a realization of a Markov chain invariant wrt the posterior distribution. The use of the Gaussian random walk proposal (2) also implies the ergodicity of the Markov chain. For details see e.g. [11].

The DAMH algorithm can be understood as a specific version of the MH algorithm with a modified proposal distribution that includes the pre-acceptance step based on the surrogate model. For the explanation see [6]. Therefore, the resulting Markov chain is also invariant wrt the posterior distribution. Under the additional condition

$$\operatorname{supp} \widetilde{f} \supset \operatorname{supp} f,$$

the Markov chain ergodic, for a detailed explanation see [4].

The DAMH-SMU algorithm was obtained by a small modification of the DAMH algorithm; however, there is a significant difference in ensuring ergodicity. Since $\widetilde{G}$ changes, the proposal distribution also changes and the algorithm becomes an adaptive MCMC method. Several possibilities of ensuring the ergodicity of this type of adaptive MCMC methods are offered by [1, 12]. In situations when it is not possible to prove the ergodicity of the DAMH-SMU algorithm, the validity of the sampling process can be ensured simply by stopping the adaptations at some point. The whole DAMH-SMU algorithm is then understood as a means to find suitable parameters of proposal distribution for the next phase (DAMH algorithm).

Besides the theoretical properties, we should also deal with practical aspects that affect the sampling process efficiency. Specifically, the choice of $q$ has a major impact on sampling efficiency. Too low variance of $q\left(\cdot|u\right)$ causes high autocorrelation of the resulting Markov chain; therefore, too many samples are required to explore the parameter space. Conversely, if $q\left(\cdot|u\right)$ has a high variance, large amount of proposed samples are likely to be rejected, which also results in a high autocorrelation.

There are various approaches attempting to find an "optimal" proposal distribution. Theoretical and experimental results are based on studying the impact of the average acceptance rate

$$\alpha = \mathbb{E}\left[a\left(u,v\right)\right] = \int_{u\in\mathcal{U}} \int_{v\in\mathcal{U}} a\left(u,v\right) \, \mathrm{d}Q\left(u,v\right) \, \mathrm{d}\mu\left(u\right)$$

on the integrated autocorrelation time (IAT) $\tau = 1 + 2\sum_{i=1}^{\infty} \rho_k$ ($\rho_k$ is the autocorrelation at lag $k$). In practice, these values are estimated as a part the the post-processing phase. The average acceptance rate requires monitoring the ratio of accepted/rejected samples. A reliable empirical estimation of IAT is difficult in practice since it requires Markov chains many times longer than the value of IAT.

When an estimation $\widehat{\tau}$ of IAT is available, the sampling efficiency can be assessed using the efficiency criterion proposed in [3] – cost per almost uncorrelated sample (CpUS). In the case of the DAMH algorithm,

$$\mathrm{CpUS} = \left(\frac{N_{\mathrm{acc}} + N_{\mathrm{rej}}}{N} + \mathrm{cost}_{\widetilde{G}}\right)\tau, \tag{4}$$

$N$ is the length of the generated Markov chain, $N_{\text{acc}}$ ($N_{\text{rej}}$) is the number of accepted (rejected) samples, and $\text{cost}_{\widetilde{G}}$ is the ratio of the average cost of $\widetilde{G}$ evaluations to the average cost of $G$ evaluations. In the case of the basic MH algorithm, CpUS is given by IAT (by definition), i.e. $\text{CpUS} = \tau$. This is in accordance with formula (4), since $N = N_{\text{acc}} + N_{\text{rej}}$ and $\text{cost}_{\widetilde{G}} = 0$ ($\widetilde{G}$ is not used).

### 5.1. Numerical experiments

The first set of numerical experiments examines the efficiency of MH and DAMH algorithms depending on the variance of the proposal distribution. These experiments can also be used to compare these algorithms in terms of the optimal CpUS.

The proposal pdf $q\left(\cdot|u\right)$ is the pdf of

$$\mathcal{N}_n\left(u;C\right) = \mathcal{N}_2\left(u;\sigma^2 I\right)$$

where $\sigma \in \{0.4, 0.6, \dots, 5.0\}$. For each $\sigma$, 20 Markov chains of sufficient length (in order to obtain a good estimation of IAT) were generated in parallel using MH and DAMH algorithms. The following methods were considered:

"MH"      Basic MH algorithm, $\text{CpUS} \approx \widehat{\tau}$.

"exact"    DAMH algorithm with a hypothetical ideal surrogate model. This does not require any simulation, data obtained using the MH algorithm are recycled. The surrogate model is assumed to be exact; therefore, there are no rejections and CpUS estimation is obtained as

$$\text{CpUS} \approx \left(\frac{N_{\text{acc}}}{N} + \text{cost}_{\widetilde{G}}\right)\widehat{\tau}.$$

"constant"  DAMH algorithm with a non-informative surrogate model, $\widetilde{G}\left(u\right) = 0$ for each $u$. CpUS is calculated using (4), $\tau$ is replaced by its estimation $\widehat{\tau}$.

Table 1 shows obtained data required for the calculation of CpUS for chosen values of $\sigma$. In this model problem, the evaluations of $G$ are very cheap; therefore, for illustrative purposes, the value of $\text{cost}_{\widetilde{G}}$ is chosen artificially. Figure 3a shows the dependence of CpUS on $\sigma$ for $\text{cost}_{\widetilde{G}} = 0.3$. For each algorithm, the optimal CpUS is marked. However, in typical applications, the value of $\text{cost}_{\widetilde{G}}$ is usually much lower. Figure 3b shows the corresponding results recalculated for $\text{cost}_{\widetilde{G}} = 0.001$. As indicated by these results, the DAMH algorithm requires higher values of $\sigma$ than the MH algorithm for higher efficiency. Furthermore, the optimal value of $\sigma$ increases with decreasing value of $\text{cost}_{\widetilde{G}}$.

In addition, Figure 3b contains results obtained for more realistic surrogate models:

"poly"     DAMH algorithm with a polynomial surrogate model.

"rbf"      DAMH algorithm with a RBF surrogate model.

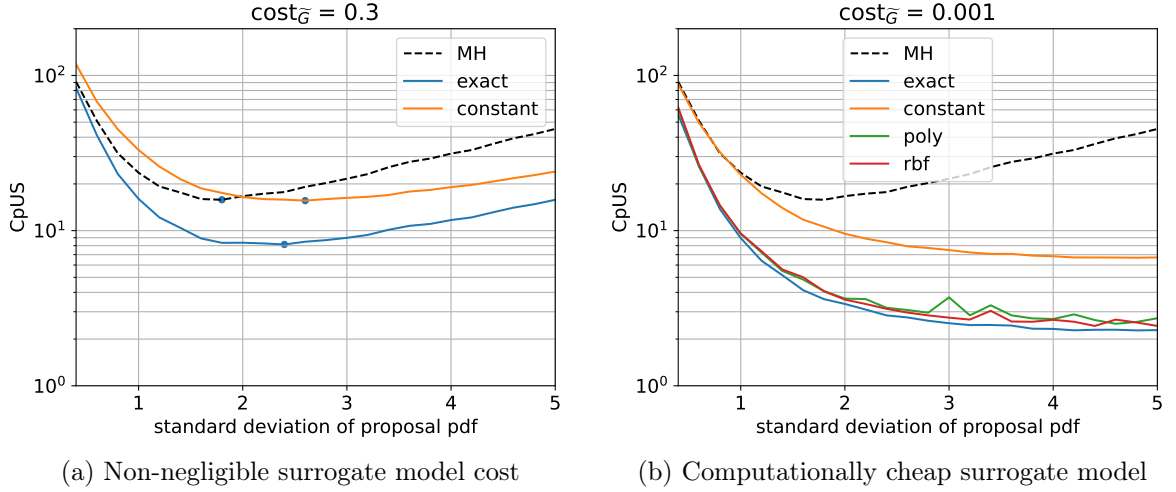| | MH, DAMH "exact" | | | | | | DAMH "constant" | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma$ | 0.4 | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 0.4 | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 |
| $\frac{N_{\text{acc}}}{N}$ | 0.61 | 0.38 | 0.20 | 0.12 | 0.07 | 0.05 | 0.56 | 0.32 | 0.16 | 0.09 | 0.06 | 0.04 |
| $\frac{N_{\text{rej}}}{N}$ | $1 - \frac{N_{\text{acc}}}{N}$ | | | | | | 0.31 | 0.36 | 0.25 | 0.16 | 0.11 | 0.08 |
| $\widehat{\tau}$ | 91.0 | 23.6 | 16.7 | 21.6 | 31.3 | 45.2 | 100.8 | 33.9 | 23.1 | 29.3 | 40.9 | 57.8 |

Table 1: Data required for the calculation of CpUS



(a) Non-negligible surrogate model cost
(b) Computationally cheap surrogate model

Figure 3: Comparison of MH and DAMH algorithms in terms of CpUS

Naturally, the values of CpUS cannot be lower than with the "exact" surrogate model. The lowest achieved CpUS is approximately 16 for MH, 2.5 for "poly" and "rbf", and 2.3 for "exact".

In the previous experiment, standard DAMH algorithm without surrogate model updates was considered. The second numerical experiment is designed to show the benefits of surrogate model updates during the sampling process. This will be shown through monitoring the amount of rejected samples in the DAMH algorithm. These samples require the evaluation of the (usually computationally expensive) observational operator $G$ but they are rejected afterwards; therefore, the number of rejected samples should be as low as possible. In the hypothetical ideal case of the exact surrogate model, there would be no rejected samples.

For this purpose, the RFB surrogate model is used and the sampling process is divided into several phases:

$$\text{MH (100)} \rightarrow \text{DAMH (long)} \overbrace{\rightarrow \text{DAMH-SMU (100)} \rightarrow \text{DAMH (long)}}^{\text{5 times}}$$

The initial phase is the basic MH algorithm with stopping criterion set to 100 evaluations of $G$. Then, there are five DAMH-SMU phases, all of them terminated after

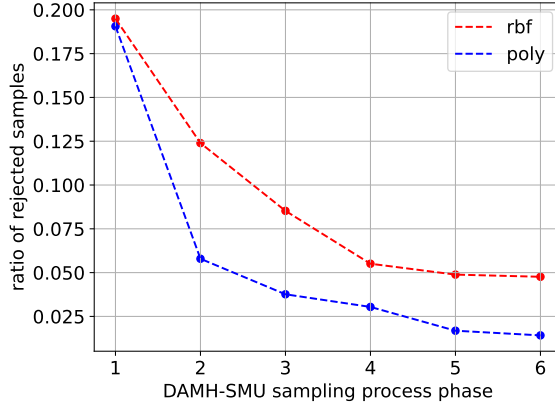Figure 4: Benefits of surrogate model updates

100 evaluations of $G$. After each of the phases, there is a long DAMH phase. These additional DAMH phases serve only for monitoring the surrogate model quality, they do not affect the DAMH-SMU phases in any way. Figure 4 shows the ratio of rejected samples (to the number of all samples) for each DAMH phase corresponding to surrogate models constructed from increasing numbers of snapshots (from 100 to 600). The figure shows the intended behavior – with increasing quality of the surrogate model, the ratio of rejected samples decreases.

## 6. Conclusions

The contribution focused on MCMC methods providing samples from the exact posterior distribution. Such methods require many evaluations of the observational operator. It was shown that the use of surrogate models and the DAMH algorithm can spare a significant number of $G$ evaluations compared to the basic MH algorithm. Also, it was shown that the surrogate model can be updated during the sampling process, leading to a further increase in the efficiency of the sampling process.

An advantage of the presented Python framework is that the implemented methods have general use, the forward model $G$ can be a "black box". The only requirements for the use of this Python package are the specification of the prior distribution and the observational noise, and the availability of a solver that evaluates the observational operator $G$.

## Acknowledgement

# References

[1] Bai, Y., Roberts, G.O., and Rosenthal, J.S.: On the containment condition for adaptive Markov chain Monte Carlo algorithms. Advances and Applications in Statistics **21** (2011), 1–54.

[2] Blaheta, R., Béreš, M., Domesová, S., and Horák, D.: Bayesian inversion for steady flow in fractured porous media with contact on fractures and hydro-mechanical coupling. Computational Geosciences (2020).

[3] Blaheta, R., Béreš, M., Domesová, S., and Pan, P.: A comparison of deterministic and Bayesian inverse with application in micromechanics. Applications of Mathematics **63** (2018), 665–686.

[4] Bérešová, S.: *Bayesian approach to the identification of parameters of differential equations.* PhD thesis, VSB - Technical University of Ostrava, Ostrava, 2022. URL http://hdl.handle.net/10084/148521.

[5] Bérešová, S.: surrDAMH, 2022. URL https://github.com/dom0015/surrDAMH/tree/Version1.

[6] Christen, J.A. and Fox, C.: Markov chain Monte Carlo Using an Approximation. Journal of Computational and Graphical Statistics **14** (2005), 795–810.

[7] Cui, T., Fox, C., and O'Sullivan, M.J.: Bayesian calibration of a large-scale geothermal reservoir model by a new adaptive delayed acceptance Metropolis Hastings algorithm. Water Resources Research **47** (2011).

[8] Cui, T., Fox, C., and O'Sullivan, M.J.: A posteriori stochastic correction of reduced models in delayed-acceptance MCMC, with application to multiphase subsurface inverse problems. International Journal for Numerical Methods in Engineering **118** (2019), 578–605.

[9] Dalcin, L.: mpi4py, 2022. URL https://mpi4py.readthedocs.io.

[10] Lykkegaard, M.B., Mingas, G., Scheichl, R., Fox, C., and Dodwell, T.J.: Multilevel Delayed Acceptance MCMC with an Adaptive Error Model in PyMC3. Tech. Rep. arXiv:2012.05668, 2020.

[11] Robert, C.P. and Casella, G.: *Monte Carlo statistical methods.* Springer texts in statistics, Springer, New York, NY, 2004. OCLC: 837651914.

[12] Roberts, G.O. and Rosenthal, J.S.: Coupling and Ergodicity of Adaptive Markov Chain Monte Carlo Algorithms. Journal of Applied Probability **44** (2007), 458–475.