Surrogate-Accelerated Markov Chain Monte Carlo Methods for Bayesian Inverse Problems

Simona Bérešová[1,2], Michal Béreš[1,2], Tomáš Luber[1]

November 6, 2024

[1]Institute of Geonics, Czech Academy of Sciences
[2]Department of Applied Mathematics, FEECS, VSB - Technical University of Ostrava

1. **Bayesian inverse problem**
2. **MCMC methods** for posterior sampling
3. Sampling process acceleration using **surrogate models** (polynomials, radial basis functions, neural networks)
4. **Geotechnical appllications**
5. Implementation (Python library)

Inverse problem

What is given: forward model $G : \mathbb{R}^n \to \mathbb{R}^m$, its noisy outputs $y \in \mathbb{R}^m$

What is unknown: input parameters $u \in \mathbb{R}^n$

In the case of the Bayesian inversion, we work with random variables:

- unknown parameters $U : \Omega \to \mathbb{R}^n$
- observed data $Y : \Omega \to \mathbb{R}^m$
- observational noise $Z : \Omega \to \mathbb{R}^m$

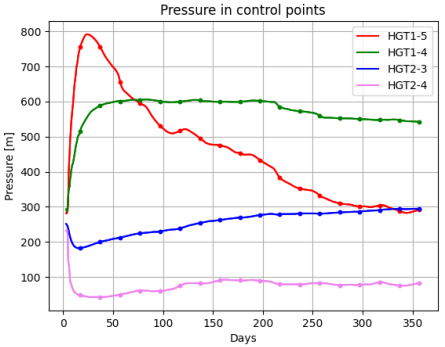
and some noise model, e.g. additive: $Y = G(U) + Z$

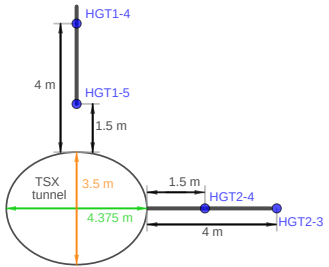
Posterior distribution (i.e. conditional distribution of U given $Y = y$) is given by its probability density function (pdf) as

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

(f_Z ... noise pdf, f_U ... prior pdf)

- **available measurements:** 4 time series (pore pressure in 4 control points during 1 year), coming from a tunnel sealing experiment (TSX) in Canada
- **observed data:** pore pressure in these 4 control points and 18 time points





- **forward model:** linear poroelasticity
- **unknown parameters:** permeability, storativity, Young's modulus, Poisson's ratio in each subdomain; initial stresses in x- and y- direction

9 subdomains:

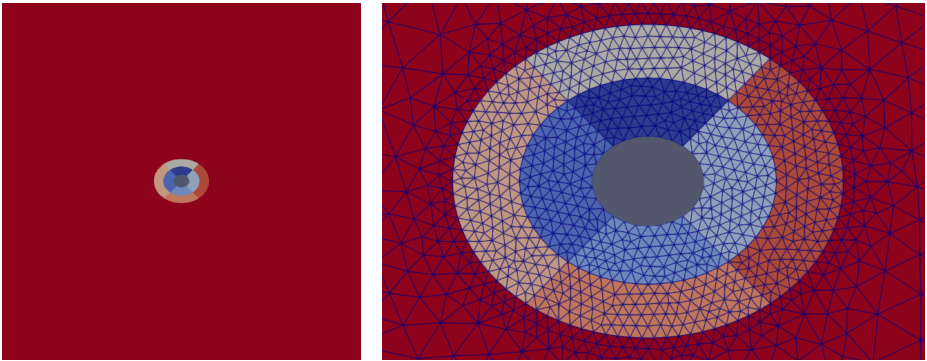


Figure 1: Whole domain (left), cutout (right)

- result of Bayesian inversion = posterior distribution

$$f_{U|Y}(u|y) \propto \underbrace{f_Z(y - G(u))}_{\text{data likelihood}} \underbrace{f_U(u)}_{\text{prior}}$$

  - given by the Bayes' theorem, known up to a multiplicative constant
  - information about this distribution can be obtained by sampling
  - MCMC methods based on the Metropolis-Hastings algorithm are designed for these situations

Notation

Since y is fixed, target distribution can be represented (up to a normalizing constant) by a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$f(u) = f_Z(y - G(u)) f_U(u).$$

Given:

- conditional density $q : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ (proposal distribution)
- initial sample $u^{(0)} \in \mathbb{R}^n$ such that $f\left(u^{(0)}\right) > 0$

Metropolis-Hastings (MH)

For $k = 0, 1, \ldots$

1. **Propose a sample** v from $q\left(\cdot|u^{(k)}\right)$.
2. With probability $a(u, v) = \min\left\{1, \frac{q\left(u^{(k)}|v\right)f(v)}{q\left(v|u^{(k)}\right)f\left(u^{(k)}\right)}\right\}$, **accept** v, i.e., set $u^{(k+1)} = v$. Otherwise, **reject** v, i.e., set $u^{(k+1)} = u^{(k)}$.

Metropolis-Hastings (MH)

For $k = 0, 1, \ldots$:

- **Propose a sample** v from $q\left(\cdot|u^{(k)}\right)$.
- **Accept** v with probability
$$\min\left\{1, \frac{q\left(u^{(k)}|v\right) f_Z\left(y - G(v)\right) f_U(v)}{q\left(v|u^{(k)}\right) f_Z\left(y - G\left(u^{(k)}\right)\right) f_U\left(u^{(k)}\right)}\right\}.$$

Delayed aceptance MH

For $k = 0, 1, \ldots$:

- **Propose a sample** v from $q\left(\cdot, u^{(k)}\right)$.
- **Pre-accept** v with probability
$$\min\left\{1, \frac{q\left(u^{(k)}|v\right) f_Z\left(y - \widetilde{G}(v)\right) f_U(v)}{q\left(v|u^{(k)}\right) f_Z\left(y - \widetilde{G}\left(u^{(k)}\right)\right) f_U\left(u^{(k)}\right)}\right\};$$
- if v is pre-accepted, **accept** it with probability
$$\min\left\{1, \frac{f_Z\left(y - \widetilde{G}\left(u^{(k)}\right)\right) f_Z\left(y - G(v)\right)}{f_Z\left(y - \widetilde{G}(v)\right) f_Z\left(y - G\left(u^{(k)}\right)\right)}\right\}.$$

- better surrogate model $\longrightarrow$ lower number of unnecessary G evaluations
- during the DAMH algorithm, new snapshots $\left(u^{(k)}, G\left(u^{(k)}\right)\right)$ are obtained
- they can be used to update the surrogate model $\widetilde{G}$
$\longrightarrow$ **DAMH algorithm with surrogate model updates (DAMH-SMU)**

Endless possibilities, for example:

- Adaptive Metropolis (Haario 2001) - adaptation of the proposal distribution $\rightarrow$ converges to Gaussian approximation of the target distribution
- Delayed rejection MH (Peskun 1973), DRAM (Haario 2006) - when a proposal is rejected, instead of retaining the same position, propose a new candidate
- Hamiltonian MC (1987), NUTS - proposal based on a Hamiltonian dynamics evolution, requires differentiation
- Delayed acceptance MH (Christen, Fox 2005)
  - adaptive error model construction (Cui et al 2011) - error between fine and coarse model approximated by Gaussian distribution
  - multilevel variants (e.g. Dodwell et al 2015, Lykkegaard et al 2020) - require a hierarchy of approximations of the target distribution
  - subchain on approximated distribution (instead of single proposal)
- DREAM (Vrugt et al 2008) - proposal based on differential evolution learning strategy, especially for multimodal target distributions
- many ways of using surrogate models of G, resulting in exact/approximate sampling

and mix of all above.

Endless possibilities, for example:

- Adaptive Metropolis (Haario 2001) - adaptation of the proposal distribution → converges to Gaussian approximation of the target distribution
- Delayed rejection MH (Peskun 1973), DRAM (Haario 2006) - when a proposal is rejected, instead of retaining the same position, propose a new candidate
- Hamiltonian MC (1987), NUTS - proposal based on a Hamiltonian dynamics evolution, requires differentiation
- Delayed acceptance MH (Christen, Fox 2005)
  - adaptive error model construction (Cui et al 2011) - error between fine and coarse model approximated by Gaussian distribution
  - multilevel variants (e.g. Dodwell et al 2015, Lykkegaard et al 2020) - require a hierarchy of approximations of the target distribution
  - subchain on approximated distribution (instead of single proposal)
- DREAM (Vrugt et al 2008) - proposal based on differential evolution learning strategy, especially for multimodal target distributions
- many ways of using surrogate models of G, resulting in exact/approximate sampling

and mix of all above.

Collect an initial set S of snapshots of G, i.e., pairs $\left(u^{(i)}, G\left(u^{(i)}\right)\right)$.

For $k = 0, 1, \ldots$

1. Construct surrogate model $\widetilde{G}^{(k)} : \mathbb{R}^n \to \mathbb{R}^m$ based on snapshots in S.
2. **Generate proposal v using a subchain**: Set $w^{(0)} = u^{(k)}$. For $m = 0, 1, \ldots, m_{\max} - 1$
   2.1 Propose a sample z from $q^{(k)}\left(\cdot|w^{(m)}\right)$.
   2.2 With probability $\widetilde{a}\left(w^{(m)}, z\right) = \min\left\{1, \frac{q^{(k)}\left(w^{(m)}|z\right) f_Z\left(y - \widetilde{G}^{(k)}(z)\right) f_U(z)}{q^{(k)}\left(z|w^{(m)}\right) f_Z\left(y - \widetilde{G}^{(k)}\left(w^{(m)}\right)\right) f_U\left(w^{(m)}\right)}\right\}$, accept z, i.e., set $w^{(m+1)} = z$. Otherwise, set $w^{(m+1)} = w^{(m)}$.
3. Set $v = w^{(m_{\max})}$.
4. With probability $a_{\widetilde{Q},\mu}\left(u^{(k)}, v\right) = \min\left\{1, \frac{f_Z\left(y - \widetilde{G}^{(k)}\left(u^{(k)}\right)\right) f_Z(y - G(v))}{f_Z\left(y - \widetilde{G}^{(k)}(v)\right) f_Z\left(y - G\left(u^{(k)}\right)\right)}\right\}$, **accept** v, i.e., set $u^{(k+1)} = v$. Otherwise, **reject** v, i.e., set $u^{(k+1)} = u^{(k)}$.
5. Add $(v, G(v))$ to the set of snapshots.
6. Construct updated proposal pdf $q^{(k+1)}$.

MCMC method = method for the construction of an ergodic Markov chain invariant with respect to (wrt) a target probability measure μ

Metropolis-Hastings algorithm

produces a Markov chain with transition kernel

$$K(u, dv) = \underbrace{a(u, v)\, Q(u, dv)}_{\text{accepting proposed sample}} + \underbrace{\left(\int_{\mathcal{U}} (1 - a(u, w))\, Q(u, dw) \right) \delta_u(dv)}_{\text{rejecting proposed sample}},$$

$Q : \mathcal{U} \times \mathcal{B}(\mathcal{U}) \to \mathbb{R}$ ... proposal kernel (e.g. Gaussian random walk),
$a : \mathcal{U} \times \mathcal{U} \to [0, 1]$ ... acceptance probability, depends on μ and Q

- invariant wrt μ
- ergodic under mild conditions (aperiodicity and μ-irreducibility)

In computational practice $\mathcal{U} = \mathbb{R}^n$ ($n \in \mathbb{N}$), Q, μ have Lebesgue densities (i.e. pdf), e.g., $Q(u, dv) = q(v|u)\, \lambda(dv) : \mathbb{R}^n \times \mathcal{B}(\mathbb{R}^n) \to \mathbb{R}$

DAMH is in fact the standard MH algorithm with proposal kernel chosen as another MH kernel

$$\widetilde{Q}(u, dv) = \widetilde{a}(u, v)\, Q(u, dv) + \left(\int_{\mathcal{U}} (1 - \widetilde{a}(u, w))\, Q(u, dw) \right) \delta_u(dv),$$

invariant wrt an auxiliary measure $\widetilde{\mu} \approx \mu$ represented by a function

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

Consider a MH algorithm producing an ergodic Markov chain invariant wrt f. Assume that $\operatorname{supp} \widetilde{f} \supset \operatorname{supp} f$. Then the DA version of the algorithm also produces an ergodic Markov chain invariant wrt f.

DAMH with surrogate model updates (DAMH-SMU)

- $\widetilde{G}$ changes during the sampling process
- with changes of $\widetilde{G}$, the proposal distribution also changes $\rightarrow$ adaptive method
- ergodicity can be established e.g. by satisfying diminishing adaptation property and containment property

$\{K_\gamma : \gamma \in \mathcal{W}\}$ ... all possible transition kernels, all of them invariant wrt μ

$U^{(k)}$ ... random variable representing the k-th sample

$\Gamma^{(k)}$ ... random variable representing the index of the k-th transiton kernel

$P^{(k)}_{u,\gamma}$ ... probability measure such that $P^{(k)}_{u,\gamma}(A) = \Pr\left(U^{(k)} \in A | U^{(0)} = u \wedge \Gamma^{(0)} = \gamma\right)$ for all $A \in \mathcal{B}(\mathcal{U})$

$A^{(k)} = \sup_{u\in\mathcal{U}}\left(d_{TV}\left(K_{\Gamma(k+1)}(u,\cdot), K_{\Gamma(k)}(u,\cdot)\right)\right)$ ... amount of adaptation done between iterations k, $k+1$

An adaptive MCMC algorithm satisfies the **diminishing adaptation property** if for all $\varepsilon > 0$, $u \in \mathcal{U}$, and $\gamma \in \mathcal{W}$,

$$\lim_{k\to\infty} \Pr\left(A^{(k)} > \varepsilon | U^{(0)} = u \wedge \Gamma^{(0)} = \gamma\right) = 0$$

Can be usually ensured by the design of the rules for the performed adaptations. With increasing k, the adaptively chosen parameters can be modified by decreasing amounts, or the adaptation step itself can be applied with decreasing probability.

An adaptive MCMC algorithm satisfies the **containment condition** if for all $u \in \mathcal{U}$, $\gamma \in \mathcal{W}$, $\varepsilon > 0$, and $\delta > 0$, there is $N \in \mathbb{N}$ such that

$$\Pr\left(\inf\left\{k \geq 1 : d_{TV}\left(K^k_{\Gamma(k)}\left(u^{(k)},\cdot\right),\mu\right) \leq \varepsilon\right\} \leq N | U^{(0)} = u \wedge \Gamma^{(0)} = \gamma\right) \geq 1 - \delta$$

for all $k \in \mathbb{N}$.

Standard Metropolis-Hastings:

$$\text{CpUS} = \tau$$

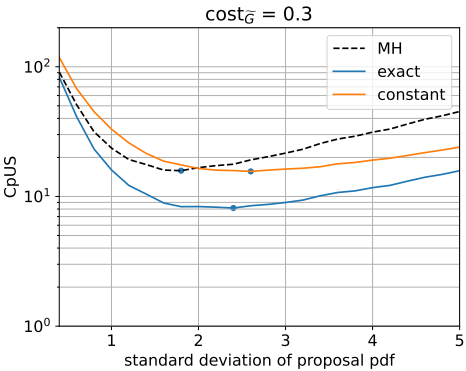
DAMH-based methods:

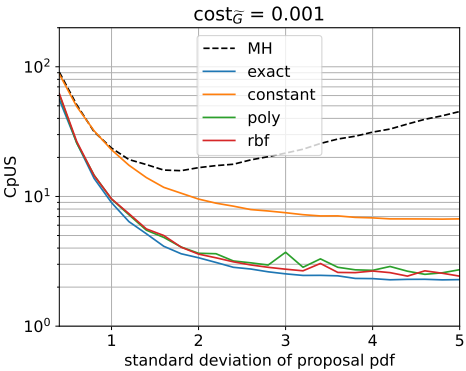
$$\text{CpUS} = \tau \cdot \text{"cost per one sample"} = \tau \frac{N_{\text{full}} + c_{\text{surr}} N_{\text{surr}}}{\text{chain length}}$$

- CpUS ... cost per one almost uncorrelated sample
- τ ... autocorrelation time estimation (!)
- c_{surr} ... cost of surrogate model evaluation (relative to the cost of the full model)
- N_{full} ... number of full model evaluations
- N_{surr} ... number of surrogate model evaluations

Effective sample size:

$$\text{ESS} = \frac{\text{chain length}}{\text{CpUS}}$$





- further improvements:
  - surrogate model updates during the sampling process - less rejections
  - subchains - lower autocorrelation

= non-intrusive surrogate models that can be constructed from collected snapshots $\left(u^{(k)}, G\left(u^{(k)}\right)\right)$ and adaptively refined

Implemented within the inhouse Python library SurrDAMH (github.com/dom0015/surrDAMH):

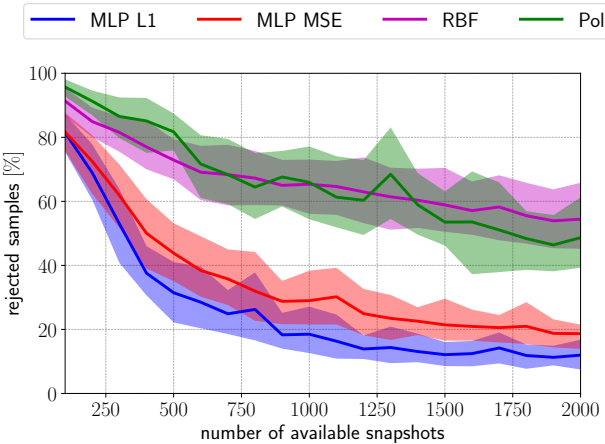
- polynomial chaos approximation - complete polynomials, adaptive increase of maximum degree based on available data (using scikit-learn)
- radial basis functions interpolation (RBF) - combined with polynomials up to chosen degree (using SciPy)
- approximation using nearest points identified using kd-tree (using SciPy)
- neural networks - multilayer perceptron regressor (using PyTorch)

- multilayer perceptron regressor (MLP)
- tanh activation
- ADAM (Adaptive Moment Estimation) learning algorithm, combination with L-BFGS also tested
- MSE (mean square error) loss function

Use of neural networks during the sampling process

- data for the surrogate (MLP) generated incrementally during the sampling process
- MLP is trained continuously, training data are added when available
- whenever new data arrives, MLP is trained on them (using ADAM) until some threshold loss is achieved before appending them to the whole training set
- training can be skipped when target loss is achieved and no new data is added (until some new data arrives)

Surrogate model updates during the sampling process lead to reducing the number of rejected samples:



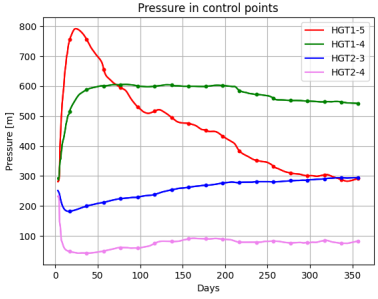
MPL: multilayer perceptron

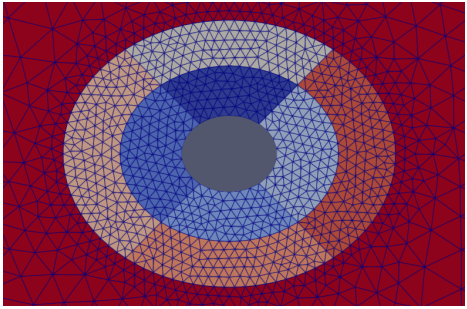
RBF: thin plate spline

$$\phi(r) = r^2 \log(r)$$

Poly: complete polynomials

- **observed data:** pore pressure in 4 control points and 18 time points
  $\rightarrow Y : \Omega \rightarrow \mathbb{R}^{72}$, $y \in \mathbb{R}^{72}$
- **unknown parameters:**
  - permeability, storativity, Young's modulus, Poisson's ratio in each of 9 subdomains ($4 \cdot 9$ parameters)
  - initial stresses σ_x, σ_y ($+2$ parameters)
    $\rightarrow U : \Omega \rightarrow \mathbb{R}^{38}$
- **forward model:** linear poroelasticity, $G : \mathbb{R}^{38} \rightarrow \mathbb{R}^{72}$ (using FEniCSx - DOLFINx)





- f_U ... prior pdf: independent components

		$\log_{10}$(mean)	$\log_{10}$(mode)
permeability	$\mathrm{Log}\mathcal{N}(-40, 3)$	-15.42	-21.28
storativity	$\mathrm{Log}\mathcal{N}(-25, 3)$	-8.90	-14.77
Young's modulus	$\mathrm{Log}\mathcal{N}(26, 2)$	12.16	9.55
Poisson's ratio	$\mathcal{U}(0, 0.5)$		
initial stresses	$\mathrm{Log}\mathcal{N}(16, 2)$	7.82	5.21

(internally, $\mathcal{N}(0, 1)$ is used, transformation is done before sending to the solver G)

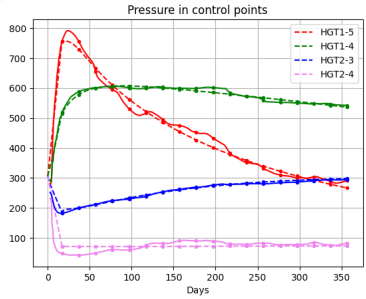
- f_Z ... noise pdf: additive Gaussian noise with covariance

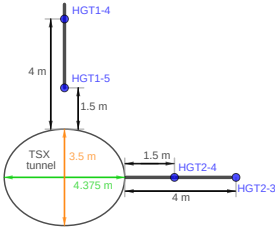
$$\mathrm{cov}(x_1, x_2) = \sigma^2 \exp\left(\frac{\|x_1 - x_2\|}{\lambda}\right), \ \lambda = 50, \ \sigma = 20$$

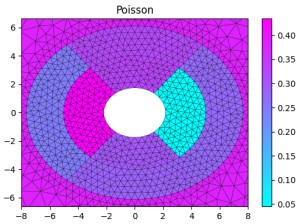
(for each time series)

$\longrightarrow$ Using MCMC methods, we obtain samples in 38-dimensional state space. How to analyse them and visualize?

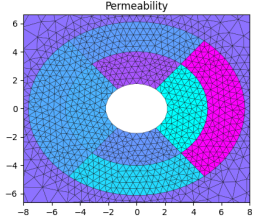
Best fit

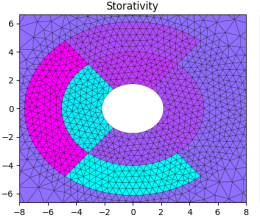


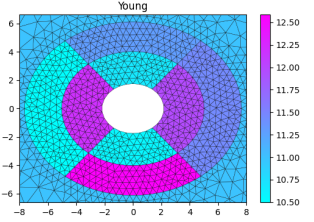


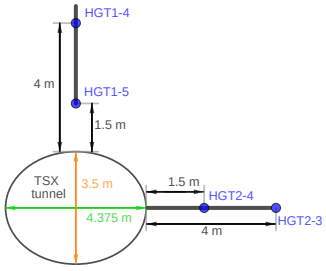


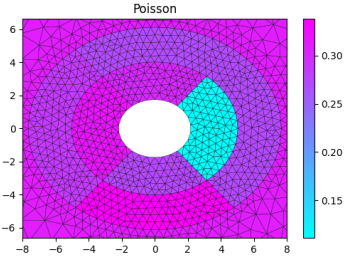
$\log_{10}$:



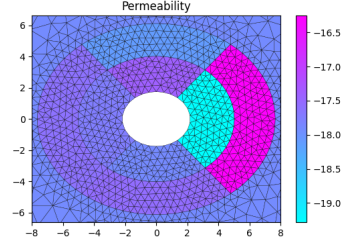


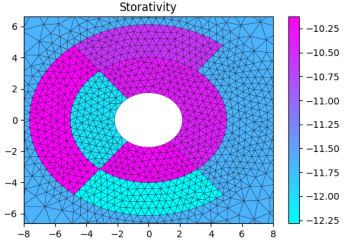


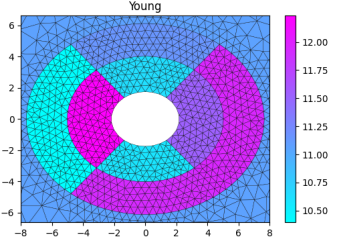

HGT1-4
4 m
HGT1-5
1.5 m
TSX
tunnel
3.5 m
4.375 m
1.5 m
HGT2-4
HGT2-3
4 m


Poisson

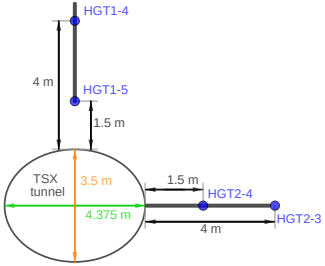
$\log_{10}$(mean):

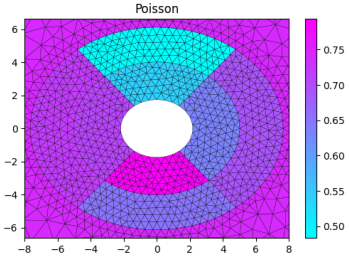

Permeability

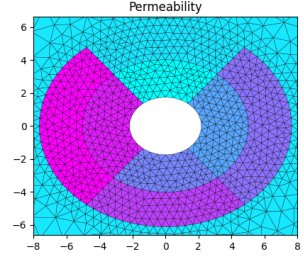

Storativity

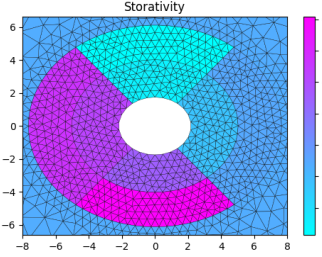

Young

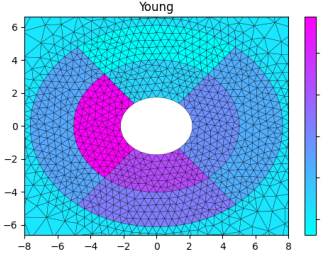
Posterior standard deviation (compared to internal prior $\mathcal{N}(0,1)$)



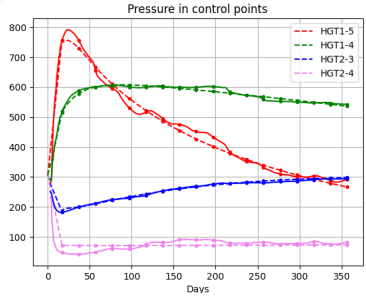


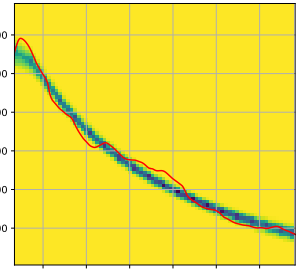


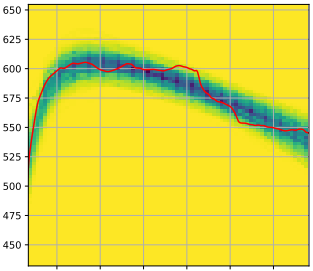


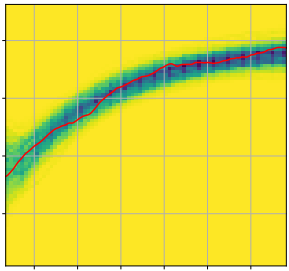


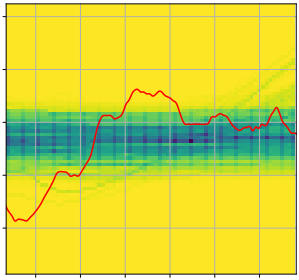
Bayesian inversion results - posterior reflected into measurements



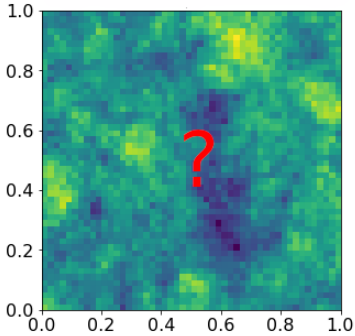


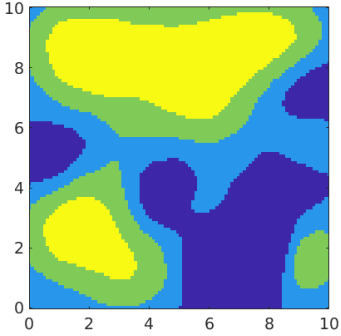






- unknown parameters: material parameters in the domain of interest (constant subdomains, Gaussian random field, etc.)



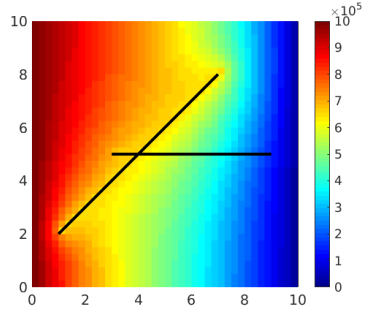


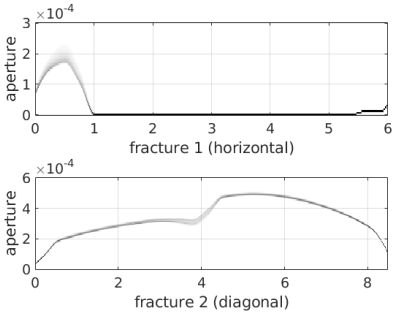
- observed data: e.g. measurements of pore pressure, boundary flow, ...

Aim: probabilistic description of unknown parameters

- unknown parameters: aperture of two fractures
- forward model: Darcy flow in a square domain $(0, 10) \times (0, 10)$
- observed data: flow through chosen boundary parts

Result: posterior distribution of fracture apertures (each vertical slice represents the histogram of the aperture in the corresponding fracture point)



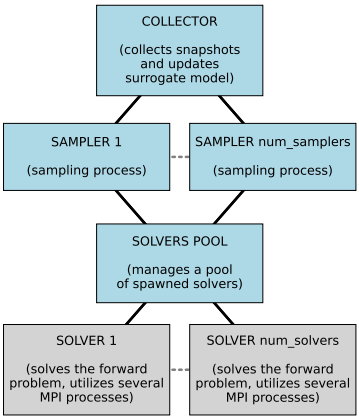


- for posterior sampling using MH, DAMH, DAMH-SMU algorithms
- available at github.com/dom0015/surrDAMH

Parallelization using MPI:

- several Markov chains generated in parallel
- the chains share one surrogate model, refined using data from all chains
- they also share a pool of solvers (processes that evaluate G)
  - computationally most demanding part
  - to keep the solvers busy, number of solvers should be lower than number of chains

(if the solver is not a parallel application, it can run locally on samplers without the use of solvers pool)



Inputs to the sampling framework:

- configuration (basic settings, e.g. number of solvers, initial samples, ...)
- prior
- likelihood
- solver specification
- surrogate model updater
- list of stages

Example:

- **stage 1:** MH (short run, construction of an initial surrogate model)
- **stage 2:** DAMH-SMU, higher proposal SD
- **stage 3:** DAMH (surrogate model is used but no longer updated)
- post-processing of obtained samples, visualization, autocorrelation analysis

SurrDAMH usage - an example

```
""" mpiexec -n 6 python3 -m mpi4py typical_example.py """

conf = surrDAMH.Configuration(no_parameters=2, no_observations=1, output_dir="out_dir",
                              use_solvers_pool=True, no_solvers=2)
solver = solver_examples.solver_spec_examples.SolverSpecExample1()  # solver example
prior = surrDAMH.distributions.Normal(mean=[0.0, 0.0], sd=1.0)  # Gaussian prior
likelihood = surrDAMH.distributions.Normal(mean=[2.3], sd=1.0)  # additive Gaussian noise
updater = surrDAMH.surrogates.RBFInterpolationUpdater(no_parameters=2, no_observations=1)

list_of_stages = [Stage(algorithm_type="MH", sd=0.5, max_evaluations=500)]  # MH
list_of_stages.append(Stage(algorithm_type="DAMH", sd=1.0, max_evaluations=500,
                            surrogate_model_updates=True))  # DAMH-SMU
list_of_stages.append(Stage(algorithm_type="DAMH", sd=1.0, max_evaluations=500,
                            surrogate_model_updates=False))  # DAMH

sam = surrDAMH.SamplingFramework(conf, prior, likelihood, list_of_stages,
                                 solver, surrogate_updater=updater)
sam.run()

# post processing, autocorrelation analysis
```