UQLab user manual
Bayesian inference for model calibration and inverse problems

P.-R. Wagner, J. Nagel, S. Marelli, B. Sudret

Chair of Risk, Safety and Uncertainty Quantification
Stefano-Franscini-Platz 5
CH-8093 Zürich

How to cite UQLAB

S. Marelli, and B. Sudret, UQLab: A framework for uncertainty quantification in Matlab, Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom, 2014, 2554-2563.

How to cite this manual

P.-R. Wagner, J. Nagel, S. Marelli, B. Sudret, UQLAB user manual – Bayesian inference for model calibration and inverse problems, Report UQLab-V1.2-113, Chair of Risk, Safety & Uncertainty Quantification, ETH Zurich, 2019.

BIBTEX entry

```
@TECHREPORT{UQdoc_12_113,
author = {Wagner, P.-R. and Nagel, J. and Marelli, S. and Sudret, B.},
title = {{UQLab user manual -- Bayesian inversion for model calibration and
validation}},
institution = {Chair of Risk, Safety \& Uncertainty Quantification, ETH Zurich},
year = {2019},
note = {Report \# UQLab-V1.2-113}
}
```

Document Data Sheet

Document Ref.	UQLab-V1.2-113
Title:	UQLab user manual – Bayesian inference for model calibration and inverse problems
Authors:	P.-R. Wagner, J. Nagel, S. Marelli, B. Sudret Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland
Date:	22/02/2019

Doc. Version	Date	Comments
V1.2	22/02/2019	First release

Abstract

Bayesian inference is a powerful tool for probabilistic model calibration and inversion. It provides a comprehensive framework for combining information about parameters *prior* to observations with information obtained from *experiments*. In Bayesian inference, this combined information is expressed in a so-called *posterior* distribution of the parameters.

The UQLAB Bayesian inference module offers an easy way to setup a Bayesian inverse problem and to compute its posterior distribution. It makes use of other available UQLAB modules (UQLAB User Manual – the INPUT module , UQLAB User Manual – the MODEL module) to define the forward model and the prior distribution. For the computation of the posterior distribution, state-of-the-art Markov chain Monte Carlo sampling algorithms are supplied. The manual for the Bayesian inversion module is divided into three parts:

- A brief introduction to the main ideas and theoretical foundations of Bayesian inversion and Markov chain Monte Carlo algorithms;
- An example-based guide with an explanation of the available options and methods;
- A comprehensive reference list detailing all available functionalities of the Bayesian inversion module.

Keywords: UQLAB, Bayesian inversion, model calibration, inverse problems, Markov chain Monte Carlo, VVUQ (verification, validation and uncertainty quantification)

Contents

Chapter 1

Theory

This section contains a short introduction to Bayesian methods (Gelman et al., 2014) with a focus on inverse problems (Tarantola, 2005; Kaipio and Somersalo, 2005). An inverse problem arises when unknown parameters that cannot be directly measured are estimated based on experimental data that is only indirectly related to the parameters through a computational model. The problem is called *inverse*, because instead of propagating information about input parameters through a computational model (so-called *forward* approach), the goal is to propagate information about the observations *backwards* to obtain insight on the model inputs. This formulation encompasses a range of problems in the engineering and natural sciences (Hadidi and Gucunski, 2008; Beck, 2010; Yuen and Kuok, 2011).

1.1 Bayesian inference

Statistics is generally described as the science that allows one to build models of complex phenomena based on data. *Statistical inference* usually considers that the data set at hand is made of independent realizations of an underlying random vector $\boldsymbol{X}$ with a probability density function (PDF) $\pi(\boldsymbol{x})$, whose properties have to be established from that data. Parametric statistical models make an assumption on the shape of this PDF (*e.g.* Gaussian, Weibull, lognormal, etc. in one-dimensional case), and the goal of inference is to estimate the *hyperparameters* $\boldsymbol{\theta}$ of the PDF from the data. The latter is further denoted by:

$$\boldsymbol{X} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta}) \tag{1.1}$$

for the sake of clarity.

When a sufficient amount of data exists, classical estimators can be used: for instance, if a Gaussian distribution $X \sim \mathcal{N}(\boldsymbol{x}|\mu, \sigma^2)$ is to be fitted to a data set $\{x_1, \ldots, x_N\}$ (with N sufficiently large), the empirical mean and standard deviation of the sample may be used as estimators of the hyperparameters $\boldsymbol{\theta} \stackrel{\text{def}}{=} (\mu, \sigma^2)$. Such a direct estimation is however not reliable when there is only a handful of data points. In technical terms, the statistical uncertainty of the estimator of $\boldsymbol{\theta}$, denoted by $\hat{\boldsymbol{\theta}}$, becomes too large in this case.

In this context, *Bayesian statistics* allows one to fit a statistical model by combining some prior knowledge on the hyperparameters with the (possibly few) observed data points, using the Bayes' theorem[1]. In the Bayesian paradigm the hyperparameters are considered as a random vector denoted by $\boldsymbol{\Theta}$ which follow the so-called *prior distribution* (with support $\mathcal{D}_{\boldsymbol{\Theta}}$):

$$\boldsymbol{\Theta} \sim \pi(\boldsymbol{\theta}). \tag{1.2}$$

This subjective choice should reflect the level of information existing on the hyperparameters $\boldsymbol{\theta}$ *before* any measurement of the parameters $\boldsymbol{X}$ is carried out. From Bayes' theorem the *posterior distribution* of the hyperparameters, denoted as $\pi(\boldsymbol{\theta}|\boldsymbol{x})$, is obtained by:

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\pi(\boldsymbol{x}|\boldsymbol{\theta})\;\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{x})}. \tag{1.3}$$

Consider now a data set of measured values of $\boldsymbol{X}$, say $\mathcal{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^\top$, whose points are viewed as independent realizations of $\boldsymbol{X} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta})$. The *likelihood function* $\mathcal{L}(\boldsymbol{\theta};\, \mathcal{X})$, seen as a function of the hyperparameters $\boldsymbol{\theta}$, is defined by:

$$\mathcal{L} :\; \boldsymbol{\theta} \mapsto \mathcal{L}(\boldsymbol{\theta};\, \mathcal{X}) \stackrel{\text{def}}{=} \prod_{k=1}^{n} \pi(\boldsymbol{x}_k|\boldsymbol{\theta}). \tag{1.4}$$

The intuitive interpretation of the above equation is that the likelihood function is the probability of observing the data at hand under the assumption that $\boldsymbol{X} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta})$.

Following Bayes' theorem, the *posterior distribution* $\pi(\boldsymbol{\theta}|\mathcal{X})$ of the hyperparameters $\boldsymbol{\theta}$ given the observations in $\mathcal{X}$ can now be computed as:

$$\pi(\boldsymbol{\theta}|\mathcal{X}) = \frac{\mathcal{L}(\boldsymbol{\theta};\, \mathcal{X})\, \pi(\boldsymbol{\theta})}{Z}, \tag{1.5}$$

where the normalizing factor Z, known as the *evidence* or *marginal likelihood*, shall ensure that this distribution integrates to 1:

$$Z = \int_{\mathcal{D}_{\boldsymbol{\Theta}}} \mathcal{L}(\boldsymbol{\theta};\, \mathcal{X})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{1.6}$$

The posterior distribution in Eq. (1.5) summarizes the information inferred about the hyperparameters by combining the prior knowledge and the observed data. In this sense, the posterior $\pi(\boldsymbol{\theta}|\mathcal{X})$ is an "update" of the prior distribution $\pi(\boldsymbol{\theta})$.

The practical computation of posterior distributions is nothing but trivial. Particular analytical solutions exist only for specific distributions of $\boldsymbol{X}$ and ad-hoc choices on the prior

[1]Bayes' theorem is an elementary result of probability theory that reads as follows: for two events A and B with non-zero probabilities, the following equation holds: $\mathbb{P}\left(B|A\right) = \dfrac{\mathbb{P}\left(A|B\right)\;\mathbb{P}\left(B\right)}{\mathbb{P}\left(A\right)}$, where $\mathbb{P}\left(A|B\right)$ denotes the *conditional probability of A given B*.

distribution on $\boldsymbol{\Theta}$, the so-called *conjugate distributions* (Gelman et al., 2014). In the general case though, sampling methods shall be used, such as *Markov Chain Monte Carlo* simulation (see Section 1.3 for details).

In practical applications related to uncertainty quantification, the purpose of Bayesian inference may not just be to find the posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{X})$, but also to propose "the best distribution" for the parameters $\boldsymbol{X}$ given the information and data at hand. One possibility is to select a *point estimator* $\hat{\boldsymbol{\theta}}_0$, *i.e.* a particular value from the posterior distribution of $\pi(\boldsymbol{\theta}|\mathcal{X})$. Then the point posterior distribution of $\boldsymbol{X}$ simply reads:

$$\pi(\boldsymbol{x}|\mathcal{X}) \stackrel{\text{def}}{=} \pi(\boldsymbol{x}|\hat{\boldsymbol{\theta}}_0). \tag{1.7}$$

Popular choices for $\hat{\boldsymbol{\theta}}_0$ are the *posterior mean*, which is the mean value of the posterior distribution (1.5) and the *posterior mode*, a.k.a. *maximum a posteriori* (MAP), which is the mode of this posterior distribution. Such choices disregard the remaining estimation uncertainty in the hyperparameters.

In contrast, it is also possible to incorporate the uncertainty on $\boldsymbol{\theta}$ into the prior and posterior assessment of $\boldsymbol{X}$. This results in the so-called *predictive distributions*. The *prior predictive* distribution π'_{pred} of $\boldsymbol{X}$ is obtained by "averaging" the conditional prior distribution $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ in Eq.(1.1) against the prior distribution $\pi(\boldsymbol{\theta})$:

$$\pi'_{pred}(\boldsymbol{x}) \stackrel{\text{def}}{=} \int_{\mathcal{D}_{\boldsymbol{\Theta}}} \pi(\boldsymbol{x}|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{1.8}$$

The *posterior predictive* distribution $\pi''_{pred}(\boldsymbol{x}|\mathcal{X})$ of $\boldsymbol{X}$ is obtained by "averaging" the conditional distribution $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ in Eq.(1.1) over the posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{X})$ in Eq.(1.5):

$$\pi''_{pred}(\boldsymbol{x}|\mathcal{X}) \stackrel{\text{def}}{=} \int_{\mathcal{D}_{\boldsymbol{\Theta}}} \pi(\boldsymbol{x}|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta}|\mathcal{X})\, \mathrm{d}\boldsymbol{\theta} = \frac{1}{Z} \int_{\mathcal{D}_{\boldsymbol{\Theta}}} \pi(\boldsymbol{x}|\boldsymbol{\theta})\, \mathcal{L}(\boldsymbol{\theta};\, \mathcal{X})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}. \tag{1.9}$$

1.2 Bayesian model calibration and inverse problems

1.2.1 A wide class of problems sharing the same methods

Let us consider a computational model $\mathcal{M}$ that allows the analyst to predict certain *quantities of interest* gathered in a vector $\boldsymbol{y} \in \mathbb{R}^{N_{out}}$ as a function of input parameters $\boldsymbol{x}$:

$$\mathcal{M}:\ \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}} \subset \mathbb{R}^M \mapsto \boldsymbol{y} = \mathcal{M}(\boldsymbol{x}) \in \mathbb{R}^{N_{out}}. \tag{1.10}$$

Such models are commonly established based on first principles in engineering sciences, *e.g.* mechanics, electromagnetism, fluid dynamics, etc. but also natural sciences (geophysics, wave propagation, etc.). Although analytical models with closed-form equations may be used, the vast majority of computational models are black-box computer codes that solve the underlying ordinary (resp. partial) differential equations that govern the system of interest.

When input parameters $\{x_i, i = 1, \ldots, M\}$ are not measurable directly, one resorts to measuring the quantities of interest. Let us consider N independent measurement $\boldsymbol{y}_i \in \mathbb{R}^{N_{out}}$ gathered in a data set $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$. In the context of computational modeling and uncertainty quantification, two main classes of applications benefit from Bayesian inference, namely *model calibration* and *inverse problems*. The two classes are very much related to each other and they share the same problem statement and solution techniques, but they differ in their final focus.

On the one hand, *Bayesian inversion* focuses on the identification of the values of the input parameters $\boldsymbol{x}$, rather than on the model used to infer them. For this reason it is also known as *Bayesian inverse modelling*: instead of predicting a model response from a set of input parameters, the latter are inferred from a set of observed model responses $\mathcal{Y}$. This is the typical usage scenario in tomographic imaging applications, where the goal is to identify the set of input parameters that caused a specific set of observations. The resulting inferred input parameters can then be directly used to identify anomalies (e.g. position and length of cracks in pressure vessels), or to identify properties of interest (e.g. location and volume of subsurface oil reservoirs).

On the other hand, *Bayesian model calibration* focuses on identifying the input parameters of a computational model to allow one to recover the observations in $\mathcal{Y}$. A common scenario in this respect is identifying unknown properties of key components of a complex system, based on their observed response to controlled external loads in a laboratory experiment. Through this procedure, known as calibration, the inferred values (and possibly the uncertainty of estimation) can then be used to predict the response of the same system to different external loads, or even to design different systems sharing the same calibrated model component. This approach is at the basis of the so-called *verification and validation under uncertainty* paradigm (VVUQ) that is gaining momentum in the engineering practice worldwide (Oberkampf et al., 2004; Oberkampf and Roy, 2010; Hu and Orient, 2016).

1.2.2 Simple problems with known discrepancy parameters

Regardless of the specific context (model calibration or inversion), all Bayesian inverse problems share the same ingredients: a computational *forward* model $\mathcal{M}$, a set of input parameters $\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}}$ that need to be inferred, and a set of experimental data $\mathcal{Y}$.

The forward model $\boldsymbol{x} \mapsto \mathcal{M}(\boldsymbol{x})$ is a mathematical representation of the system under consideration. All models are always simplifications of the real world. Thus, to connect model predictions $\tilde{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x})$ to the observations $\mathcal{Y}$, a *discrepancy term* shall be introduced. We consider the following well-established format:

$$\boldsymbol{y} = \mathcal{M}(\boldsymbol{x}) + \boldsymbol{\varepsilon}, \tag{1.11}$$

where $\boldsymbol{\varepsilon}$ is the term that describes the discrepancy between an experimental observation $\boldsymbol{y}$

and the model prediction. For the sake of simplicity we consider it as an *additive Gaussian discrepancy*[2] with zero mean value and given covariance matrix $\boldsymbol{\Sigma}$ in this introduction:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\varepsilon \,|\mathbf{0}, \boldsymbol{\Sigma}). \tag{1.12}$$

This discrepancy term represents in practice the effects of *measurement error* (on $\boldsymbol{y}_i \in \mathcal{Y}$) and *model inaccuracy*. In the above equation, again for the sake of simplicity, this term is supposed to have a zero mean, but it could more generally include a model bias term.

In the context of model inversion or calibration, the goal is to find the optimal values of the input parameters $\boldsymbol{x}$ that allow one to fit the model predictions to the observations. In this respect the epistemic uncertainty (lack of knowledge) on the input parameters is modelled by considering the input parameters as a random vector $\boldsymbol{X} \sim \pi(\boldsymbol{x})$, with given prior distribution as in Eq. (1.2).

Note: In this section, the measurement data is denoted by $\mathcal{Y}$, which plays the role of $\mathcal{X}$ in Section 1.1. In contrast, the parameters to infer are the input parameters $\boldsymbol{X}$ of the computational model $\mathcal{M}$, which plays the role of hyperparameters $\boldsymbol{\Theta}$ in Section 1.1.

From Eqs. (1.11), (1.12) a particular measurement point $\boldsymbol{y}_i \in \mathcal{Y}$ is a realization of a Gaussian distribution with mean value $\mathcal{M}(\boldsymbol{x})$ and covariance matrix $\boldsymbol{\Sigma}$. The likelihood function, as a function of the parameters $\boldsymbol{x}$, thus reads:

$$\mathcal{L}(\boldsymbol{x}; \boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\mathcal{M}(\boldsymbol{x}), \boldsymbol{\Sigma}). \tag{1.13}$$

If N *independent* measurements $\boldsymbol{y}_i$ are available and gathered in the data set $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, the likelihood can be written as

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}; \mathcal{Y}) &= \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\mathcal{M}(\boldsymbol{x}), \boldsymbol{\Sigma}) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^{N_{out}} \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x})\right)^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x})\right)\right). \end{aligned} \tag{1.14}$$

Combining the prior $\pi(\boldsymbol{x})$ and the above likelihood $\mathcal{L}(\boldsymbol{x}; \mathcal{Y})$, the posterior distribution in Eq. (1.5) establishes the solution of the inverse problem:

$$\pi(\boldsymbol{x}|\mathcal{Y}) = \frac{1}{Z} \pi(\boldsymbol{x}) \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\mathcal{M}(\boldsymbol{x}), \boldsymbol{\Sigma}) \tag{1.15}$$

[2] It is noted here that this simple *Gaussian discrepancy* assumption is only one out of many possible models. In a more general setting, other distributions for the discrepancy are used as well (Schoups and Vrugt, 2010). Due to the widespread use of the additive Gaussian models in engineering disciplines, the discussion is limited to this discrepancy type.

It summarizes the collected information about the unknowns $\boldsymbol{x}$ after conditioning on the data. In this sense, the data are *inverted* through the forward model $\mathcal{M}$.

1.2.3 General case: discrepancy with unknown parameters

In many practical situations it is unrealistic to assume that the residual covariance matrix $\boldsymbol{\Sigma}$ in Eq. (1.12) is perfectly known. However, by parametrizing the matrix as $\boldsymbol{\Sigma}(\boldsymbol{x}_\varepsilon)$, one may treat its parameters $\boldsymbol{x}_\varepsilon$ as additional unknowns that can be inferred jointly with the input parameters of $\mathcal{M}$. In this setting the parameter vector is defined by $\boldsymbol{x} = (\boldsymbol{x}_\mathcal{M}, \boldsymbol{x}_\varepsilon)$, *i.e.* a combined vector of forward model parameters $\boldsymbol{x}_\mathcal{M}$ and discrepancy parameters $\boldsymbol{x}_\varepsilon$. For the sake of simplicity, consider a diagonal covariance matrix of the form $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_{N_{out}}$ with unknown residual variances $\sigma^2 = \mathrm{Var}[\varepsilon_i]$. This way, the discrepancy parameter vector reduces to a single scalar, *i.e.* $\boldsymbol{x}_\varepsilon \equiv \sigma^2$.

Assuming that one can elicit a prior distribution $\pi(\boldsymbol{x}_\varepsilon)$ for the unknown variance σ^2, and by treating the uncertain model and the discrepancy parameters as being priorly independent, one gets the joint prior distribution

$$\pi(\boldsymbol{x}) = \pi(\boldsymbol{x}_\mathcal{M})\pi(\sigma^2). \tag{1.16}$$

The likelihood is then given as

$$\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{Y}) = \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi\sigma^2)^{N_{out}}}} \exp\left(-\frac{1}{2\sigma^2}\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)^\mathsf{T}\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)\right). \tag{1.17}$$

With the prior distribution in Eq. (1.16) and likelihood function in Eq. (1.17), the corresponding posterior distribution can then again be computed as in Eq. (1.15):

$$\pi(\boldsymbol{x}_\mathcal{M}, \sigma^2 | \mathcal{Y}) = \frac{1}{Z}\pi(\boldsymbol{x}_\mathcal{M})\pi(\sigma^2)\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{Y}). \tag{1.18}$$

This posterior distribution summarizes the updated information about the unknowns $(\boldsymbol{x}_\mathcal{M}, \boldsymbol{x}_\varepsilon \equiv \sigma^2)$ after conditioning on the data $\mathcal{Y}$. One may then extract the marginals $\pi(x_{\mathcal{M},i}|\mathcal{Y})$ of individual forward model inputs or the marginal $\pi(\boldsymbol{x}_\varepsilon|\mathcal{Y}) \equiv \pi(\sigma^2|\mathcal{Y})$ of the residual variance.

More generally, any parametrization $\boldsymbol{\Sigma}(\boldsymbol{x}_\varepsilon)$ of the positive definite covariance matrix can be incorporated into the Bayesian analysis. This just requires the specification of a prior distribution $\pi(\boldsymbol{x}_\varepsilon)$ and construction of a likelihood function of the more general form:

$$\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \boldsymbol{x}_\varepsilon; \mathcal{Y}) = \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi)^{N_{out}} \det(\boldsymbol{\Sigma}(\boldsymbol{x}_\varepsilon))}} \exp\left(-\frac{1}{2}\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)^\mathsf{T} \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_\varepsilon)\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)\right). \tag{1.19}$$

1.2.4 Inverse solution

The posterior distribution of the parameters computed by Eq. (1.15), is often characterized through its first statistical moments. The posterior mean vector is given as

$$\mathbb{E}\left[\boldsymbol{X}\,|\,\mathcal{Y}\right] = \int\limits_{\mathcal{D}_{\boldsymbol{X}}} \boldsymbol{x}\,\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.20}$$

It can be considered as a *point estimate* of the unknown parameter values. The estimation uncertainty can be quantified through the *posterior covariance matrix*

$$\mathrm{Cov}[\boldsymbol{X}\,|\,\mathcal{Y}] = \int\limits_{\mathcal{D}_{\boldsymbol{X}}} (\boldsymbol{x} - \mathbb{E}\left[\boldsymbol{X}\,|\,\mathcal{Y}\right])(\boldsymbol{x} - \mathbb{E}\left[\boldsymbol{X}\,|\,\mathcal{Y}\right])^{\intercal}\,\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.21}$$

One may also be interested in the posterior marginals. The marginal of a specific parameter x_i with $i \in \{1, \ldots, M\}$ can be computed by integration over the other components (sometimes called nuisance parameters):

$$\pi(x_i|\mathcal{Y}) = \int\limits_{\mathcal{D}_{\boldsymbol{X}_{\sim i}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\sim i}, \tag{1.22}$$

where $\boldsymbol{x}_{\sim i}$ refers to the parameter vector $\boldsymbol{x}$ excluding the i-th parameter x_i.

In practical inverse problems, the posterior distribution $\pi(\boldsymbol{x}|\mathcal{Y})$ can also be an intermediate quantity that is further used for computing the conditional expectation of a certain *quantity of interest* (QoI) $h\colon \mathcal{D}_{\boldsymbol{X}} \to \mathbb{R}$. This can be anything from a simple analytical function to complex secondary models. This conditional expectation is simply the expectation of $h(\boldsymbol{X})$ under the posterior distribution and is computed by the integral

$$\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] = \int\limits_{\mathcal{D}_{\boldsymbol{X}}} h(\boldsymbol{x})\,\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.23}$$

1.2.5 Model predictions

To assess the predictive capabilities of a computational model, the Bayesian inference framework offers the possibility to compute *predictive distributions*, as seen in Section 1.1. With the definition of the likelihood function the *prior predictive* distribution from Eq. (1.8) can also be written as

$$\pi'_{pred}(\boldsymbol{y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \mathcal{L}(\boldsymbol{x};\boldsymbol{y})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{1.24}$$

It is not necessary in practice to explicitly compute the above integral: samples from this distribution are obtained by sampling first $\boldsymbol{x}$'s according to $\pi(\boldsymbol{x})$, then sampling the following distribution conditioned on $\mathcal{M}(\boldsymbol{x})$:

$$\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{y}|\mathcal{M}(\boldsymbol{x}), \boldsymbol{\Sigma}) \tag{1.25}$$

This corresponds to simply sampling a realization of the discrepancy term ε according to $\mathcal{N}(\varepsilon|\boldsymbol{0}, \boldsymbol{\Sigma})$ and adding it to $\mathcal{M}(\boldsymbol{x})$.

The *posterior predictive* distribution (see Eq. (1.9)) can be similarly written as

$$\pi''_{pred}(\boldsymbol{y}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \mathcal{L}(\boldsymbol{x}; \boldsymbol{y})\pi(\boldsymbol{x}|\mathcal{Y})\, \mathrm{d}\boldsymbol{x}. \tag{1.26}$$

A sample from the posterior predictive distribution is obtained by drawing $\boldsymbol{x}$'s according to $\pi(\boldsymbol{x}|\mathcal{Y})$, then evaluating $\mathcal{M}(\boldsymbol{x})$ and adding an independently sampled discrepancy term ε.

1.3 Markov chain Monte Carlo

Posterior distribution as in Eq. (1.15) do not have a closed-form solution in practice. One widespread option to solve inverse problems relies upon *Markov chain Monte Carlo* (MCMC) simulations (Robert and Casella, 2004; Liu, 2004).

The basic idea of MCMC simulations is to construct a Markov chain $(\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots)$ over the prior support $\mathcal{D}_{\boldsymbol{X}}$ with an invariant distribution that equals the posterior distribution of interest. Markov chains can be uniquely defined by their transition probability $\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})$ from the step $\boldsymbol{x}^{(t)}$ of the chain at iteration t to the step $\boldsymbol{x}^{(t+1)}$ at the subsequent iteration $t+1$. Then, the posterior is the invariant distribution of the Markov chain if the specified transition probability fulfils the so-called *detailed balance* condition:

$$\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\, \mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}) = \pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y})\, \mathcal{K}(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t+1)}). \tag{1.27}$$

This condition ensures that the Markov chain is reversible, *i.e.*, that the probability to be at $\boldsymbol{x}^{(t)}$ and move to $\boldsymbol{x}^{(t+1)}$ is equal the probability to be at $\boldsymbol{x}^{(t+1)}$ and move to $\boldsymbol{x}^{(t)}$. By integrating this condition over $\mathrm{d}\boldsymbol{x}^{(t)}$, it can be shown that the invariant distribution of the Markov chain is the posterior distribution:

$$\pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\, \mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})\, \mathrm{d}\boldsymbol{x}^{(t)}. \tag{1.28}$$

A Markov chain constructed this way can be used to approximate the expectation in Eq. (1.23) as the iteration average of the $T+1$ generated sample points $\boldsymbol{x}^{(t)}$

$$\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] \approx \frac{1}{T}\sum_{t=1}^{T} h(\boldsymbol{x}^{(t)}). \tag{1.29}$$

A prototypical technique to ensure that this equation is fulfilled is the *Metropolis–Hastings* (MH) algorithm (Metropolis et al., 1953; Hastings, 1970) that is based on proposing and subsequently accepting or rejecting candidate points. In the following, the original MH algorithm and three other popular MCMC algorithms are discussed as techniques to efficiently sample from the posterior distribution.

1.3.1 Metropolis–Hastings algorithm

In the *Metropolis–Hastings algorithm* (MH), a chain is initialized at a certain seed point $\boldsymbol{x}^{(0)} \in \mathcal{D}_{\boldsymbol{X}}$ from the admissible domain. At iteration t from the current point $\boldsymbol{x}^{(t)}$, one then draws a candidate point $\boldsymbol{x}^{(\star)}$ from a proposal distribution $p(\boldsymbol{x}^{(\star)}|\boldsymbol{x}^{(t)})$. Subsequently, the candidate is accepted (*i.e.*, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(\star)}$) with probability:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})\, p(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(\star)})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\, p(\boldsymbol{x}^{(\star)}|\boldsymbol{x}^{(t)})}\right\}, \tag{1.30}$$

and rejected otherwise (*i.e.*, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$). With this procedure, the transition probability fulfills Eq. (1.28) and the chain of sample points eventually follows the posterior distribution. These sample points can then, for example, be used to approximate expectations under the posterior distribution as shown in Eq. (1.29).

It is advantageous that the model evidence cancels out from the acceptance probability in Eq. (1.30). Hence, the MH algorithm only calls for pointwise evaluations of the *unnormalized* posterior density $\pi(\boldsymbol{x}|\mathcal{Y}) \propto \mathcal{L}(\boldsymbol{x};\mathcal{Y})\pi(\boldsymbol{x})$. This avoids the calculation of the often intractable integral in Eq. (1.6).

The original *Metropolis algorithm* is based on a symmetrical proposal distribution with $p(\boldsymbol{x}^{(\star)}|\boldsymbol{x}^{(t)}) = p(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(\star)})$. In this case, the acceptance probability in Eq. (1.30) reduces to:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})}\right\}. \tag{1.31}$$

A commonly used symmetric proposal is the Gaussian distribution $p(\boldsymbol{x}|\boldsymbol{x}^{(t)}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}^{(t)}, \boldsymbol{\Sigma}_p)$ centered around the current step $\boldsymbol{x}^{(t)}$ with a covariance matrix $\boldsymbol{\Sigma}_p$. This proposal corresponds to the classical *random walk Metropolis* (RWM) sampler. Note that with a symmetric proposal distribution, a candidate $\boldsymbol{x}^{(\star)}$ is always accepted if one has $\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y}) \geq \pi(\boldsymbol{x}^{(t)}|\mathcal{Y})$, *i.e.*, if it is more likely to belong to the posterior distribution than $\boldsymbol{x}^{(t)}$. For $\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y}) < \pi(\boldsymbol{x}^{(t)}|\mathcal{Y})$, however, the proposed candidate is not rejected, but accepted only with probability $\alpha = \pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})/\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})$.

In practice, in order to accept and reject the proposed candidates with the probability in Eq. (1.30) or Eq. (1.31), one usually samples a random variate $u \in [0,1]$ according to a standard uniform distribution $U \sim \mathcal{U}(u|0,1)$ and compares it to the ratio α from Eq. (1.30).

If $\alpha \geq u$, then the proposed candidate is accepted. Otherwise, in the event that $\alpha < u$, the proposed candidate is rejected.

1.3.2 Adaptive Metropolis algorithm

A practical weak point of the standard Metropolis–Hastings algorithm is the need to choose a proposal distribution $p(\boldsymbol{x}^{(\star)}|\boldsymbol{x}^{(t)})$. Ideally, this distribution should be as similar to the posterior distribution as possible. In most applications, however, the posterior shape is not known a priori. Moreover, a badly chosen proposal distribution significantly affects the MCMC performance up to the point where the MCMC algorithm fails because it does not accept any proposed candidates. This typically occurs in high dimensions with strongly correlated posterior distributions.

A workaround was proposed in Haario et al. (2001). In this approach known as *adaptive Metropolis algorithm* (AM), the Gaussian proposal distribution of the classic Metropolis algorithm (see Section 1.3.1) is tuned during the sampling procedure based on previously generated samples. The algorithm starts as a standard random walk Metropolis algorithm with an initial proposal covariance $\boldsymbol{C}_0$. Following a starting period t_0, the proposal distribution covariance matrix is updated to:

$$\boldsymbol{C}(t+1) = \begin{cases} \boldsymbol{C}_0, & t+1 \leq t_0, \\ s_d(M)\tilde{\boldsymbol{C}}(t), & t+1 > t_0, \end{cases} \tag{1.32}$$

where $s_d(M) = \frac{2.38^2}{M}$ is a tuning parameter that depends only on the dimension of the problem (Gelman et al., 1996). The empirical covariance $\tilde{\boldsymbol{C}}(t)$ is estimated based on available sample points generated up to step t and can be computed through:

$$\tilde{\boldsymbol{C}}(t) = \frac{1}{t-1}\left(\sum_{i=1}^{t}(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(t)})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(t)})^{\intercal}\right), \quad \text{where} \quad \bar{\boldsymbol{x}}^{(t)} = \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{x}^{(i)}. \tag{1.33}$$

In practical applications, an iterative approach can be used to compute the empirical covariance matrices $\tilde{\boldsymbol{C}}(t)$ with a negligible computational burden (*i.e.*, updated from one step t to the next) (Haario et al., 2001). To avoid singularity of this estimated covariance matrix, a small constant ϵ is added to the diagonal of its correlation matrix. This modified empirical covariance matrix $\boldsymbol{C}^{\star}(t)$ is then used in the definition of the Gaussian proposal distribution centered at the current step of the Markov chain:

$$p(\boldsymbol{x}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}^{(t)}, \boldsymbol{C}^{\star}(t)). \tag{1.34}$$

After drawing a candidate point $\boldsymbol{x}^{(\star)}$ from the proposal distribution $p(\boldsymbol{x}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$, this candidate is accepted with probability:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})}\right\}, \tag{1.35}$$

which is the acceptance probability already defined in Eq. (1.30) for symmetric proposal distributions. As the transition probability $\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(t)})$ at each step t depends on all previous steps through the proposal distribution, the generated chain is non-Markovian and it does not fulfil the symmetry condition from Eq. (1.27). Nonetheless, it was shown in Haario et al. (2001) that the generated sample points can be used to approximate posterior properties by Eq. (1.29).

1.3.3 Hamiltonian Monte Carlo algorithm

Instead of purely relying on a random walk to sample from the posterior distribution, *Hamiltonian Monte Carlo algorithms* (HMC) exploit the gradient of the posterior distribution to construct a Markov chain using Hamiltonian dynamics. The connection between Hamiltonian dynamics and MCMC algorithms was originally established in Duane et al. (1987) and a more detailed description of the HMC algorithm can be found in Neal (2011); Nagel and Sudret (2016).

The core idea of the algorithm lies in randomly assigning a momentum to a particle and letting it travel over a potential surface. This can be formalized by defining a potential $U(\boldsymbol{x})$ and a kinetic energy function $K(\boldsymbol{p})$:

$$U(\boldsymbol{x}) = -\log\left(\pi(\boldsymbol{x})\mathcal{L}(\boldsymbol{x};\mathcal{Y})\right), \qquad K(\boldsymbol{p}) = \frac{\boldsymbol{p}^\intercal \boldsymbol{M}^{-1}\boldsymbol{p}}{2}, \tag{1.36}$$

where $\boldsymbol{p} = (p_1, \ldots, p_M)^\intercal$ is the momentum vector and $\boldsymbol{M}$ is a mass matrix for the particle, which is often assumed to be a diagonal or simply a scaled identity matrix. This mass property can be considered a tuning parameter of the algorithm.

The *Hamiltonian Monte Carlo algorithm* then uses the energy functions from Eq. (1.36) to define the Hamiltonian:

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{p}) = U(\boldsymbol{x}) + K(\boldsymbol{p}). \tag{1.37}$$

The Hamiltonian captures the total energy of a particle at position $\boldsymbol{x}$ with a given momentum $\boldsymbol{p}$. According to Hamiltonian dynamics, the movement of a particle in this system can be calculated by (where the dot denotes the time derivative):

$$\dot{x}_i = \frac{\partial\mathcal{H}(\boldsymbol{x}, \boldsymbol{p})}{\partial p_i}, \qquad \dot{p}_i = \frac{\partial\mathcal{H}(\boldsymbol{x}, \boldsymbol{p})}{\partial x_i}, \quad \text{for} \quad i = 1, \ldots, M. \tag{1.38}$$

These equations can be solved by the well-known *leapfrog integration algorithm* (Neal, 2011).

It starts out at the current position $\boldsymbol{x}(0) = \boldsymbol{x}^{(t)}$ of the particle and a given momentum $\boldsymbol{p}(0)$ drawn from the distribution:

$$\boldsymbol{p}(0) \sim \mathcal{N}(\boldsymbol{p}|\boldsymbol{0}, \boldsymbol{I}_M). \tag{1.39}$$

The leapfrog algorithm then solves the Hamiltonian equations for a total duration τ, using a discrete timestep size of $\frac{\tau}{N_\tau}$. The result is the proposal position $\boldsymbol{x}(\tau)$ and momentum $\boldsymbol{p}(\tau)$ of the particle at time τ.

Given the new location and momentum of the particle, one again computes the Hamiltonian and accepts the new candidate with probability:

$$\alpha\left(\boldsymbol{x}(\tau), \boldsymbol{p}(\tau), \boldsymbol{x}(0), \boldsymbol{p}(0)\right) = \min\left\{1, \exp\left(\mathcal{H}(\boldsymbol{x}(0), \boldsymbol{p}(0)) - \mathcal{H}(\boldsymbol{x}(\tau), \boldsymbol{p}(\tau))\right)\right\}. \tag{1.40}$$

If the new candidate is accepted, the next position of the Markov chain is set to $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}(\tau)$; if it is rejected, it is set to $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$. It was shown in Neal (2011) that the invariant distribution of the generated chain is the posterior distribution.

As the Hamiltonian is generally invariant between the initial point $\mathcal{H}(\boldsymbol{x}(0), \boldsymbol{p}(0))$ and last point $\mathcal{H}(\boldsymbol{x}(\tau), \boldsymbol{p}(\tau))$ of the dynamic simulation, the acceptance probability is theoretically always one (*i.e.*, no proposal points are rejected). Due to the numerical integration carried out by the leapfrog method this is only true approximately. Nevertheless, the acceptance probability of Hamiltonian Monte Carlo is typically close to one.

1.3.4 Affine invariant ensemble algorithm

Most MCMC algorithms perform poorly when the target (*i.e.*, posterior) distribution shows strong correlation between the parameters. The performance of these algorithms can typically only be improved by considerable amount of tuning. The *affine invariant ensemble algorithm* (AIES) originally presented in Goodman and Weare (2010) alleviates this problem. It has the desirable property of being invariant to affine transformations of the target distribution. This means that if there exists an affine transformation of the *difficult-to-sample* (by standard MCMC methods) target distribution to an *easier-to-sample* target distribution, AIES samples both distributions equally easily without explicitly requiring this affine transformation.

The algorithm simultaneously runs an ensemble of C Markov chains $\{\mathcal{X}_1, \ldots, \mathcal{X}_C\}$, where each chain is called a *walker*. The Markov chain locations $\boldsymbol{x}_i$ are updated walker by walker. One such update consists of picking randomly a conjugate walker $\boldsymbol{x}_j^{(t)}$ from the set of walkers excluding the current i-th walker $(j \neq i)$.

The affine invariance property is achieved by generating proposals according to a so-called *stretch move*. This refers to proposing a new candidate by:

$$\boldsymbol{x}_i^{(\star)} = \boldsymbol{x}_i^{(t)} + Z\left(\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}\right), \quad \text{where} \quad Z \sim p(z) = \begin{cases} \frac{1}{\sqrt{z}(2\sqrt{a}-\frac{2}{\sqrt{a}})} & \text{if } z \in [1/a, a]\,, \\ 0 & \text{otherwise.} \end{cases} \tag{1.41}$$

This requires sampling from the distribution $p(z)$ defined by the tuning parameter $a > 1$. The candidate $\boldsymbol{x}_i^{(\star)}$ is then accepted as the new location of the i-th walker with probability:

$$\alpha\left(\boldsymbol{x}_i^{(\star)}, \boldsymbol{x}_i^{(t)}, z\right) = \min\left\{1, z^{M-1}\frac{\pi(\boldsymbol{x}_i^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}_i^{(t)}|\mathcal{Y})}\right\}. \tag{1.42}$$

This is repeated for all C walkers in the ensemble. The resulting chains fulfill the detailed balance condition and the generated sample can thus be combined to estimate expectations under the posterior distribution using Eq. (1.29). A practical advantage of the AIES algorithm is that it only has a single scalar tuning parameter a, which is often set to $a = 2$ (Goodman and Weare, 2010; Allison and Dunkley, 2013; Wicaksono, 2017). On the other hand, due to its sequential nature, the algorithm cannot be parallelized which makes it comparably slow.

1.3.5 Assessing convergence in MCMC simulations

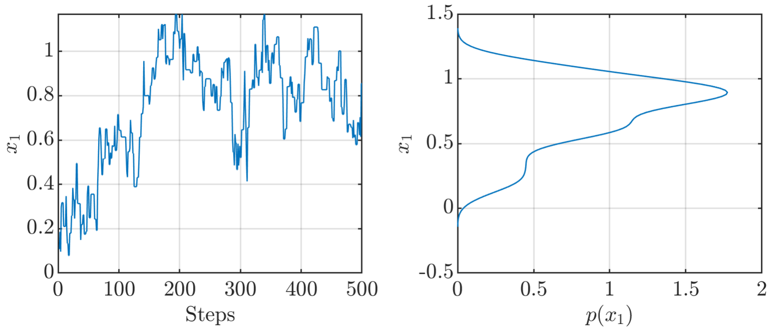
All MCMC algorithms produce chains of sample points that will eventually follow the posterior distribution. In practice, however, one is forced to make decisions about convergence based on a finite number of sample points. As MCMC algorithms lack a convergence criterion, numerous heuristics have been developed to allow practitioners to assess the quality of the produced Markov chains.

1.3.5.1 Acceptance rate

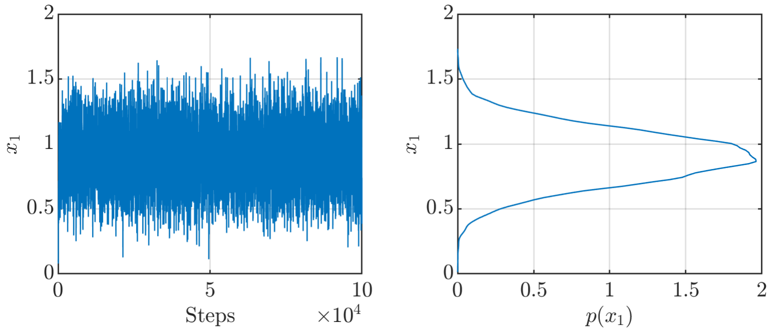
The acceptance rate r_a gives a quantitative indication of how many proposed sample points were accepted. It can be simply computed as the ratio between the number of accepted points and the total number of iterations $T + 1$.

In MCMC algorithms, the acceptance rate depends mostly on their tuning parameters. For the Metropolis algorithm with a Gaussian proposal, the optimal acceptance rate is shown to approach $r_a = 0.23$ as $M \to \infty$ (Roberts et al., 1997). For Hamiltonian Monte Carlo algorithms, it was already mentioned that r_a is typically close to one. It is difficult to assess the quality of a generated MCMC chain purely based on the computed acceptance rate, but it can serve as an indicator of a badly tuned algorithm.

In practical applications, acceptance rates close to one (except in the Hamiltonian Monte Carlo algorithm) typically indicate that the proposal distribution does not sufficiently explore



(a) After 500 steps



(b) After 10^5 steps

Figure 1: Trace plot and corresponding KDE at two different iterations of the a chain.

the target distribution. Acceptance rates close to zero indicate instead that the proposed candidate points are in low probability regions. The most common reasons for this are too wide proposal distributions or proposal distributions that do not sufficiently resemble the target distributions.

1.3.5.2 Trace and density plots

Trace plots show the evolution of a Markov chain. As chains are typically initialized at random points, the evolution of an MCMC chain can give valuable insights about convergence. Trace plots are typically assessed visually for each dimension individually.

A sample generated by the chain should eventually be distributed according to the posterior distribution. A kernel density estimation (KDE) scheme (Wand and Jones, 1995) can thus be employed to obtain an approximation of the posterior marginal. If the chain has reached its steady state, this KDE of the posterior marginal should not change considerably with further iterations.

An example of a trace plot with a corresponding KDE is displayed in Figure 1. It can be clearly seen that the chain has not reached its steady state after 500 steps (Figure 1a) whereas it has after 10^5 steps (Figure 1b).

1.3.5.3 Gelman-Rubin diagnostics

A quantitative approach to assess convergence was introduced by Gelman and Rubin (1992) and later generalized by Brooks and Gelman (1998). The idea presented there is to compare a set of C independent Markov chains that were initiated at different seed points. If the chains are converged, the empirical second moments computed from the individual chains should be the same as the empirical second moments computed from combining the samples from all C chains.

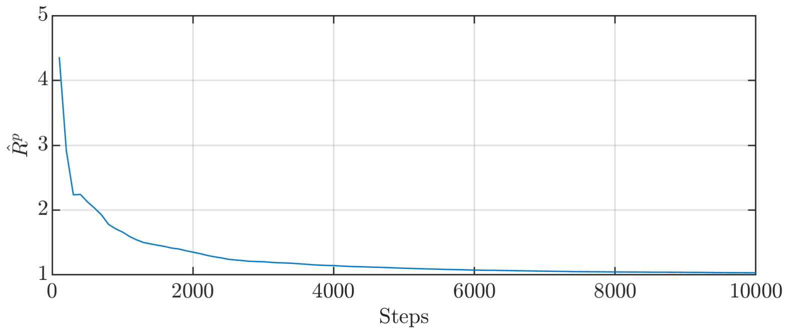


Figure 2: Convergence of the MPSRF $\hat{R}^p$.

More formally, let $\{\mathcal{X}_1, \ldots, \mathcal{X}_C\}$ be the C chains run in parallel from different seed points $\boldsymbol{x}_i^{(0)}$, where each chain $\mathcal{X}_i = (\boldsymbol{x}_i^{(0)}, \ldots, \boldsymbol{x}_i^{(T)})$ contains $T+1$ sample points with $\boldsymbol{x}_i^{(t)} \in \mathbb{R}^M$.

The Gelman-Rubin diagnostic requires the computation of two covariance matrices. The covariance matrix of the i-th chain is estimated by:

$$\boldsymbol{W}_i = \frac{1}{T} \sum_{t=0}^{T} \left(\boldsymbol{x}_i^{(t)} - \bar{\boldsymbol{x}}_i\right) \left(\boldsymbol{x}_i^{(t)} - \bar{\boldsymbol{x}}_i\right)^\intercal, \quad \bar{\boldsymbol{x}}_i = \frac{1}{T+1} \sum_{t=0}^{T} \boldsymbol{x}_i^{(t)}. \tag{1.43}$$

The matrices for all C chains are then averaged to obtain the *within-sequence* covariance $\boldsymbol{W} \in \mathbb{R}^{M \times M}$:

$$\boldsymbol{W} = \frac{1}{C} \sum_{i=1}^{C} \boldsymbol{W}_i. \tag{1.44}$$

The second matrix required is the so-called *between-sequence* variance $\boldsymbol{B} \in \mathbb{R}^{M \times M}$. It captures the covariance between the individual MCMC chains and is estimated as:

$$\boldsymbol{B} = \frac{1}{C-1} \sum_{i=1}^{C} \left(\bar{\boldsymbol{x}}_i - \bar{\bar{\boldsymbol{x}}}\right) \left(\bar{\boldsymbol{x}}_i - \bar{\bar{\boldsymbol{x}}}\right)^\intercal, \quad \text{where:} \tag{1.45}$$

$$\bar{\bar{\boldsymbol{x}}} = \frac{1}{C(T+1)} \sum_{i=1}^{C} \sum_{t=0}^{T} \boldsymbol{x}_i^{(t)} \tag{1.46}$$

is the average of all $(T+1)$ states of the C chains. To estimate the difference between the *within-sequence* covariance estimate $\boldsymbol{W}$ and the *between-sequence* covariance estimate $\boldsymbol{B}$, the following *multivariate potential scale reduction factor* (MPSRF) is proposed in Brooks and Gelman (1998):

$$\hat{R}^p = \frac{T}{T+1} + \left(\frac{C+1}{C}\right)\lambda_1, \tag{1.47}$$

where λ_1 is the largest eigenvalue of the symmetric positive definite matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$. This $\hat{R}^p$ approaches 1 (from above) with increasing convergence of the MCMC algorithm. This convergence is showcased for a sample MCMC chain in Figure 2. This method requires a set of independent, parallel MCMC chains.

1.3.5.4 Burn-in

Once an MCMC chain has reached its steady state, the generated sample follows the posterior distribution. When, however, a finite number of sample points is used to estimate posterior properties (e.g. moments), the sample points generated prior to convergence can *pollute* the estimation.

It is therefore common practice in practical MCMC applications to discard sample points that were generated prior to convergence (Brooks et al., 2011). This discarded fraction is called *burn-in*.

Chapter 2

Usage

In this chapter the implementation of Bayesian inversion as discussed in Chapter 1 is described. A simple engineering inverse problem is solved with UQLAB to exemplify the usage of the Bayesian inversion module. This simple example is then extended to treat more complex problems.

2.1 Reference problem: calibration of a simply supported beam model

We consider a simply-supported beam such as the one shown in Figure 3. The beam has a known rectangular cross-section of width b and height h and a known span of length L.

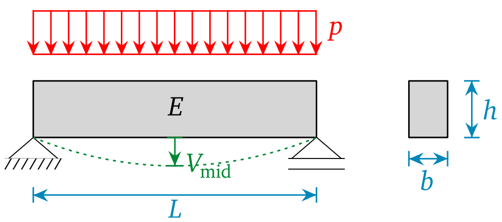


Figure 3: Simple beam bending test.

A set of $N = 5$ independent experiments are carried out with this beam, with the goal of inferring the unknown material stiffness, *i.e.* its Young's modulus E. In the experiments, the beam is subject to a constant distributed load p and the mid-span deflection V_{mid} is measured. The measurements are reported in Table 1. Due to measurement error, the measured deflections vary across experiments.

Table 1: Beam deflection: measured deflections.

Experiment	1	2	3	4	5
V_{mid} (mm)	12.84	13.12	12.13	12.19	12.67

The parameters (b, h, L, p) are considered known and their values are given in Table 2.

Table 2: Beam experiments: nominal values of the beam properties.

Variable	Nominal Value
b (m)	0.15
h (m)	0.3
L (m)	5
p (kN/m)	0.012

The analytical expression for the mid-span deflection V_{mid} of the beam according to the standard beam theory is:

$$V_{\text{mid}} = \frac{5}{32}\frac{pL^4}{Ebh^3}. \tag{2.1}$$

This simple equation serves as the forward model and relates the unknown Young's modulus to the measurable mid-span deflection.

Additionally, it is known from prior experiments that the Young's modulus of the material follows a lognormal distribution:

$$E \sim \mathcal{LN}(\lambda, \zeta), \quad \text{with} \quad \mu_E = 30\,000\ (\text{MPa}) \text{ and } \sigma_E = 4500\ (\text{MPa}). \tag{2.2}$$

In Bayesian inversion terms, the prior information on the model parameter $\boldsymbol{x}_{\mathcal{M}} \equiv E$ is $E \sim \mathcal{LN}(\lambda, \zeta)$. Due to a lack of more information, an unknown additive Gaussian experimental discrepancy model is assumed. As a weakly informative prior on the positive discrepancy variance $\boldsymbol{x}_{\varepsilon} \equiv \sigma^2$, a uniform distribution $\sigma^2 \sim \mathcal{U}(0, \mu^2_{V_{\text{mid}}})$ with $\mu_{V_{\text{mid}}}$ equal to the empirical mean of the observations given in Table 1.

2.2 Problem setup and solution

Solving an inverse problem with the Bayesian inverse module of UQLAB typically requires the specification of a prior distribution $\pi(\boldsymbol{x})$, N independent observations $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, where $\boldsymbol{y}_i \in \mathbb{R}^{N_{out}}$ and a forward model $\mathcal{M}$. All these ingredients are briefly discussed in this section.

2.2.1 Initialize UQLAB

The first step is to initialize UQLAB and fixing a random seed for reproducibility:

```
uqlab
rng(100)
```

2.2.2 Specify a prior distribution

The prior distribution of the model parameters is defined as an INPUT object by:

```
PriorOpts.Name = 'Model parameters prior';
PriorOpts.Marginals(1).Name = 'b';
PriorOpts.Marginals(1).Type = 'Constant';
PriorOpts.Marginals(1).Parameters = 0.15; % (m)

PriorOpts.Marginals(2).Name = 'h';
PriorOpts.Marginals(2).Type = 'Constant';
PriorOpts.Marginals(2).Parameters = 0.3; % (m)

PriorOpts.Marginals(3).Name = 'L';
PriorOpts.Marginals(3).Type = 'Constant';
PriorOpts.Marginals(3).Parameters = 5; % (m)

PriorOpts.Marginals(4).Name = 'E';
PriorOpts.Marginals(4).Type = 'Lognormal';
PriorOpts.Marginals(4).Moments = [30000 4500] ; % (MPa)

PriorOpts.Marginals(5).Name = 'p';
PriorOpts.Marginals(5).Type = 'Constant';
PriorOpts.Marginals(5).Parameters = 0.012; % (kN/m)

myPriorDist = uq_createInput(PriorOpts);
```

As the prior distribution is specified as an INPUT object, all the features of the UQLAB INPUT module (UQLAB User Manual – the INPUT module) can be used, *i.e.* constant and marginal distributions of any kind (including user-defined ones) can be used. Dependence may also be specified using copulas.

Note: The known parameters from Table 2 are defined as `Constant` input marginals and will not be considered during the calibration procedure, except when evaluating the forward model.

2.2.3 Create a forward model

The computational model, given in Eq. (2.1), is defined as a MATLAB `m-file` which is stored in UQLAB's distribution in the folder:

`Examples/SimpleTestFunctions/uq_SimplySupportedBeam.m`

A UQLAB MODEL is then created as:

```
ModelOpts.Name = 'Forward model';
ModelOpts.mFile = 'uq_SimplySupportedBeam';

myForwardModel = uq_createModel(ModelOpts);
```

For more details about the configuration options available for a MODEL object, please refer to the UQLAB User Manual – the MODEL module .

2.2.4 Provide measurements

The measurements $\mathcal{Y}$ are stored in an $N \times N_{out}$ matrix V_mid, where N_{out} is the number of model outputs:

```
V_mid = [12.84; 13.12; 12.13; 12.19; 12.67]/1000; % (m)

myData.Name = 'Beam mid-span deflection';
myData.y = V_mid;
```

Note: In the present case where $N_{out} = 1$, this matrix reduces to a column vector.

2.2.5 Perform the Bayesian inverse analysis

The options are then gathered in a MATLAB structure, here called BayesOpts:

```
BayesOpts.Type = 'Inversion';
BayesOpts.Data = myData;
```

The BayesOpts structure contains all information required to solve the inverse problem. If not explicitly specified by the user, by default the Bayesian inversion module uses the *last created* INPUT object (in this case myPriorDist) as a prior distribution and the last created MODEL object (in this case myForwardModel) as the forward model. Therefore, to perform the analysis, it is sufficient to create the corresponding ANALYSIS object:

```
myBayesianAnalysis = uq_createAnalysis(BayesOpts);
```

Note: With no explicitly-specified discrepancy model, UQLAB assumes by default an unknown additive Gaussian discrepancy term, with a single unknown residual parameter $\boldsymbol{x}_\varepsilon \equiv \sigma^2$. The prior distribution of σ^2 is a weakly informative uniform distribution $\sigma^2 \sim \mathcal{U}(0, \mu^2_{\mathcal{Y}})$ with $\mu_{\mathcal{Y}}$ equal to the empirical mean of the provided data $\mathcal{Y}$.

Note: If not otherwise specified UQLAB uses the affine invariant ensemble sampler (Section 1.3.4) with $C = 100$ parallel chains and initial points drawn from the prior distribution and performs $T = 300$ iterations.

The sample set generated by MCMC algorithms often needs to be post-processed before it can be used as a true posterior sample (*e.g.*, to remove burn-in, Section 1.3.5.4). In UQLAB this is done with the uq_postProcessInversion function:

```
uq_postProcessInversion(myBayesianAnalysis);
```

By default this function removes the first half of the sample points generated by all chains as burn-in (see Section 1.3.5.4) and estimates the empirical parameter mean $\mathbb{E}[\boldsymbol{X}|\mathcal{Y}]$ along with the 5th and 95th percentile based on the post-processed sample. Additionally, 10,000 sample points from the prior and 1,000 sample points from the prior predictive and posterior predictive distributions are drawn (see Section 1.2.5) and stored in the myBayesianAnalysis object. For all available post-processing options see Section 3.3.

Note: The uq_postProcessInversion function only operates on a copy of the generated posterior sample. The sample originally generated by the MCMC algorithm along with the associated forward model evaluations is always kept in myBayesianAnalysis.

Note: Sample points drawn from the prior predictive distribution through uq_postProcessInversion require additional model evaluations.

A brief report of the analysis can then be generated by:

```
uq_print(myBayesianAnalysis)
```

which produces:

```
%------------------------ Inversion output ------------------------%
Number of calibrated model parameters:          1
Number of non-calibrated model parameters:      4

Number of calibrated discrepancy parameters:    1

%------------------- Data and Discrepancy
%  Data-/Discrepancy group 1:
Number of independent observations:             5

Discrepancy:
Type:                                           Gaussian
Discrepancy family:                             Scalar
Discrepancy parameters known:                   No

Associated outputs:
Model 1:
Output dimensions:                              1

%------------------- Solver
Solution method:                                MCMC
```

```
Algorithm:                                    AIES
Duration (HH:MM:SS):                          00:00:24
Number of sample points:                      6.00e+04



%------------------- Posterior Marginals
---------------------------------------------------------------------
||Parameter||Mean    ||Std      ||(0.05-0.95) quant.   ||Type       ||
---------------------------------------------------------------------
||E         ||2.48e+04||3.05e+03||(2.22e+04 - 2.69e+04)||Model      ||
||Sigma2    ||1.23e-05||2.63e-05||(1.21e-07 - 1.51e-05)||Discrepancy||
---------------------------------------------------------------------

%------------------- Point estimate
---------------------------------------------
|| Parameter || Mean     || Parameter Type ||
---------------------------------------------
|| E         || 2.37e+04 || Model          ||
|| Sigma2    || 3.51e-06 || Discrepancy    ||
---------------------------------------------
```

The results can also be visualized by:

```
uq_display(myBayesianAnalysis)
```

which produces the images in Figure 4.

2.2.6 Advanced options: discrepancy model

The discrepancy model defines the connection between the supplied data and the forward model. The Bayesian module of UQLAB currently supports the option to specify a *Gaussian additive discrepancy* as defined in Eq. (1.12), or to directly define a custom likelihood function as detailed in Section 2.6.

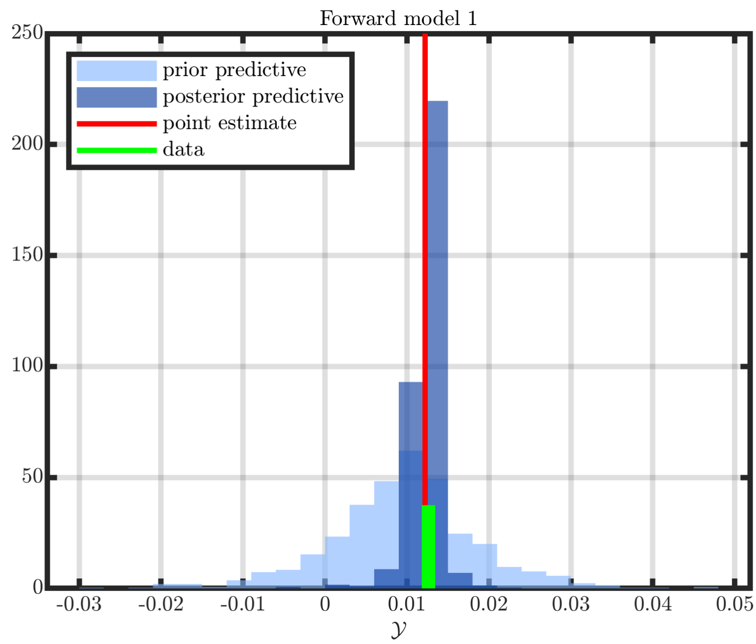
In real applications, it is often beneficial to specify more accurately the discrepancy model.

The options are to either specify a *known* residual variance σ^2 (*e.g.* tabulated measurement discrepancies from the instrument supplier) or an *unknown* $\boldsymbol{x}_\varepsilon = \sigma^2$ which can be inferred together with the model parameters $\boldsymbol{x}_\mathcal{M}$.

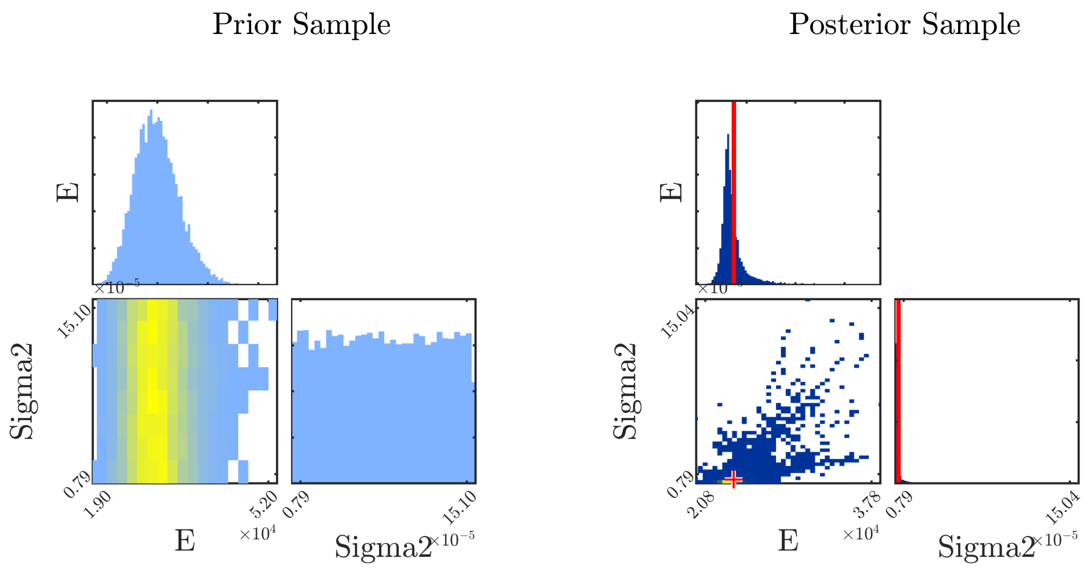
2.2.6.1 Known residual variance

If the variance is known a priori, as detailed in Section 1.2.2, it can be directly defined in an `DiscrepancyOpts` structure (assuming that $\sigma^2 = 10^{-6}$):

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = 1e-6;
```



(a) Prior and posterior predictive distribution, data and mean point estimate



(b) Scatterplot of prior and posterior sample points

Figure 4: Histograms of the prior and posterior predictive distributions with the empirical mean $\mathbb{E}\left[\boldsymbol{X} \,|\, \mathcal{Y}\right]$ estimated from the MCMC sample and scatterplots of the prior and posterior sample.

Which is then passed to BayesOpts by:

```
BayesOpts.Discrepancy = DiscrepancyOpts;
```

2.2.6.2 Unknown residual variance

If σ^2 is not known a priori, as detailed in Section 1.2.3, the Bayesian framework can infer the distribution of the discrepancy parameter $\boldsymbol{x}_\varepsilon \equiv \sigma^2$. This requires the initial specification of a prior distribution of the discrepancy parameter $\pi(\sigma^2)$ (see Eq. (1.2)).

The prior distribution of the parameter $\pi(\sigma^2)$ can be defined as a UQLAB INPUT object and then assigned to the DiscrepancyOpts structure's Prior field:

```
DiscrepancyPriorOpts.Name = 'Prior of discrepancy parameter';
DiscrepancyPriorOpts.Marginals.Name = 'Sigma2';
DiscrepancyPriorOpts.Marginals.Type = 'Uniform';
DiscrepancyPriorOpts.Marginals.Parameters = [0, mean(V_mid)^2];
myDiscrepancyPrior = uq_createInput(DiscrepancyPriorOpts);

DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Prior = myDiscrepancyPrior;
```

Note: Here a uniform prior on σ^2 with bounds $\mathcal{U}(0, \mu^2_{V_{\mathrm{mid}}})$ is implemented. This is the default (see above Section 2.2.5) when no discrepancy options are provided.

Note: As the prior $\pi(\sigma^2)$ is defined for the variance σ^2, only distributions with positive support can be used here.

2.3 Multiple model outputs

Models with multiple outputs are often encountered in real calibration problems. These multiple outputs can be different measurable quantities (*e.g.*, temperature and displacement), quantities at different locations (*e.g.*, deformations at different physical points), or at different times (*e.g.*, time series).

To show how such problems can be treated in UQLAB, the reference problem from Section 2.1 is slightly extended. It is assumed that additionally to measurements of the deflection at the beam mid-span at $L/2$, measurements are also available at $L/4$ as shown in Figure 5.

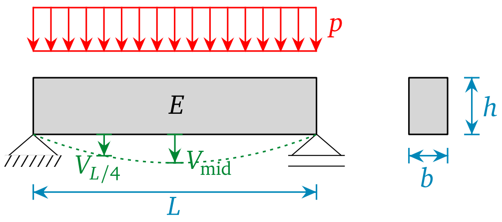


Figure 5: Simple beam bending test.

The $N = 5$ measurements of the deflections V_{mid} and $V_{L/4}$ are given in Table 3.

Table 3: Beam deflection: measured deflections.

Experiment	1	2	3	4	5
$V_{L/4}$ (mm)	8.98	8.66	8.85	9.19	8.64
V_{mid} (mm)	12.84	13.12	12.13	12.19	12.67

Similarly to (2.1), the quarter deflection at $L/4$ can be computed analytically by the standard beam theory:

$$V_{L/4} = \frac{57}{512} \frac{pL^4}{Ebh^3}. \tag{2.3}$$

Again due to a lack of information on the discrepancy, two independent unknown experimental discrepancies ε_1 and ε_2 are assumed for the measured displacements V_{mid} and $V_{L/4}$, respectively. As a weakly informative prior on the positive discrepancy variances $\boldsymbol{x}_\varepsilon = (\sigma_1^2, \sigma_2^2)$, two independent uniform distributions $\pi(\sigma_i^2) = \mathcal{U}(0, \mu_{\mathcal{Y}_i}^2)$ are chosen with $\mu_{\mathcal{Y}_i}$ equal to the mean of the observations V_{mid} and $V_{L/4}$ respectively (see Table 3).

2.3.1 Create a forward model

The equations for the beam mid-span ($L/2$) and quarter-span ($L/4$) deflection, given in Eq. (2.1) and Eq. (2.3), are implemented as a MATLAB `m-file` in:
`Examples/SimpleTestFunctions/uq_SimplySupportedBeamTwo.m`
This function returns the two beam deflections for a single model parameter realization $\boldsymbol{x}_{\mathcal{M}}$ in a row vector of length $N_{out} = 2$.

This is added to UQLAB with the following commands:

```
ModelOpts.Name = 'Forward model';
ModelOpts.mFile = 'uq_SimplySupportedBeamTwo';
myForwardModel = uq_createModel(ModelOpts);
```

2.3.2 Provide measurements

The measurements are stored in an $N \times N_{out}$ matrix V_mid and assigned to the BayesOpts object:

```
V = [[ 8.98;  8.66;  8.85;  9.19;  8.64],...    % L/4  (m)
     [12.84; 13.12; 12.13; 12.19; 12.67]]/1000; % L/2  (m)

myData.Name = 'Beam quarter and midspan deflection'
myData.y = V
```

Note: By default, UQLAB assigns the i-th column of the data matrix to the i-th output of the forward model. For more advanced options, refer to Section 2.3.4.3.

Note: With no explicitly-specified discrepancy model, the Bayesian inversion module by default assumes unknown additive Gaussian discrepancies for all N_{out} outputs of the forward model, with the residual parameter vector $\boldsymbol{x}_\varepsilon = (\sigma_1^2, \dots, \sigma_{N_{out}}^2)$. The prior distribution is by default taken as independent uniform distributions $\pi(\boldsymbol{x}_\varepsilon) = \prod_{i=1}^{N_{out}} \pi(\sigma_i^2)$ where $\pi(\sigma_i^2) = \mathcal{U}(0, \mu_{\mathcal{Y}_i}^2)$ with $\mu_{\mathcal{Y}_i}$ equal to the empirical mean of measurements available for the i-th output dimension.

2.3.3 Perform the inverse analysis

The analysis can be run with:

```
myBayesianAnalysis = uq_createAnalysis(BayesOpts);
```

Again post-processing is recommended (see Section 3.3 for all available options):

```
uq_postProcessInversion(myBayesianAnalysis);
```

After the myBayesianAnalysis object is created, the results can be summarized with:

```
uq_print(myBayesianAnalysis)
```

which produces the report:

```
%----------------------- Inversion output -----------------------%
Number of calibrated model parameters:          1
Number of non-calibrated model parameters:      4

Number of calibrated discrepancy parameters:    2

%------------------- Data and Discrepancy
%  Data-/Discrepancy group 1:
Number of independent observations:             5
```

```
Discrepancy:
Type:                                               Gaussian
Discrepancy family:                                 Row
Discrepancy parameters known:                       No

Associated outputs:
Model 1:
Output dimensions:                                  1


%------------------- Solver
Solution method:                                    MCMC

Algorithm:                                          AIES
Duration (HH:MM:SS):                                00:00:29
Number of sample points:                            9.00e+04

%------------------- Posterior Marginals
---------------------------------------------------------------------
||Parameter||Mean    || Std    ||(0.05-0.95) quant.||Parameter Type||
---------------------------------------------------------------------
|| E       ||3.18e+04||2.52e+03||2.95e+04 - 3.40e+04||Model        ||
|| Sigma2  ||6.75e-06||1.40e-05||4.01e-08 - 8.57e-06||Discrepancy  ||
|| Sigma2  ||6.94e-05||3.68e-05||2.18e-05 - 1.39e-04||Discrepancy  ||
---------------------------------------------------------------------

%------------------- Point estimate
---------------------------------------------
|| Parameter || Mean     || Parameter Type ||
---------------------------------------------
|| E         || 3.23e+04 || Model          ||
|| Sigma2    || 1.74e-06 || Discrepancy    ||
|| Sigma2    || 6.73e-05 || Discrepancy    ||
---------------------------------------------
```

and visualized graphically with:

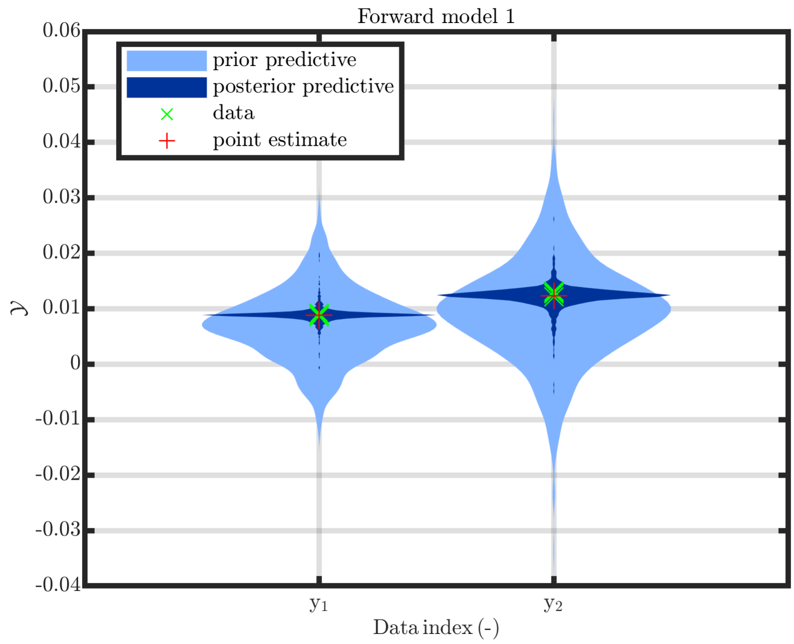
```
uq_display(myBayesianAnalysis)
```

which produces a set of plots similar to those in Figure 6.

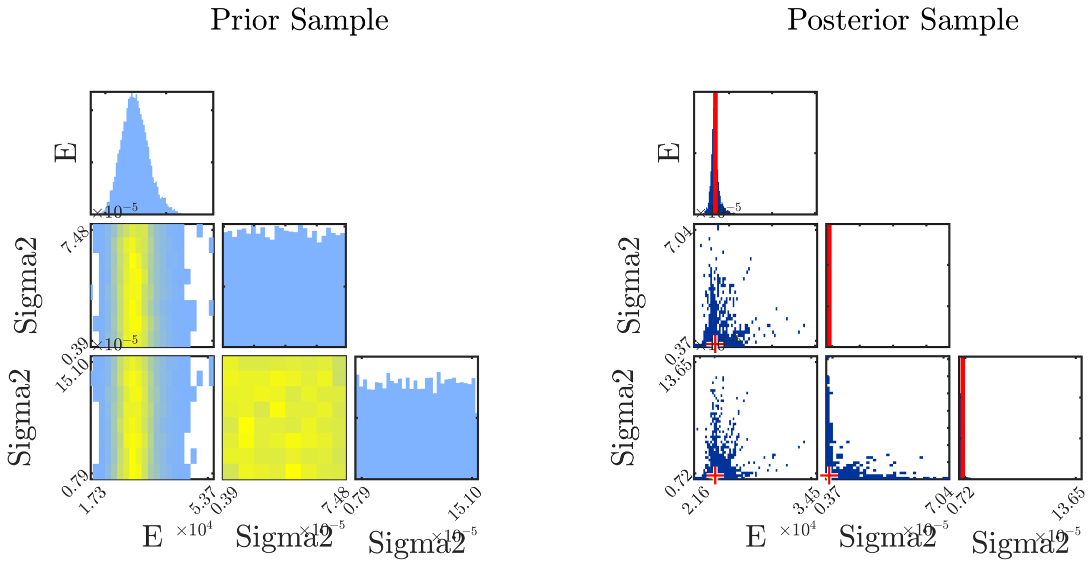
2.3.4 Advanced options: discrepancy model

In the case of models with multiple outputs, the full covariance matrix $\mathbf{\Sigma}$ of the residual discrepancy vector $\boldsymbol{\varepsilon} = (\varepsilon_i, \dots, \varepsilon_{N_{out}})$ has to be defined for the likelihood function. In this section, all options that are available in the UQLAB Bayesian inversion module to define this covariance matrix are presented:

1. Known residual variance: Section 2.3.4.1, see also Section 1.2.2



(a) Prior and posterior predictive distribution, data and mean point estimate



(b) Scatterplot of prior and posterior sample

Figure 6: Advanced discrepancy options: violin plots of the prior and posterior predictive distributions with the empirical mean $\mathbb{E}[\boldsymbol{X}|\mathcal{Y}]$ estimated from the MCMC sample and scatterplots of the prior and posterior sample.

2. Unknown residual variance: Section 2.3.4.2, see also Section 1.2.3

3. Data and discrepancy groups: Section 2.3.4.3

2.3.4.1 Known residual variance

In special cases, the residual variance may be known. This can happen when the forward model is supposed to perfectly represent the experimental setup, and when the discrepancy term reduces to measurement error. If the variance of this error is provided by the instrument supplier, it can be directly used to specify the covariance matrix $\mathbf{\Sigma}$ of the residual vector $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_{N_{out}})$. This known covariance matrix can be specified in three different ways in the Bayesian module of UQLAB.

Independent and identically distributed ε_i: In the case when all elements ε_i of the residual vector $\boldsymbol{\varepsilon}$ independently follow the same distribution $\mathcal{N}(0, \sigma^2)$ with a known variance σ^2 (*e.g.*, $\sigma^2 = 10^{-6}$), this can specified by a `DiscrepancyOpts` structure like

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = 1e-6; % single scalar
```

This structure is then passed to `BayesOpts` as follows:

```
BayesOpts.Discrepancy = DiscrepancyOpts;
```

Independent ε_i: If each element of the residual vector ε_i follows a Normal distribution $\mathcal{N}(0, \sigma_i^2)$ with a specific known residual variance σ_i^2, but independence can still be assumed, the `DiscrepancyOpts` structure can be defined as:

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = [1e-6 5e-7];% row vector of length N_out
```

where the length of the `.Parameters` vector is equal to N_{out}.

Dependent ε_i: In the general case where a known Gaussian distribution can be assumed for the discrepancy term, the covariance matrix can be passed to UQLAB as follows:

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = [ 1e-6  -1e-7;...
                              -1e-7   5e-7 ]; % N_out x N_out matrix
```

This covariance matrix introduces negative correlation between the first and second discrepancy parameter σ_1^2 and σ_2^2. Any *positive-definite matrix* can be used as a covariance matrix.

2.3.4.2 Unknown residual variance

In most practical applications, the parameters σ_i^2 are not known a priori. As detailed in Section 1.2.3, the Bayesian framework can infer the distribution of the discrepancy parameters gathered in $\boldsymbol{x}_\varepsilon$. This requires the initial specification of a prior distribution of the discrepancy parameters $\pi(\boldsymbol{x}_\varepsilon)$ (see Eq. (1.2)). Similarly to the previous section, there are different ways of specifying an unknown variance parameter σ_i^2, depending on the distribution of the residuals ε_i.

Note: In contrast to the known residual variance case (see Section 2.3.4.1), the definition of *dependent* unknown discrepancy terms ε_i (see Section 2.3.4.1) is not currently supported.

If an unknown residual variance is used, it becomes necessary to explicitly assign the INPUT object defining the prior of the model parameters $\pi(\boldsymbol{x}_\mathcal{M})$ to the BayesOpts structure. This is necessary to avoid confusions between the model parameter INPUT and error parameter INPUT objects. It can be assigned by:

```
BayesOpts.Prior = myPriorDist;
```

Independent and identically distributed ε_i: If the residuals of all observations are independent and identically distributed, a single *unknown* variance parameter σ^2 can be used in the distribution of all residuals ε_i. The prior distribution of the parameter $\pi(\sigma^2)$ can be defined as a UQLAB INPUT object and then assigned to the DiscrepancyOpts structure:

```
DiscrepancyPriorOpts.Name = 'Prior of discrepancy parameter';
DiscrepancyPriorOpts.Marginals.Name = 'Sigma2';
DiscrepancyPriorOpts.Marginals.Type = 'Uniform';
DiscrepancyPriorOpts.Marginals.Parameters = [0, 1e-4];
myDiscrepancyPrior = uq_createInput(DiscrepancyPriorOpts);

DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Prior = myDiscrepancyPrior;
```

Here a uniform prior $\pi(\sigma^2) = \mathcal{U}(0, 10^{-4})$ is chosen.

Note: As the prior $\pi(\sigma^2)$ is defined for the variance σ^2, only distributions with positive support can be used here.

Independent ε_i: If the residuals are independent but *not* identically distributed, the user can specify a dedicated discrepancy distribution for each σ_i^2. In the present case of $N_{out} = 2$, two independent prior distributions $\pi(\sigma_i^2)$ for the discrepancy parameters have to be specified. For the sake of illustration a lognormal prior is chosen for σ_1^2 and a uniform prior for σ_2^2 in the following code:

```
DiscrepancyPriorOpts.Name = 'Prior of discrepancy parameter';
DiscrepancyPriorOpts.Marginals(1).Name = 'Sigma2_1';
DiscrepancyPriorOpts.Marginals(1).Type = 'Lognormal';
DiscrepancyPriorOpts.Marginals(1).Moments = [1e-5, 5e-6];
DiscrepancyPriorOpts.Marginals(2).Name = 'Sigma2_2';
DiscrepancyPriorOpts.Marginals(2).Type = 'Uniform';
DiscrepancyPriorOpts.Marginals(2).Parameters = [0, 1e-4];
myDiscrepancyPrior = uq_createInput(DiscrepancyPriorOpts);

DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Prior = myDiscrepancyPrior;
```

2.3.4.3 Data and discrepancy groups

It often occurs in inverse problems that different types or number of data are collected for individual model outputs. In this case, it is often also necessary to define dedicated *discrepancy options* for individual outputs y_i.

In UQLAB this is achieved through so-called *data-* and *discrepancy groups*. The groups are defined by specifying the `DiscrepancyOpts` and `Data` as structure arrays, rather than simple structures. All options that were discussed in the previous sections can then be assigned to each structure in this array independently.

Consider the following case: for the first residual ε_1 the variance σ^2 is known to be $\sigma_1^2 = 10^{-6}$, while for the second residual ε_2 the variance σ_2^2 is unknown and assigned a uniform distribution $\sigma_2^2 \sim \mathcal{U}(0, 10^{-4})$. These discrepancy options can be passed to UQLAB by defining $N_{\text{gr}} = 2$ dedicated `DiscrepancyOpts` and `Data` structures in the following way:

```
% group 1
V_quart = [10.51;  9.60; 10.22;  8.16;  7.47]/1000;  % L/4 (m)
Data(1).y = V_quart;
Data(1).Name = 'Deflection measurements at L/4';
Data(1).MOMap = 1; % Model Output Map

DiscrepancyOpts(1).Type = 'Gaussian';
DiscrepancyOpts(1).Parameters = 1e-6;

% group 2
V_mid = [12.59; 11.23; 15.28; 12.45; 13.21]/1000; % L/2 (m)
Data(2).y = V_mid;
Data(2).Name = 'Deflection measurements at L/4';
Data(2).MOMap = 2; % Model Output Map

DiscrepancyPriorOpts.Name = 'Prior of sigma';
DiscrepancyPriorOpts.Marginals(1).Name = 'Sigma2_2';
DiscrepancyPriorOpts.Marginals(1).Type = 'Uniform';
```

```
DiscrepancyPriorOpts.Marginals(1).Parameters = [0, 1e-4];
DiscrepancyPrior = uq_createInput(DiscrepancyPriorOpts);

DiscrepancyOpts(2).Type = 'Gaussian';
DiscrepancyOpts(2).Prior = DiscrepancyPrior;
```

To link the defined group pairs with the model outputs, every `Data` structure requires a `MOMap` vector that maps the output indices $i \in \{1, \ldots, N_{out}\}$ to the respective group.

Through the use of data and discrepancy groups it also becomes possible to address problems where the number of measurements is not the same for each model output.

A simple example on how to use the `MOMap` array is given in Example 4 of the Bayesian inversion module (`uq_Example_Inversion_04_PredPrey`).

Note: In the above example the `MOMap` is not mandatory. By default it is indeed assigned consecutive indices based on the number of columns in each `Data.y` matrix and number of supplied groups.

`MOMap` can also be used to only select subsets of outputs of the model, even in the case of a single error group.

Note: The output IDs specified in the `MOMap` vector, are not unique. By giving the same index in the `MOMap` vectors of different data groups, it becomes possible to calibrate the same computational model using measurements gathered in different experiments.

2.4 Advanced options: solver

In the previous sections, the solver was not explicitly specified. By default UQLAB uses an MCMC algorithm with the affine invariant ensemble sampler (see Section 1.3.4) with $a = 2$.

Currently, the Bayesian module offers two different solution methods which are detailed in this section:

- MCMC: Section 2.4.1
- None: Section 2.4.2

2.4.1 MCMC

Currently, four MCMC samplers are shipped with the inversion module of UQLAB: Metropolis Hastings (MH), Adaptive-Metropolis (AM), Hamiltonian Monte Carlo (HMC) and affine invariant ensemble sampler (AIES). Their theoretical foundations are detailed in Section 1.3. An exhaustive list of all available options is given in Section 3.1.4.

To select an MCMC sampler, the following fields have to be specified:

```
Solver.Type = 'MCMC';
Solver.MCMC.Sampler = 'MH'; % AM, HMC, AIES
```

The number of iterations done by the sampler is given as a scalar in the `.Steps` field:

```
Solver.MCMC.Steps = 200;
```

Note that the cost per iteration depends on the specific sampler (see Section 1.3). All MCMC samplers require a set of initial seeds for the individual chains. These seeds are specified through an $M \times C$ matrix `Seed`, where C is the number of desired parallel chains:

```
Solver.MCMC.Seed =  Seed;
```

Alternatively it is also possible to just pass the number of chains C to the methods by passing a scalar value to the `NChains` field. For $C = 20$ this can be done by:

```
Solver.MCMC.NChains = 20;
```

The Bayesian module then automatically samples seeds from the prior distribution $\pi(\boldsymbol{x})$.

The field `Sampler` specifies the sampling algorithm. The options are `MH` (Metropolis-Hastings, Section 1.3.1), `AM` (adaptive Metropolis, Section 1.3.2), `HMC` (Hamiltonian Monte Carlo, Section 1.3.3), and `AIES` (affine invariant ensemble sampler, Section 1.3.4). Depending on the sampler, different options can be specified that are discussed in more detail next.

2.4.1.1 Metropolis-Hastings algorithm

The only parameter of the Metropolis-Hastings algorithm is the proposal distribution (see Section 1.3.1), which can be specified by (see also Table 12):

```
Solver.MCMC.Proposal = Proposal;
```

If the algorithm is to use a Gaussian proposal distribution centered at the previous sample (standard random walk algorithm), the `Proposal` field can be directly set to:

```
Proposal.PriorScale = 0.1;
```

This `PriorScale` is then used to define a covariance matrix $\boldsymbol{\Sigma}_p$ as a diagonal matrix proportional to the M prior marginal variances. Alternatively this covariance matrix can also be directly specified:

```
Proposal.Cov = Cov;
```

where `Cov` is an $M \times M$ positive-definite matrix.

Alternatively, if more advanced (*e.g.* non-Gaussian) proposal distributions are required, the `Proposal` field can also be specified as:

```
Proposal.Distribution = myProposal;
Proposal.Conditioning = 'Previous'; % Other valid option: 'Global'
```

where `myProposal` is a UQLAB INPUT object (UQLAB User Manual – the INPUT module). The field `Conditioning` can either be set to `'Global'` (default), to draw proposals from the distribution specified by `myProposal` independently of the previous sample point, or to `'Previous'`, which sets the mean value of the proposal distribution to the previous sample point in every step.

Note: A proposal distribution specified with the `'Previous'` option can be very slow compared to `'Global'` proposals. Unless there is a justified reason to use this option it is thus not recommended.

2.4.1.2 Adaptive Metropolis algorithm

The adaptive Metropolis algorithm takes the fields `Proposal`, `T0`, and `Epsilon` (see also Table 13). They can be, for example, set to:

```
Solver.MCMC.Proposal = Proposal;
Solver.MCMC.T0 = 1e2;
Solver.MCMC.Epsilon = 1e-4;
```

where the `Proposal` field can be specified as detailed in Section 2.4.1.1 and `T0` is the number of iterations t_0 during which the sampler uses the supplied proposal distribution `Proposal` before switching to the Gaussian distribution with the empirical covariance matrix as detailed in Eq. (1.32). The small number `Epsilon` specifies the ϵ added to the empirical correlation matrix to avoid singularity (see Section 1.3.2). If it is not specified, it is automatically set to $\epsilon = 10^{-6}$.

2.4.1.3 Hamiltonian Monte Carlo algorithm

The parameters of the Hamiltonian Monte Carlo algorithm are (see also Table 15):

```
Solver.MCMC.LeapfrogSteps = 40;
Solver.MCMC.LeapfrogSize = 0.1;
Solver.MCMC.Mass = 1;
```

with the number of leapfrog steps `.LeapfrogSteps` (N_τ), the leapfrog step size `.LeapfrogSize` ($\frac{\tau}{N_\tau}$), and the mass matrix `M` (see Section 1.3.3).

The mass matrix can be passed as a scalar value m, in which case the mass matrix takes the form $\boldsymbol{M} = m\boldsymbol{I}_M$ or directly as an $M \times M$ matrix.

2.4.1.4 Affine invariant ensemble algorithm

The only parameter of the affine invariant ensemble algorithm is the scalar `a` (see also Table 16). It can be set to any scalar parameter $a > 1$, for example for $a = 3$:

```
Solver.MCMC.a = 3;
```

This defines the parameter used for the stretch move distribution in Eq. (1.41). If this parameter is not set, a value of $a = 2$ is assumed in accordance with Goodman and Weare (2010); Allison and Dunkley (2013); Wicaksono (2017).

2.4.1.5 Visualization

As MCMC algorithms often take a long time to execute, it can be helpful to consult trace plots during runtime to assess convergence or prematurely terminate the algorithm in case of mistuning.

To enable live trace plots in UQLAB, the following options shall be added to the `.MCMC` field:

```
Solver.MCMC.Visualize.Parameters = Parameters;
Solver.MCMC.Visualize.Interval = Interval;
```

The `Parameters` variable determines which parameters x_i are plotted and can be passed as a vector of variable indices. The `Interval` variable fixes the interval at which the plot is refreshed and is given as a scalar integer. Low numbers lead to more frequent plot updates, but can significantly slow down the algorithm.

Note: If `Parameters` is a vector of indices, a separate figure is generated for each parameter x_i specified.

When the UQLAB ANALYSIS is run with an active visualization option, a plot like the one in Figure 7 is produced.

2.4.1.6 Post-processing

Following the analysis, the sample points generated by the MCMC algorithm are stored in the `Results` field of the `myBayesianAnalysis` structure:

```
myBayesianAnalysis.Results =

   struct with fields:
```

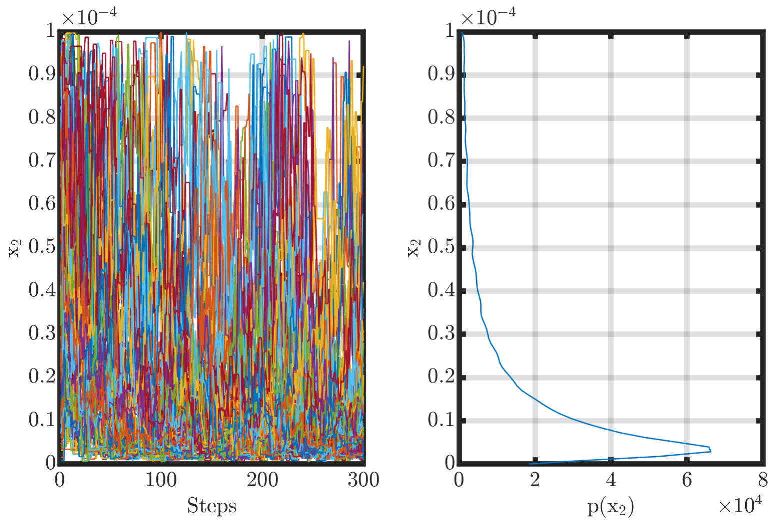


Figure 7: Trace plot and corresponding KDE during execution of the MCMC algorithm.

```
          Sample: [TxMxC double]
      Acceptance: [1xC double]
            Time: double
    ForwardModel: [1x1 struct]
   LogLikeliEval: [TxC double]
        PostProc: [1x1 struct]
```

The sample points are stored in the $T \times M \times C$ `Sample` array. The associated forward model and log likelihood evaluations are stored in the `ForwardModel` and `LogLikeliEval` structures respectively.

Sample points generated by MCMC algorithms typically require post-processing before they can be used as a true posterior sample. In the Bayesian module of UQLab this post-processing can be done with the `uq_postProcessInversion` function:

```
myBayesianAnalysis = uq_postProcessInversion(myBayesianAnalysis)
```

If this function is called, it performs a set of default post-processing procedures: the first half of all sample points generated by the MCMC chains are removed, the empirical parameter mean is estimated from these remaining sample points along with the 5th and 95th percentiles and samples are drawn from the prior and posterior predictive distributions.

These post-processing results are then stored in a `.PostProc` field in the `.Results` structure. For an extensive list of the available post-processing options, see Section 3.3.

2.4.2 No solver: posterior point by point evaluation

Sometimes it is not required to solve the inverse problem, but only to evaluate the prior/-posterior PDFs, or the likelihood function for specific parameter points $\boldsymbol{x}_0$ (*e.g.*, maximum a posteriori estimation). In this case, the solver type has to be set to `'None'`:

```
Solver.Type = 'None';
```

After the analysis creation with `uq_createAnalysis()`, the analysis object contains the following:

```
myBayesianAnalysis =

uq_analysis with properties:

              Options: [1x1 struct]
              Results: 1
             Internal: [1x1 struct]
                 Name: 'Bayesian multiple models'
                 Type: 'uq_inversion'
                 Data: [struct array]
      UnnormPosterior: [function_handle]
                Prior: [function_handle]
             LogPrior: [function_handle]
           Likelihood: [function_handle]
          Discrepancy: [struct array]
            PriorDist: [1x1 uq_input]
   UnnormLogPosterior: [function_handle]
        LogLikelihood: [function_handle]
         ForwardModel: [struct array]
```

If the solver type is set to `'None'`, no results are generated. Instead the analysis generates function handles to the prior distribution, the likelihood function and the *unnormalized* posterior distribution. The handle to the unnormalized posterior distribution can be used to find the maximum a posteriori (MAP) parameter value as shown in the supplied Example 7 of Bayesian inversion module (`uq_Example_Inversion_07_MAP`).

2.5 Advanced feature: multiple forward models

When data from multiple data sources are available (e.g. stresses and temperatures), different computational models may be needed. In the Bayesian module of UQLab it is possible to perform inversion on multiple models with different output and discrepancy options that depend on the same set, or subsets, of the parameters $\boldsymbol{x}_{\mathcal{M}}$. This can be achieved by specifying structure arrays for the `ForwardModel` field.

A simple example on how to use the multiple forward model feature is given in Example 5 of the Bayesian inversion module (`uq_Example_Inversion_05_MultipleModels`).

To show how such a problem is set up in UQLAB, it is assumed now that additional longitudinal tensile tests are carried out on a beam as shown in Figure 8.

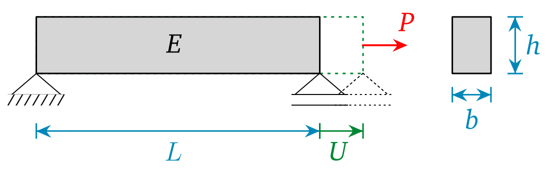


Figure 8: Tensile test.

The nominal value of the load P is $0.05\ kN$. In total $N = 3$ tensile tests are carried out, resulting in the deformations given in Table 4.

Table 4: Beam elongation: measured deformations.

Experiment	1	2	3
U (mm)	0.485	0.466	0.486

The elongation U of the specimen under a load P can be computed with (Sudret, 2018):

$$U = \frac{PL}{Ebh}. \tag{2.4}$$

The measurements in this case were carried out with a measuring device that has a known measurement error distribution of $\varepsilon \sim \mathcal{N}(0, 10^{-6})$.

2.5.1 Specify a prior distribution

The model prior object from Section 2.2.2 can be extended to contain the point load P:

```
PriorOpts.Marginals(6).Name = 'P';
PriorOpts.Marginals(6).Type = 'Constant';
PriorOpts.Marginals(6).Parameters = 0.05; % (kN)

myPriorDist = uq_createInput(PriorOpts);
```

2.5.2 Create a forward model

Each forward model has to be set up as a dedicated UQLAB MODEL. To do this Eq. (2.4) is translated to a string UQLAB MODEL and assigned together with the `uq_SimplySupportedBeam` model:

```
ModelOpts1.Name = 'Beam bending deflection';
ModelOpts1.mFile = 'uq_SimplySupportedBeam';
ForwardModels(1).Model = uq_createModel(ModelOpts1);
ForwardModels(1).PMap = [1 2 3 4 5];
```

```
ModelOpts2.Name = 'Beam elongation';
ModelOpts2.mString = 'X(:,5).*X(:,3)./(X(:,1).*X(:,2).*X(:,4))';
ForwardModels(2).Model = uq_createModel(ModelOpts2);
ForwardModels(2).PMap = [1 2 3 4 6];
```

The *parameter map* `PMap` defines which parameters from the model parameter vector $\boldsymbol{x}_{\mathcal{M}}$ are used for the respective model. The tensile test in `ForwardModels(2)` takes the 6-th parameter from the model prior distribution as a 5-th input (`'X(:,5)'` in the equation string).

Note: By default the `PMap` vector is set to address each parameter in the model parameter vector $\boldsymbol{x}_{\mathcal{M}}$. However, in most realistic usage scenarios it should be updated to properly list the desired parameters.

2.5.3 Provide measurements

The results of the tensile test are stored in a vector `U` and are passed to a discrepancy group `Data`:

```
U = [0.485; 0.466; 0.486]/1000; % (m)

% Data group 1
myData(1).y = V_mid;
myData(1).Name = 'Beam mid-span deflection';
myData(1).MOMap = [ 1;... % Model ID
                    1];   % Output ID

% Data group 2
myData(2).y = U;
myData(2).Name = 'Beam elongation';
myData(2).MOMap = [ 2;... % Model ID
                    1];   % Output ID
```

Note: In the case of multiple models the `MOMap` is mandatory and has to be defined as a two-row matrix. Its first row contains the indices of the corresponding models and the second row lists the indices of the corresponding model outputs.

2.5.4 Define a discrepancy model

The discrepancy can then be specified separately for the first model $\mathcal{M}_1(\boldsymbol{x}_{\mathcal{M}})$, where the variance of the discrepancy term is inferred from data, and the second model $\mathcal{M}_2(\boldsymbol{x}_{\mathcal{M}})$, where the measurement error variance is known, as follows:

```
% Discrepancy group 1
```

```
DiscrepancyPriorOpts1.Name = 'Prior of sigma_1^2';
DiscrepancyPriorOpts1.Marginals(1).Name = 'Sigma2';
DiscrepancyPriorOpts1.Marginals(1).Type = 'Uniform';
DiscrepancyPriorOpts1.Marginals(1).Parameters = [0, 1e-4];
myDiscrepancyPrior1 = uq_createInput(DiscrepancyPriorOpts1);

DiscrepancyOpts(1).Type = 'Gaussian';
DiscrepancyOpts(1).Parameters = myDiscrepancyPrior1;

% Discrepancy group 2
DiscrepancyOpts(2).Type = 'Gaussian';
DiscrepancyOpts(2).Parameters = 1e-6; % known discrepancy variance
```

2.5.5 Perform the inverse analysis

As in the previous case, all the options are then gathered in a single structure that contains all the information to perform the Bayesian inversion:

```
BayesOpts.Type = 'Inversion';
BayesOpts.Name = 'Bayesian multiple models';
BayesOpts.Prior = PriorDist;
BayesOpts.ForwardModel = ForwardModels;
BayesOpts.Data = myData;
BayesOpts.Discrepancy = DiscrepancyOpts;
```

In this case, `PriorDist` has to be explicitly assigned to the `BayesOpt` structure to avoid confusion with the INPUT object in the discrepancy model. It is also necessary to explicitly assign `myForwardModel` as there are multiple forward models now.

The solution of the inverse problem can then be started by running the analysis:

```
myBayesianAnalysis = uq_createAnalysis(BayesOpts);
```

Run post-processing with the default options:

```
uq_postProcessInversion(myBayesianAnalysis);
```

The results of this analysis can be assessed with a brief report generated by:

```
uq_print(myBayesianAnalysis)
```

which returns:

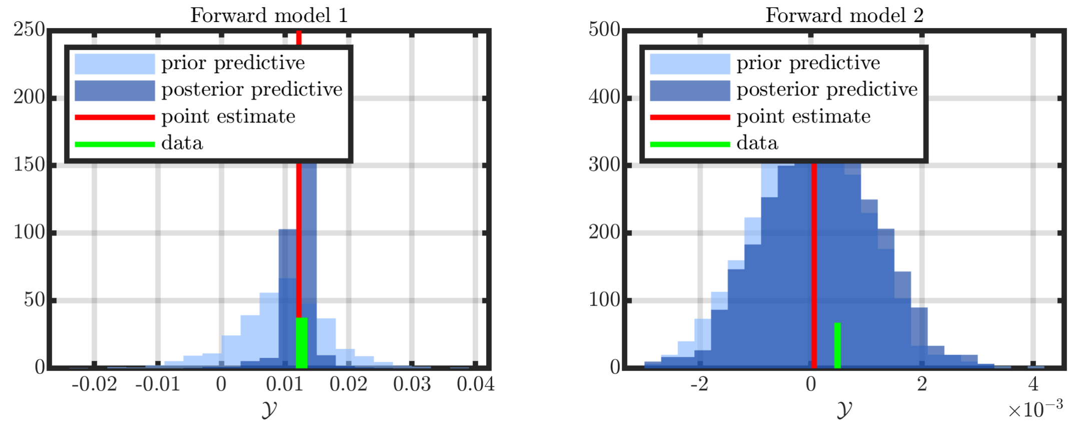
```
%----------------------- Inversion output -----------------------%
Number of calibrated model parameters:          1
Number of non-calibrated model parameters:      5

Number of calibrated discrepancy parameters:    1

%------------------- Data and Discrepancy
%  Data-/Discrepancy group 1:
```

```
Number of independent observations:          5

Discrepancy:
Type:                                        Gaussian
Discrepancy family:                          Scalar
Discrepancy parameters known:                No

Associated outputs:
Model 1:
Output dimensions:                           1

%  Data-/Discrepancy group 2:
Number of independent observations:          3

Discrepancy:
Type:                                        Gaussian
Discrepancy family:                          Scalar
Discrepancy parameters known:                Yes

Associated outputs:
Model 2:
Output dimensions:                           1

%--------------------  Solver
Solution method:                             MCMC

Algorithm:                                   AIES
Duration (HH:MM:SS):                         00:00:37
Number of sample points:                     6.00e+04

%--------------------  Posterior Marginals
-----------------------------------------------------------------------
||Parameter||Mean    ||Std     ||(0.05-0.95) quant. ||Parameter Type|
-----------------------------------------------------------------------
|| E       ||2.46e+04||2.84e+03||2.22e+04 - 2.72e+04|| Model        |
|| Sigma2  ||9.21e-06||1.81e-05||1.21e-07 - 1.54e-05|| Discrepancy  |
-----------------------------------------------------------------------

%--------------------  Point estimate
---------------------------------------------
|| Parameter || Mean     || Parameter Type ||
---------------------------------------------
|| E         || 2.37e+04 || Model          ||
|| Sigma2    || 3.44e-06 || Discrepancy    ||
---------------------------------------------
```

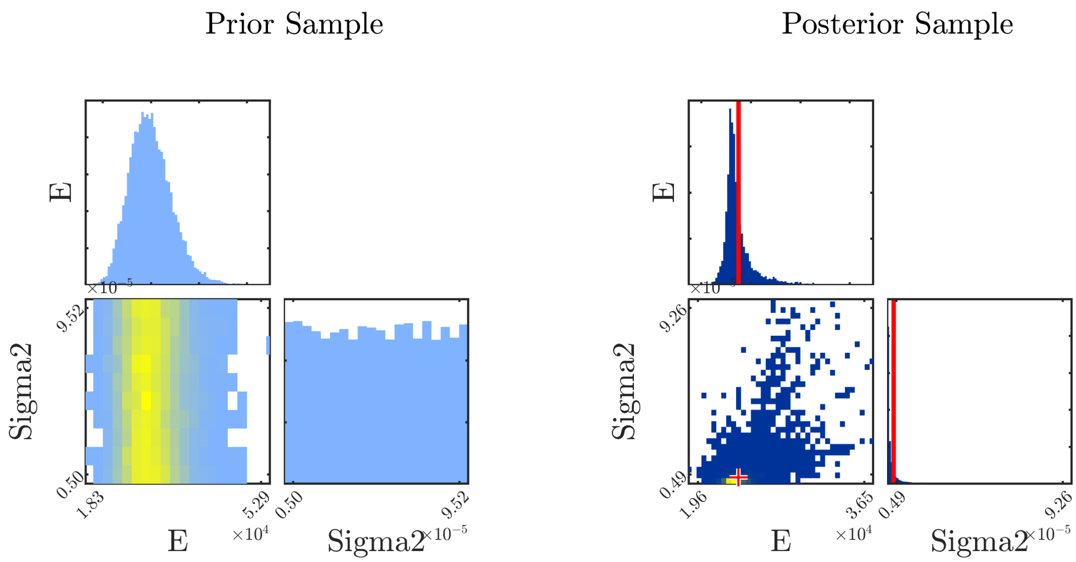
A visualization of the analysis can be produced by:

```
uq_display(myBayesianAnalysis)
```

which produces the images shown in Figure 9.



(a) Prior and posterior predictive distribution, data and mean point estimate for both forward models



(b) Scatterplot of prior and posterior sample

Figure 9: Multiple forward models: Histogram of the prior and posterior predictive distributions with the empirical mean $\mathbb{E}[\boldsymbol{X} \mid \mathcal{Y}]$ estimated from the MCMC sample and scatterplots of the prior and posterior sample.

2.6 Advanced options: user-defined likelihood function

Due to the wide class of possible discrepancy models that are employed in the Bayesian inversion practice, it is virtually impossible to provide an exhaustive set. Therefore, UQLab offers the possibility to directly specify a user-defined likelihood function, which entirely bypasses the built-in discrepancy model construction. This means that the user directly specifies a function that serves as the likelihood function (see Eq. (1.4)).

When a user-defined likelihood function is supplied, the `ForwadModel` and the `DiscrepancyOpts` fields are ignored by the Bayesian inversion module, as they are not required anymore. The user only needs to specify a prior distribution and a handle to the logarithm of a likelihood function as follows:

```
myLogLikelihood = @(params,y) customLogLikelihood(params,y);
```

The function handle above takes two inputs: the parameters `params` ($\boldsymbol{x}$) and the data `y` ($\mathcal{Y}$) and returns the log-likelihood function $\log \mathcal{L}(\boldsymbol{x}; \mathcal{Y})$ at these points.

During the execution of the analysis, the `params` are passed as $C \times M$ matrices, where C is the number of MCMC chains and M is the number of parameters in $\boldsymbol{x}$. If $C > 1$, *i.e.*, multiple parallel chains are run, the `customLogLikelihood` function needs to return a column vector of C log likelihood evaluations.

The `y` are passed as the `myData.y` field defined in Section 2.2.4. Generally,any data type can be assigned to the `y` field, as long as the supplied `customLogLikelihood` can process it.

To finalize the problem setup, these options are then assigned to a `BayesOpts` structure by:

```
BayesOpts.Type = 'Inversion';
BayesOpts.Name = 'User-defined likelihood inversion';
BayesOpts.Prior = myPriorDist;
BayesOpts.Data = myData;
BayesOpts.LogLikelihood = myLogLikelihood;
```

An example on how to use user-defined likelihood functions to define more advanced discrepancy models is given in Example 6 of the Bayesian inversion module (`uq_Example_Inversion_06_UserDefinedLikelihood`).

Chapter 3

Reference list

How to read the reference list

Structures play an important role throughout the UQLAB syntax. They offer a natural way to group configuration options and output quantities semantically. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested structures to fine-tune inputs/outputs. Throughout this reference guide, we adopt a table-based description of the configuration structures.

The simplest case is given when a field of the structure is a simple value/array of values:

Table X: `Input`			
●	`.Name`	String	A description of the field is put here

which corresponds to the following syntax:

```
Input.Name = 'My Input';
```

The columns correspond to name, data type and a brief description of each field. At the beginning of each row a symbol is given to inform as to whether the corresponding field is mandatory, optional, mutually exclusive, etc. The comprehensive list of symbols is given in the following table:

●	Mandatory
□	Optional
⊕	Mandatory, mutually exclusive (only one of the fields can be set)

⊞	Optional, mutually exclusive (one of them can be set, if at least one of the group is set, otherwise none is necessary)

When one of the fields of a structure is a nested structure, we provide a link to a table that describes the available options, as in the case of the `Options` field in the following example:

Table X: `Input`			
●	`.Name`	String	Description
□	`.Options`	Table Y	Description of the `Options` structure

Table Y: `Input.Options`			
●	`.Field1`	String	Description of `Field1`
□	`.Field2`	Double	Description of `Field2`

In some cases an option value gives the possibility to define further options related to that value. The general syntax would be

```
Input.Option1 = 'VALUE1' ;
Input.VALUE1.Val1Opt1 = ...;
Input.VALUE1.Val1Opt2 = ...;
```

This is illustrated as follows:

Table X: `Input`			
●	`.Option1`	String	Short description
		`'VALUE1'`	Description of `'VALUE1'`
		`'VALUE2'`	Description of `'VALUE2'`
⊞	`.VALUE1`	Table Y	Options for `'VALUE1'`
⊞	`.VALUE2`	Table Z	Options for `'VALUE2'`

Table Y: `Input.VALUE1`			
□	`.Val1Opt1`	String	Description
□	`.Val1Opt2`	Double	Description

Table Z: `Input.VALUE2`			
□	`.Val2Opt1`	String	Description
□	`.Val2Opt2`	Double	Description

Note: In the sequel, `double/doubles` mean a real number represented in double precision (resp. a set of such real numbers).

3.1 Create a Bayesian inverse analysis

Syntax

```
myBayesianAnalysis = uq_createAnalysis(BayesOpts);
```

Input

The structure `BayesOpts` contains the information for a Bayesian inverse analysis. N_{mod} is the number of computational forward models and N_{gr} is the number of data- and discrepancy groups.

Table 5: `BayesOpts`

●	`.Type`	`'Inversion'`	Inverse modeling
●	`.Data`	Table 6	The data used for inversion. See also Section 2.2.4, Section 2.3.4.3 and Section 2.5.3.
⊕	`.LogLikelihood`	Function handle	Function handle to the user specified log-likelihood function. Cannot be set together with `.ForwardModel` or `.Discrepancy`. See also Section 2.6.
□	`.Prior`	UQLAB INPUT	Prior distribution of the parameters. • If not specified and no INPUT object is used in the `.Discrepancy` or the `.Solver` field, the currently selected INPUT object is used • If `LogLikelihood` field not specified, prior distribution of model parameters $\pi(\boldsymbol{x}_{\mathcal{M}})$ • If `LogLikelihood` field specified, full prior distribution $\pi(\boldsymbol{x})$ See also UQLAB User Manual – the INPUT module .
⊞	`.ForwardModel`	UQLAB MODEL or Table 7	The forward model used for inversion. • If not specified, the currently selected MODEL object will be used Cannot be set together with `.LogLikelihood` See also UQLAB User Manual – the MODEL module and Section 2.5.2.

⊞	`.Discrepancy`	Table 8	The discrepancy model. Cannot be set together with `.LogLikelihood` See also Section 2.2.6, Section 2.3.4 and Section 2.5.4.
☐	`.Solver`	Table 9	Solver used for the inverse analysis. See also Section 2.4.
☐	`.Display`	String default: `'standard'`	Level of information displayed by the methods.
		`'quiet'`	Minimum display level, displays nothing or very few information.
		`'standard'`	Default display level, shows the most important information.
		`'verbose'`	Maximum display level, shows all the information on runtime, like updates on iterations, etc.
☐	`.Name`	String	Name of the module. If not set by the user, a unique string is automatically assigned to it.

3.1.1 Data structure

The `Data` field contains a $1 \times N_{\mathrm{gr}}$ structure array. The i-th structure in this array defines the i-th data group used in solving the inverse problem. $N_{out,i}$ is the number of model outputs related to the current data group (Section 2.3.4.3):

Table 6: `BayesOpts.Data`

●	`.y`	$N \times N_{out,i}$ Double or Arbitrary	The observations $\mathcal{Y}$ in the i-th data group. Only if a user-defined `.LogLikelihood` is given, `y` can have an arbitrary type. See also Section 2.6.
☐	`.MOMap`	Integers	Model output map relating the i-th data group to specific model outputs. See also Section 2.3.4.3.
		$1 \times N_{out,i}$ Integer default: consecutive numbering $1 : N_{out}$	Vector of model output indices related to the i-th data group. If $N_{\mathrm{mod}} > 1$ the definition below has to be used.
		$2 \times N_{out,i}$ Integer	First row contains the model index and second row the model output index.
☐	`.Name`	String	Name of the data group.

Note: The data and discrepancy groups are closely related. There always have to be as many data groups as discrepancy groups. See also Section 2.3.4.3.

3.1.2 Forward model structure

The `ForwardModel` field contains a $1 \times N_{\text{mod}}$ structure array. The i-th structure in this array contains the i-th computational forward model (Section 2.5.2):

Table 7: `BayesOpts.ForwardModel`

●	`.Model`	UQLab MODEL	i-th forward model used in the inverse analysis.
□	`.PMap`	$1 \times M$ default: `(1,...,M)`	Parameter map of the i-th forward model. Defines which parameters from `.Prior` are used for the evaluation of this forward model. See also Section 2.5.2.

3.1.3 Discrepancy model options

The `Discrepancy` field contains a $1 \times N_{\text{gr}}$ structure array. The i-th structure in this array defines the i-th discrepancy group used in solving the inverse problem. $N_{out,i}$ is the number of model outputs related to the current discrepancy group (Section 2.3.4):

Table 8: `BayesOpts.Discrepancy`

⊕	`.Parameters`	Doubles	Parameters for a known discrepancy model. See also Section 2.3.4.1.
		1×1 double	Independent discrepancies with same variance
		$1 \times N_{out,i}$ double	Independent discrepancies with individual variances
		$N_{out,i} \times N_{out,i}$ double	Full covariance matrix
⊕	`.Prior`	UQLab INPUT	Prior distribution for an unknown variance. See also Section 2.3.4.2.
□	`.Type`	String default: `'Gaussian'`	Type of the discrepancy distribution
		`'Gaussian'`	Only Gaussian discrepancies are currently supported

Note: The data and discrepancy groups are closely related. There always have to be as many data as discrepancy groups. See also Section 2.3.4.3.

3.1.4 Solver options

The solver used in analyzing the inverse problem can be specified with the following structure:

Table 9: `BayesOpts.Solver`

☐	`.Type`	String default: `'MCMC'`	Solution method of analysis
		`'MCMC'`	Markov chain Monte Carlo. See also Section 2.4.1.
		`'None'`	Only initialize and provide handles. See also Section 2.4.2.
☐	`.MCMC`	Table 10	Parameters of the MCMC algorithm

Table 10: `BayesOpts.Method.MCMC`

☐	`.Sampler`	String default: `'AIES'`	MCMC algorithm
		`'MH'`	Metropolis-Hastings algorithm. See also Table 12 and Section 2.4.1.1.
		`'AM'`	Adaptive Metropolis algorithm. See also Table 13 and Section 2.4.1.2.
		`'HMC'`	Hamiltonian Monte Carlo algorithm. See also Table 15 and Section 2.4.1.3.
		`'AIES'`	Affine invariant ensemble sampler algorithm. See also Table 16 and Section 2.4.1.4.
☐	`.Steps`	Integer scalar default: 300	Number of MCMC iterations T per chain
☐	`.Visualize`	Table 11	MCMC runtime visualization. See also Section 2.4.1.5
⊞	`.NChains`	Integer scalar default: 100	Number of parallel chains C. Initial points are randomly drawn from $\pi(\boldsymbol{x})$
⊞	`.Seed`	$1 \times M \times C$ Double or $M \times C$ Double	Initial points of the MCMC algorithm

Table 11: `BayesOpts.Solver.MCMC.Visualize`			
☐	`.Parameters`	Integer scalar	Which parameter should be visualized
☐	`.Interval`	Integer scalar	Plot interval

Depending on the chosen `.Sampler`, different additional options can be specified:

- Metropolis-Hastings algorithm - Table 12
- Adaptive Metropolis algorithm - Table 13
- Hamiltonian Monte Carlo algorithm - Table 15
- Affine invariant ensemble sampler algorithm - Table 16

Table 12: `BayesOpts.Method.MCMC (Metropolis-Hastings)`			
☐	`.Proposal`	Table 14	Proposal distribution $p(\boldsymbol{x}\|\boldsymbol{x}^{(t)})$

Table 13: `BayesOpts.Method.MCMC (Adaptive Metropolis)`			
☐	`.Proposal`	Table 14	Proposal distribution $p(\boldsymbol{x}\|\boldsymbol{x}^{(t)})$ until `T0`
☐	`.T0`	Integer scalar default: 300	Number of iterations t_0 until the adaptive proposal distribution is used.
☐	`.Epsilon`	Double scalar default: `1e-6`	Correction ϵ added to adaptive correlation diagonal to avoid singularity.

Table 14: `BayesOpts.Solver.MCMC.Proposal`			
⊞	`.PriorScale`	Doubles default: 0.1	Uses the scaled prior marginal variances as the covariance matrix $\boldsymbol{\Sigma}_p$ for Gaussian proposal centered at $\boldsymbol{x}^{(t)}$.
⊞	`.Cov`	$M \times M$ Double	Full covariance matrix $\boldsymbol{\Sigma}_p$
⊞	`.Distribution`	INPUT object	Custom proposal distribution. Requires also `.propCond` field
⊞	`.Conditioning`	String	Type of conditioning of proposal distribution
		`'Previous'`	Proposal distribution mean is set to $\boldsymbol{x}^{(t)}$

		'Global'	Samples are not conditioned on $\boldsymbol{x}^{(t)}$ but are directly drawn from supplied `.propDist`.

Table 15: `BayesOpts.Method.MCMC (Hamiltonian Monte Carlo)`

☐	`.LeapfrogSteps`	Integer scalar default: 10	Number of leapfrog integration steps N_τ
☐	`.LeapfrogSize`	Double scalar default: 0.01	Size of leapfrog integration steps $\frac{\tau}{N_\tau}$
☐	`.Mass`	Doubles default: 1	Mass matrix $\boldsymbol{M}$.
		Double scalar	Scale factor for identity matrix used as $\boldsymbol{M}$.
		$M \times M$ Double	Full $\boldsymbol{M}$.

Table 16: `BayesOpts.Method.MCMC (Affine invariant ensemble sampler)`

☐	`.a`	Double scalar > 1 default: 2	Parameter a for stretch move proposal.

3.2 Accessing analysis results

Syntax

```
myBayesianAnalysis = uq_createAnalysis(BayesOpts);
```

Output

After the analysis, the object `myBayesianAnalysis` contains the following important fields:

Table 17: `myBayesianAnalysis`

`.LogPrior`	Function handle	Log Prior probability density function $\log \pi(\boldsymbol{x})$.
`.Prior`	Function handle	Prior probability density function $\pi(\boldsymbol{x})$.
`.LogLikelihood`	Function handle	Log-likelihood function $\log \mathcal{L}(\boldsymbol{x}; \boldsymbol{y})$.
`.Likelihood`	Function handle	Likelihood function. If a custom likelihood was supplied (Section 2.6), this field is not available.

.Results	Table 18	Results of the specified .Solver. If the 'None' option was specified, this field contains 0.

Note: During the analysis all constant parameters from the input distribution are removed. The above handles expect parameters $\boldsymbol{x}$ without constants.

If the solver .Type was set to 'MCMC' the result field contains:

Table 18: myBayesianAnalysis.Results (MCMC)

.Sample	$T \times M \times C$ Double	Sample generated by the MCMC algorithm.
.ForwardModel	Structure array	The forward model evaluations associated with the sample stored in the .Sample field.
.Acceptance	$1 \times C$ Double	Acceptance rate of each chain.
.Time	Double scalar	Time required for simulation.
.PostProc	Structure array	The post processed MCMC results as generated by uq_postProcessInversion (see Section 3.3).

3.3 Post-processing results

Syntax

```
uq_postProcessInversion(myBayesianAnalysis);
uq_postProcessInversion(myBayesianAnalysis, Name, Value);
```

Description

uq_postProcessInversion(myBayesianAnalysis) post-processes the results stored in myBayesianAnalysis. By default the first half of all chains is discarded and the empirical posterior mean is estimated along with the 5th and 95th percentile from this reduced sample. Additionally, 10,000 sample points from the prior and 1,000 sample points from the prior predictive and posterior predictive distributions are drawn.

uq_postProcessInversion(myBayesianAnalysis, Name, Value) post-processes the results stored in myBayesianAnalysis using additional options specified by Name, Value pairs given in any order. These options are summarized in Table 19.

Note: If prior predictive samples are drawn, additional forward model evaluations are computed.

Table 19: `uq_postProcessInversion(..., Name, Value)`

'burnIn'	Integer or Double scalar default: 0.5	The burn-in for the MCMC sample (see Section 1.3.5.4).
	Double between 0 and 1	Specifies the fraction of sample points that are discarded as burn-in.
	Integer between 1 and T	Specifies the number of sample points that are discarded as burn-in.
'percentiles'	Double array default: $[0.05, 0.95]$	Computes the specified percentiles using the supplied sample.
'badChains'	Integer array	Removes the specified chains from the sample.
'prior'	Integer default: 10,000	Draws a specified number of sample points from the prior distribution.
'priorPredictive'	Integer default: 1,000	Draws a specified number of sample points from the prior predictive distribution, see also Eq. (1.24).
'posteriorPredictive'	Integer default: 1,000	Draws a specified number of sample points from the posterior predictive distribution, see also Eq. (1.26).
'pointEstimate'	String default: 'mean'	Computes a point estimate
	'Mean'	Computes the empirical mean from the supplied sample
	'MAP'	Estimates the maximum a posteriori point from the supplied sample
'gelmanRubin'	Logical	if `true` the multivariate potential scale reduction factor $\hat{R}^p$ is computed (see Section 1.3.5.3)

Examples:

```
uq_postProcessInversion(myBayesianAnalysis, 'badChains', [1 5],'pointEstimate'
,'MAP')
```

will remove the sample points generated by the first and fifth chain from the sample and return the maximum a posterior (MAP) point taken as the maximum unnormalized posterior evaluation from the available sample.

3.4 Printing/Visualizing results

UQLAB offers two commands to conveniently print reports containing contextually relevant information for a given object. If the post-processing was carried out with the uq_postProcessInversion function, the post-processed sample is used.

3.4.1 Printing the results: uq_print

Syntax

```
uq_print(myBayesianAnalysis)
uq_print(myBayesianAnalysis, Name, Value)
```

Description

uq_print(myBayesianAnalysis) print a report on the results of the Bayesian analysis in object myBayesianAnalysis.

3.4.2 Graphically display the results: uq_display

Syntax

```
uq_display(myBayesianAnalysis)
uq_display(myBayesianAnalysis, Name, Value)
```

Description

uq_display(myBayesianAnalysis) create a visualization of the Bayesian analysis in object myBayesianAnalysis. By default a scatterplot of the posterior sample and, if available, the prior and posterior predictive distributions are plotted.

uq_display(myBayesianAnalysis, Name, Value) create a visualization of the Bayesian analysis in object myBayesianAnalysis using only options specified by Name, Value pairs given in any order. These options are summarized in Table 20.

Table 20: uq_display(..., Name, Value)

'scatterplot'	String or Integer default: 'all'	Multi dimensional scatterplot
	'all'	Plots an M dimensional scatterplot of the generated sample

	Integer array	Plots the scatterplot with the specified parameters
'predDist'	Logical	Requires initial call to uq_postProcessInversion to draw prior and/or posterior predictive sample points (see Eq. (1.24) and Eq. (1.26)). •if $N_{out} = 1$ displays *histogram plots* based on the sample points generated by uq_postProcessInversion and scatterplots of $\mathcal{Y}$ •if $N_{out} > 1$ displays *violin plots* based on the sample points generated by uq_postProcessInversion and scatterplots of $\mathcal{Y}$
'chains'	String or Integer	Trace plot of MCMC chains.
	'all'	Displays trace plot of all M parameters
	Integer array	Displays trace plot of the parameters specified by the passed Integer array
'meanConvergence'	String or Integer	Convergence plot of the empirical mean averaged over all chains.
	'all'	Displays convergence of mean estimate of all M parameters.
	Integer array	Displays convergence of mean estimate of parameters specified by the array of Integers
'acceptance'	Logical	If true a plot of the acceptance ratio for all chains is displayed

Examples:

uq_display(myBayesianAnalysis,'scatterplot',[1 3]) will display a scatterplot of the first and third parameter x_1 and x_3.

References

Allison, R. and J. Dunkley (2013). Comparison of sampling techniques for Bayesian parameter estimation. *Monthly Notices of the Royal Astronomical Society 437*(4), 3918–3928. 13, 35

Beck, J. L. (2010). Bayesian system identification based on probability logic. *Structural Control & Health Monitoring 17*(7), 825–847. 1

Brooks, S., A. Gelman, G. L. Jones, and X.-L. Meng (Eds.) (2011). *Handbook of Markov Chain Monte Carlo*. Handbooks of Modern Statistical Methods. Chapman & Hall/CRC. 16

Brooks, S. P. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics 7*(4), 434–455. 15, 16

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). Hybrid Monte Carlo. *Physics Letters B 195*(2), 216–222. 11

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin (2014). *Bayesian Data Analysis* (3 ed.). Texts in Statistical Science. Boca Raton, Florida, USA: CRC Press. 1, 3

Gelman, A., G. O. Roberts, and W. R. Gilks (1996). Efficient Metropolis jumping rules. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (Eds.), *Bayesian Statistics 5, Proceedings of the 5th Valencia International Meeting, June 5-9, 1994*, pp. 599–607. Oxford University Press. 10

Gelman, A. and D. B. Rubin (1992). Inference from iterative simulation using multiple sequences. *Statistical Science 7*(4), 457–472. 15

Goodman, J. and J. Weare (2010). Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science 5*(1), 65–80. 12, 13, 35

Haario, H., E. Saksman, and J. Tamminen (2001). An adaptive Metropolis algorithm. *Bernoulli 7*(2), 223–242. 10, 11

Hadidi, R. and N. Gucunski (2008). Probabilistic approach to the solution of inverse problems in civil engineering. *Journal of Computing in Civil Engineering 22*(6), 338–347. 1

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*(1), 97–109. 9

Hu, K. T. and G. Orient (2016). The 2014 Sandia verification and validation challenge: problem statement. *Journal of Verification, Validation and Uncertainty Quantification 1*(1), 1–10. 4

Kaipio, J. and E. Somersalo (2005). *Statistical and Computational Inverse Problems*. Number 160 in Applied Mathematical Sciences. New York: Springer. 1

Liu, J. S. (2004). *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. New York: Springer. 8

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics 21*(6), 1087–1092. 9

Nagel, J. B. and B. Sudret (2016). Hamiltonian Monte Carlo and borrowing strength in hierarchical inverse problems. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 2*(3), 1–12. 11

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng (Eds.), *Handbook of Markov Chain Monte Carlo*, Handbooks of Modern Statistical Methods, Chapter 5, pp. 113–162. Boca Raton, Florida, USA: Chapman & Hall/CRC. 11, 12

Oberkampf, W. and C. Roy (2010). *Verification and Validation in Scientific Computing*. Cambridge University Press. 4

Oberkampf, W., T. Trucano, and C. Hirsch (2004). Verification, validation, and predictive capability in computational engineering and physics. *Applied Mechanics Reviews 57*(5), 345–384. 4

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods* (2 ed.). Springer Series in Statistics. New York: Springer. 8

Roberts, G. O., A. Gelman, and W. R. Gilks (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability 7*(1), 110–120. 13

Schoups, G. and J. A. Vrugt (2010). A formal likelihood function for parameter and predictive inference of hydrologic models with correlated, heteroscedastic, and non-Gaussian errors. *Water Resources Research 46*(10), W10531. 5

Sudret, B. (2018). *Einführung in die Baustatik*. ETH Zürich. 38

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Philadelphia, Pennsylvania, USA: Society for Industrial and Applied Mathematics (SIAM). 1

Wand, M. and M. C. Jones (1995). *Kernel smoothing*. Chapman and Hall, Boca Raton. 14

Wicaksono, D. C. (2017). *Bayesian uncertainty quantification of physical models in thermal-hydraulics system codes*. Ph. D. thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland. 13, 35

Yuen, K.-V. and S.-C. Kuok (2011). Bayesian methods for updating dynamic models. *Applied Mechanics Reviews 64*(1), 1–18. 1