ETH zürich

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

Risk, Safety & Uncertainty Quantification

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-V2.0-113, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland, 2022

BIBTEX entry

```
@TechReport{UQdoc_20_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 and Uncertainty Quantification, ETH Zurich,
Switzerland},
year = {2022},
note = {Report UQLab-V2.0-113}
}
```

Document Data Sheet

Document Ref.	UQLAB-V2.0-113
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:	01/02/2022

Doc. Version	Date	Comments
V2.0	01/10/2022	UQLab V2.0 release • Added spectral likelihood expansion (SLE) and stochastic spectral likelihood embedding (SSLE) solvers.
V1.4	01/02/2021	UQLab V1.4 release • Support simultaneous estimation of multiple point estimators. • Added section on data groups.
V1.3	19/09/2019	UQLab V1.3 release • Minor updates to plotting tools.
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 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 discussions on Markov chain Monte Carlo, spectral likelihood expansion, and stochastic spectral embedding 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, spectral likelihood expansion, stochastic spectral embedding

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). This is a very important extension to the Bayesia manual.

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 this data $\mathcal{X} \overset{\text{def}}{=} \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is made of independent realizations of an underlying random vector with an associated probability density function (PDF) $\pi(\cdot)$, 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 in the one-dimensional case), and the goal of inference is to estimate the *parameters* $\boldsymbol{\theta}$ of this PDF *given* the data or more formally:

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

where $\boldsymbol{\Theta}$ is the random vector associated with the parameters $\boldsymbol{\theta}$ and the vertical line $|$ is used to denote a conditional dependence of the quantity on the left on the quantity on the right.

When a sufficient amount of data exists, classical estimators can be used: for instance, if a Gaussian distribution $X|\boldsymbol{\theta} \sim \mathcal{N}(x|\mu, \sigma^2)$ is to be fitted to a sufficiently large data set $\mathcal{X}$, the

empirical mean and standard deviation of the sample may be used as estimators $\widehat{\boldsymbol{\theta}}$ of the parameters $\boldsymbol{\theta} \stackrel{\text{def}}{=} (\mu, \sigma^2)$. 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 $\widehat{\boldsymbol{\theta}}$ 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 parameters with the (possibly few) observed data points, using Bayes' theorem[1]. Before considering the data, in the Bayesian paradigm, the parameters of the parametric distribution $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ are considered as a random vector denoted by $\boldsymbol{\Theta}$ which is assumed to 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 parameters $\boldsymbol{\theta}$ *before* any measurement of $\boldsymbol{X}$ is carried out. From Bayes' theorem, the *posterior distribution* of the parameters, 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 $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, whose points are viewed as independent realizations of $\boldsymbol{X}|\boldsymbol{\theta} \sim \pi(\boldsymbol{x}|\boldsymbol{\theta})$. With these measurements the *likelihood function* $\mathcal{L}(\boldsymbol{\theta};\, \mathcal{X})$, a function of the parameters $\boldsymbol{\theta}$, can be defined:

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

This implicitly assumes independence between individual measurements in $\mathcal{X}$. Intuitively the likelihood function for a given $\boldsymbol{\theta}$ returns the *relative likelihood* of observing the data at hand, under the assumption that it follows the prescribed parametric distribution $\pi(\boldsymbol{x}|\boldsymbol{\theta})$.

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

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

The posterior distribution in Eq. (1.5) summarizes the information inferred about the param-

[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: $\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\ \mathbb{P}(B)}{\mathbb{P}(A)}$, where $\mathbb{P}(A|B)$ denotes the *conditional probability of A given B*.

eters 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 not trivial. Particular analytical solutions exist only for specific combinations of prior distributions on $\boldsymbol{\Theta}$ and likelihood functions, 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 $\boldsymbol{X}|\mathcal{X}$ given the information and data at hand. One possibility is to select a *point estimator* $\widehat{\boldsymbol{\theta}}$, *i.e.* a particular value from the posterior distribution of $\pi(\boldsymbol{\theta}|\mathcal{X})$. Then the point posterior distribution of $\boldsymbol{X}|\mathcal{X}$ simply reads:

$$\pi(\boldsymbol{x}|\mathcal{X}) \overset{\text{def}}{=} \pi(\boldsymbol{x}|\widehat{\boldsymbol{\theta}}). \tag{1.7}$$

Popular choices for $\widehat{\boldsymbol{\theta}}$ 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 estimation uncertainty in the parameters $\boldsymbol{\theta}$.

In contrast, it is also possible to incorporate the uncertainty in $\boldsymbol{\theta}$ into the prior and posterior assessment of $\boldsymbol{X}$ and $\boldsymbol{X}|\mathcal{X}$ respectively. This results in the so-called *predictive distributions*. The *prior predictive* distribution of $\boldsymbol{X}$ is obtained by "averaging" the parametric distribution $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ over the prior distribution $\pi(\boldsymbol{\theta})$:[2]

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

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

$$\pi(\boldsymbol{x}|\mathcal{X}) \overset{\text{def}}{=} \int_{\mathcal{D}_{\boldsymbol{\Theta}}} \pi(\boldsymbol{x}|\boldsymbol{\theta})\,\pi(\boldsymbol{\theta}|\mathcal{X})\,\mathrm{d}\boldsymbol{\theta}. \tag{1.9}$$

[2] It is noted here that for a single measurement $\mathcal{X} = \boldsymbol{x}_1$, the prior predictive distribution at $\mathcal{X}$ equals the normalization constant Z defined in Eq. (1.6).

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 $\boldsymbol{y} \in \mathbb{R}^{N_{\text{out}}}$ as a function of input parameters $\boldsymbol{x}$:

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

Such models are commonly established based on first principles in engineering sciences (*e.g.* mechanics, electromagnetism, fluid dynamics), but also natural sciences (*e.g.* geophysics, wave propagation). 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 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 $\boldsymbol{y}_i \in \mathbb{R}^{N_{\text{out}}}$ gathered in a data set $\mathcal{Y} \overset{\text{def}}{=} \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$. In the context of computational modelling 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 the 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 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 $\boldsymbol{\varepsilon} \in \mathbb{R}^{N_{\text{out}}}$ 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*[3] 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 parameters $\boldsymbol{\Theta}$ 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 $\boldsymbol{\Sigma}$. This distribution is called the *error* or *discrepancy model* $\pi(\boldsymbol{y}|\boldsymbol{x})$ given by:

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

If N *independent* measurements $\boldsymbol{y}_i$ are available and gathered in the data set $\mathcal{Y} =$

[3] 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.

$\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$, the likelihood function can thus be written as

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

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:

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

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 $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_{N_{\text{out}}}$ with unknown residual variances $\sigma^2 = \text{Var}[\varepsilon_i], i = 1, \ldots, N_{\text{out}}$. 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

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

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:

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

1.2.4 Multiple data groups

In practice, it occurs frequently that the measurements $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ stem from various measurement devices or experimental conditions with different discrepancy properties. In these cases, it is necessary to arrange the elements of $\mathcal{Y}$ in disjoint *data groups* and define different likelihood functions for each data group.

Denoting the g-th data group by $\mathcal{G}^{(g)} = \{\boldsymbol{y}_i\}_{i\in\mathbf{u}}$, where $\mathbf{u} \subseteq \{1, \ldots, N\}$, the full data set can be combined by

$$\mathcal{Y} = \bigcup_{g=1}^{N_{\text{gr}}} \mathcal{G}^{(g)}. \tag{1.20}$$

Each of the N_{gr} data groups contains measurements collected with the same instruments under similar measurement conditions. With this, it is clear that each data group requires a different likelihood function $\mathcal{L}^{(g)}$ describing the experimental conditions that led to measuring $\mathcal{G}^{(g)}$. Possible choices for $\mathcal{L}^{(g)}$ are presented in Section 1.2.2 and Section 1.2.3. Under the assumption of independence between the N_{gr} measurement conditions, the full likelihood function can then be written as

$$\mathcal{L}(\boldsymbol{x}_{\mathcal{M}},\, \boldsymbol{x}_{\varepsilon}; \mathcal{Y}) = \prod_{g=1}^{N_{\text{gr}}} \mathcal{L}^{(g)}(\boldsymbol{x}_{\mathcal{M}}, \boldsymbol{x}_{\varepsilon}^{(g)}; \mathcal{G}^{(g)}), \tag{1.21}$$

where $\boldsymbol{x}_{\varepsilon}^{(g)}$ are the parameters of the g-th discrepancy group.

1.2.5 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

$$\mathbb{E}\left[\boldsymbol{X}\,|\,\mathcal{Y}\right] = \int_{\mathcal{D}_{\boldsymbol{X}}} \boldsymbol{x}\, \pi(\boldsymbol{x}|\mathcal{Y})\, \mathrm{d}\boldsymbol{x}. \tag{1.22}$$

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

$$\mathrm{Cov}[\boldsymbol{X}\,|\,\mathcal{Y}] = \int_{\mathcal{D}_{\boldsymbol{X}}} (\boldsymbol{x} - \mathbb{E}\,[\boldsymbol{X}\,|\,\mathcal{Y}])(\boldsymbol{x} - \mathbb{E}\,[\boldsymbol{X}\,|\,\mathcal{Y}])^{\intercal}\,\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.23}$$

One may also be interested in the posterior marginals. The univariate posterior 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):

$$\pi_i(x_i|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}_{\sim i}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\sim i}, \tag{1.24}$$

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

More generally, it is also possible to define multivariate posterior marginals by integrating the posterior over all but the parameters of interest. To this end, we split the random vector $\boldsymbol{X}$ into two vectors $\boldsymbol{X}_{\mathbf{u}}$ with components $\{X_i\}_{i\in\mathbf{u}} \in \mathcal{D}_{\boldsymbol{X}_{\mathbf{u}}}$ and $\boldsymbol{X}_{\mathbf{v}}$ with components $\{X_i\}_{i\in\mathbf{v}} \in \mathcal{D}_{\boldsymbol{X}_{\mathbf{v}}}$, where $\mathbf{u}$ and $\mathbf{v}$ are two non-empty disjoint index sets such that $\mathbf{u}\cup\mathbf{v} = \{1,\ldots,M\}$. The multivariate posterior marginals then read:

$$\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}_{\mathbf{v}}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\mathbf{v}}. \tag{1.25}$$

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 $h(\boldsymbol{X}\,|\,\mathcal{Y})$ and is computed by the integral

$$\mathbb{E}\,[h(\boldsymbol{X}\,|\,\mathcal{Y})] = \int_{\mathcal{D}_{\boldsymbol{X}}} h(\boldsymbol{x})\,\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.26}$$

1.2.6 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. Using the previously defined discrepancy model, the *prior predictive* distribution from Eq. (1.8) can be written as

$$\pi(\boldsymbol{y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \pi(\boldsymbol{y}|\boldsymbol{x})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{1.27}$$

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

$$\pi(\boldsymbol{y}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \pi(\boldsymbol{y}|\boldsymbol{x})\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}. \tag{1.28}$$

Owing to the considered additive Gaussian discrepancy model in Eq. (1.13), a sample from those predictive distributions can be obtained by using samples from $\boldsymbol{X}$ and $\boldsymbol{X}|\mathcal{Y}$ respectively, propagating them through the model $\mathcal{M}$ and adding an independently sampled discrepancy term $\boldsymbol{\varepsilon}$.

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:

$$\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\,\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}) = \pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y})\,\mathcal{K}(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t+1)}). \tag{1.29}$$

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:

$$\pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}}} \pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\,\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})\,\mathrm{d}\boldsymbol{x}^{(t)}. \tag{1.30}$$

A Markov chain constructed this way can be used to approximate the expectation in Eq. (1.26) as the iteration average of the $T+1$ generated sample points $\boldsymbol{x}^{(t)}$

$$\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] \approx \frac{1}{T}\sum_{t=1}^{T} h(\boldsymbol{x}^{(t)}). \tag{1.31}$$

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 al-

gorithm 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:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})\, p(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(\star)})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\, p(\boldsymbol{x}^{(\star)}|\boldsymbol{x}^{(t)})}\right\}, \tag{1.32}$$

and rejected otherwise (*i.e.*, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$). With this procedure, the transition probability fulfills Eq. (1.30) 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. (1.31).

It is advantageous that the model evidence cancels out from the acceptance probability in Eq. (1.32). 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. (1.32) reduces to:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})}\right\}. \tag{1.33}$$

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. (1.32) or Eq. (1.33), 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. (1.32). 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:

$$\boldsymbol{C}(t+1) = \begin{cases} \boldsymbol{C}_0, & t+1 \leq t_0, \\ s_d(M)\tilde{\boldsymbol{C}}(t), & t+1 > t_0, \end{cases} \tag{1.34}$$

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:

$$\tilde{\boldsymbol{C}}(t) = \frac{1}{t-1}\left(\sum_{i=1}^{t}(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(t)})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}}^{(t)})^{\intercal}\right), \quad \text{where} \quad \bar{\boldsymbol{x}}^{(t)} = \frac{1}{t}\sum_{i=1}^{t}\boldsymbol{x}^{(i)}. \tag{1.35}$$

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:

$$p(\boldsymbol{x}|\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}^{(t)}, \boldsymbol{C}^{\star}(t)). \tag{1.36}$$

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:

$$\alpha\left(\boldsymbol{x}^{(\star)}, \boldsymbol{x}^{(t)}\right) = \min\left\{1, \frac{\pi(\boldsymbol{x}^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})}\right\}, \tag{1.37}$$

which is the acceptance probability already defined in Eq. (1.32) 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. (1.29). Nonetheless, it was shown in Haario et al. (2001) that the generated sample points can be used to approximate posterior properties by Eq. (1.31).

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 (2016a).

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 $K(\boldsymbol{p})$:

$$U(\boldsymbol{x}) = -\log\left(\pi(\boldsymbol{x})\mathcal{L}(\boldsymbol{x};\mathcal{Y})\right), \qquad K(\boldsymbol{p}) = \frac{\boldsymbol{p}^{\intercal}\boldsymbol{M}^{-1}\boldsymbol{p}}{2}, \tag{1.38}$$

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. (1.38) to define the Hamiltonian:

$$\mathcal{H}(\boldsymbol{x},\boldsymbol{p}) = U(\boldsymbol{x}) + K(\boldsymbol{p}). \tag{1.39}$$

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):

$$\dot{x_i} = \frac{\partial\mathcal{H}(\boldsymbol{x},\boldsymbol{p})}{\partial p_i}, \qquad \dot{p_i} = \frac{\partial\mathcal{H}(\boldsymbol{x},\boldsymbol{p})}{\partial x_i}, \quad \text{for} \quad i = 1,\ldots,M. \tag{1.40}$$

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:

$$\boldsymbol{p}(0) \sim \mathcal{N}(\boldsymbol{p}|\boldsymbol{0},\boldsymbol{I}_M). \tag{1.41}$$

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:

$$\alpha\left(\boldsymbol{x}(\tau), \boldsymbol{p}(\tau), \boldsymbol{x}(0), \boldsymbol{p}(0)\right) = \min\left\{1, \exp\left(\mathcal{H}(\boldsymbol{x}(0), \boldsymbol{p}(0)) - \mathcal{H}(\boldsymbol{x}(\tau), \boldsymbol{p}(\tau))\right)\right\}. \tag{1.42}$$

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:

$$\boldsymbol{x}_i^{(\star)} = \boldsymbol{x}_i^{(t)} + Z \cdot \left(\boldsymbol{x}_j^{(\tilde{t})} - \boldsymbol{x}_i^{(t)}\right), \tag{1.43}$$

where $\tilde{t} = t + 1$ if $j < i$ and $\tilde{t} = t$ otherwise, *i.e.*it denotes the latest state of the j-th walker.

Z is randomly drawn from the PDF

$$p(z|a) = \begin{cases} \frac{1}{\sqrt{z}(2\sqrt{a}-\frac{2}{\sqrt{a}})} & \text{if } z \in [1/a, a]\,, \\ 0 & \text{otherwise,} \end{cases} \tag{1.44}$$

The candidate $\boldsymbol{x}_i^{(\star)}$ is then accepted as the new location of the i-th walker with probability:

$$\alpha\left(\boldsymbol{x}_i^{(\star)}, \boldsymbol{x}_i^{(t)}, z\right) = \min\left\{1, z^{M-1}\frac{\pi(\boldsymbol{x}_i^{(\star)}|\mathcal{Y})}{\pi(\boldsymbol{x}_i^{(t)}|\mathcal{Y})}\right\}. \tag{1.45}$$

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. (1.31). 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

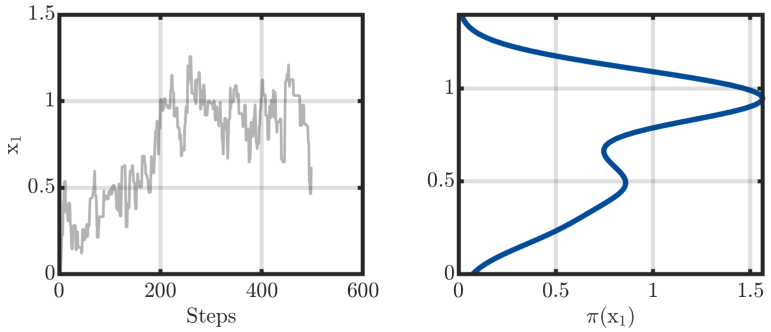
All MCMC algorithms produce chains of sample points that will eventually follow the posterior distribution. In practice, however, one is forced to make decisions about convergence based on a finite number of sample points. As MCMC algorithms lack a convergence criterion, numerous heuristics have been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

1.3.5.1 Acceptance rate

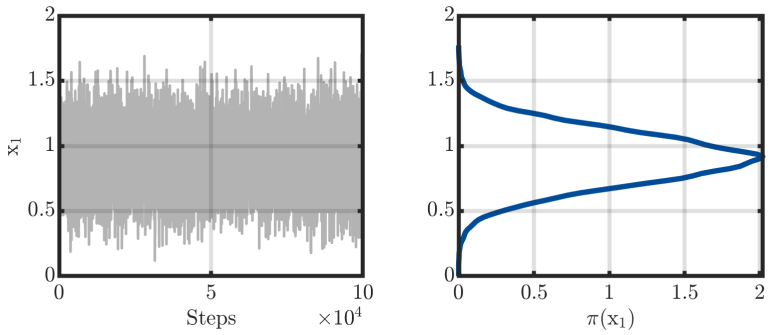
The acceptance rate r_a gives a quantitative indication of how many proposed sample points were accepted. It can be simply computed as the ratio between the number of accepted points and the total number of iterations $T + 1$.

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 the target distribution. Acceptance rates close to zero indicate instead that the proposed



(a) After 500 steps



(b) After 10^5 steps

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

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.

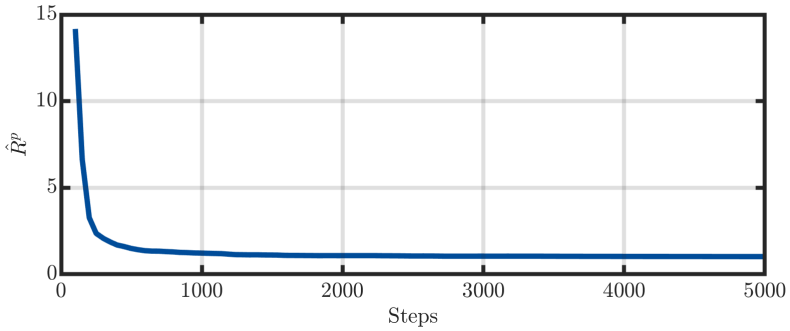


Figure 2: Convergence of the MPSRF $\hat{R}^p$.

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:

$$\boldsymbol{W}_i = \frac{1}{T}\sum_{t=0}^{T}\left(\boldsymbol{x}_i^{(t)} - \bar{\boldsymbol{x}}_i\right)\left(\boldsymbol{x}_i^{(t)} - \bar{\boldsymbol{x}}_i\right)^{\intercal}, \quad \bar{\boldsymbol{x}}_i = \frac{1}{T+1}\sum_{t=0}^{T}\boldsymbol{x}_i^{(t)}. \tag{1.46}$$

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

$$\boldsymbol{W} = \frac{1}{C}\sum_{i=1}^{C}\boldsymbol{W}_i. \tag{1.47}$$

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:

$$\boldsymbol{B} = \frac{1}{C-1}\sum_{i=1}^{C}\left(\bar{\boldsymbol{x}}_i - \bar{\bar{\boldsymbol{x}}}\right)\left(\bar{\boldsymbol{x}}_i - \bar{\bar{\boldsymbol{x}}}\right)^{\intercal}, \quad \text{where:} \tag{1.48}$$

$$\bar{\bar{\boldsymbol{x}}} = \frac{1}{C(T+1)}\sum_{i=1}^{C}\sum_{t=0}^{T}\boldsymbol{x}_i^{(t)} \tag{1.49}$$

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):

$$\hat{R}^p = \frac{T}{T+1} + \left(\frac{C+1}{C}\right)\lambda_1, \tag{1.50}$$

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*.

1.4 Spectral likelihood expansion

A different way to solving Bayesian inverse problem named *spectral likelihood expansion* (SLE) was proposed in Nagel and Sudret (2016b). This sampling-free approach uses spectral expansion techniques such as *polynomial chaos expansion* (PCE, UQLAB User Manual – Polynomial Chaos Expansions) to approximate the likelihood function at the core of Bayesian inverse problems.

Likelihood functions can be seen as scalar functions of the input random vector $\boldsymbol{X} \sim \pi(\boldsymbol{x})$ with finite output variance (Nagel and Sudret, 2016b). Assuming independent priors, *i.e.* $\pi(\boldsymbol{x}) = \prod_{i=1}^{M} \pi_i(x_i)$, their spectral expansion reads:

$$\mathcal{L}(\boldsymbol{X}) \approx \mathcal{L}_{\mathrm{SLE}}(\boldsymbol{X}) \overset{\text{def}}{=} \sum_{\boldsymbol{\alpha}\in\mathcal{A}} a_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}), \tag{1.51}$$

where $\Psi_{\boldsymbol{\alpha}}$ are basis functions (polynomials in the case of PCE) that are orthogonal w.r.t. the prior distribution $\pi(\boldsymbol{x})$ and $c_{\boldsymbol{\alpha}}$ are the corresponding coefficients. Following this approximation, it becomes possible to exploit the orthogonality of the spectral basis functions to post-process the expansion coefficients and extract the following important posterior quantities of interest:

Evidence The evidence emerges as the coefficient of the constant polynomial $a_{\mathbf{0}}$

$$Z = \int_{\mathcal{D}_{\boldsymbol{X}}} \mathcal{L}(\boldsymbol{x})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \approx \mathbb{E}\left[\mathcal{L}_{\mathrm{SLE}}(\boldsymbol{X})\right] = a_{\mathbf{0}}. \tag{1.52}$$

Posterior Upon computing the evidence Z, the posterior can be evaluated directly through

$$\pi(\boldsymbol{x}|\mathcal{Y}) \approx \frac{\mathcal{L}_{\mathrm{SLE}}(\boldsymbol{x})\pi(\boldsymbol{x})}{Z} = \frac{\pi(\boldsymbol{x})}{a_{\mathbf{0}}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}} a_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}). \tag{1.53}$$

Posterior marginals An approximation of the univariate posterior marginals defined in Eq. (1.24) can then also be derived analytically:

$$\pi_i(x_i|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}_{\sim i}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\sim i} \approx \frac{\pi_i(x_i)}{a_{\mathbf{0}}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\sim i=0}} a_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\alpha}}(x_i), \tag{1.54}$$

where $\mathcal{A}_{\sim i=0} = \{\boldsymbol{\alpha}\in\mathcal{A}\colon \alpha_j = 0 \Leftrightarrow j \neq i\}$.

Similarly, an expression for the multivariate posterior marginal defined in Eq. (1.25) can be derived. Denote by $\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}) \overset{\text{def}}{=} \prod_{i\in\mathbf{u}}\pi_i(x_i)$ and $\pi_{\mathbf{v}}(\boldsymbol{x}_{\mathbf{v}}) \overset{\text{def}}{=} \prod_{i\in\mathbf{v}}\pi_i(x_i)$ the prior marginal density functions of $\boldsymbol{X}_{\mathbf{u}}$ and $\boldsymbol{X}_{\mathbf{v}}$ respectively, the posterior marginal then reads:

$$\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}_{\mathbf{v}}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\mathbf{v}} \approx \frac{\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}})}{a_{\mathbf{0}}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}_{\mathbf{v}=0}} a_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}_{\mathbf{u}}), \tag{1.55}$$

where $\mathcal{A}_{\mathbf{v}=0} = \{\boldsymbol{\alpha}\in\mathcal{A}\colon \alpha_i = 0 \Leftrightarrow i\in\mathbf{v}\}$. The series in Eq. (1.54) is a subexpansion that contains non-constant polynomials only along the dimensions $i\in\mathbf{u}$.

Quantities of interest It is also possible to analytically compute posterior expectations of functions that admit a polynomial chaos expansion on the same basis, of the form $h(\boldsymbol{X}) \approx \sum_{\boldsymbol{\alpha}\in\mathcal{A}} b_{\boldsymbol{\alpha}}\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$. Eq. (1.26) then reduces to the spectral product:

$$\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] = \frac{1}{a_{\mathbf{0}}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}} a_{\boldsymbol{\alpha}}b_{\boldsymbol{\alpha}}. \tag{1.56}$$

This expression can be used to compute posterior moments like mean, variance or covariance.

The quality of these results depends only on the approximation error introduced in Eq. (1.51). The latter, in turn, depends mainly on the chosen PCE truncation strategy (Nagel and Sudret, 2016b; Lüthen et al., 2020) and the number of points used to compute the coefficients (*i.e.* the experimental design). It is known that informative likelihood functions have quasi-compact supports (*i.e.* $\mathcal{L}(\boldsymbol{X}) \approx 0$ on a majority of $\mathcal{D}_{\boldsymbol{X}}$). Such functions require a very high polynomial degree to be approximated accurately, which in turn can lead to the need for prohibitively large experimental designs.

1.5 Stochastic spectral likelihood embedding

An extension of SLE (see Section 1.4), namely *stochastic spectral likelihood embedding* (SSLE), was recently proposed in Wagner et al. (2021). This approach, similarly to SLE, directly approximates the likelihood function, but replaces the global spectral expansion in Eq. (1.51) for an SSE metamodel ((Marelli et al., 2021), see also UQLAB User Manual – Stochastic spectral embedding). Due to their local approximation strength, SSE metamodels are more suitable for approximating functions with quasi-compact supports than purely global approximation approaches.

Again, viewing the likelihood as a function of a random vector $\boldsymbol{X}$ with independent components, we can write its SSLE representation as:

$$\mathcal{L}(\boldsymbol{X}) \approx \mathcal{L}_{\mathrm{SSLE}}(\boldsymbol{X}) \stackrel{\mathrm{def}}{=} \sum_{k\in\mathcal{K}} \mathbf{1}_{\mathcal{D}_{\boldsymbol{X}}^k}(\boldsymbol{X})\, \widehat{\mathcal{R}}_{\mathrm{S}}^k(\boldsymbol{X}), \tag{1.57}$$

where

$$\widehat{\mathcal{R}}_{\mathrm{S}}^k(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}^k} a_{\boldsymbol{\alpha}}^k \Psi_{\boldsymbol{\alpha}}^k(\boldsymbol{X}) \tag{1.58}$$

are residual expansions.

The variable $\boldsymbol{X}$ is distributed according to the prior distribution $\pi(\boldsymbol{x})$ and, consequently, the local basis $\Psi_{\boldsymbol{\alpha}}^k$ is orthonormal w.r.t. that distribution.

Due to the local spectral properties of the residual expansions, the SSLE representation of the likelihood function retains all of the post-processing properties of SLE (see Section 1.4):

Evidence The normalization constant Z emerges as the sum of the constant polynomial coefficients weighted by the prior mass:

$$Z = \sum_{k\in\mathcal{K}} \sum_{\boldsymbol{\alpha}\in\mathcal{A}^k} a_{\boldsymbol{\alpha}}^k \int_{\mathcal{D}_{\boldsymbol{X}}^k} \Psi_{\boldsymbol{\alpha}}^k(\boldsymbol{x})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \sum_{k\in\mathcal{K}} \mathcal{V}^k a_{\boldsymbol{0}}^k, \quad \text{where} \quad \mathcal{V}^k = \int_{\mathcal{D}_{\boldsymbol{X}}^k} \pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{1.59}$$

Posterior This allows us to write the posterior density as

$$\pi(\boldsymbol{x}|\mathcal{Y}) \approx \frac{\mathcal{L}_{\mathrm{SSLE}}(\boldsymbol{x})\pi(\boldsymbol{x})}{Z} = \frac{\pi(\boldsymbol{x})}{\sum_{k\in\mathcal{K}} \mathcal{V}^k a_{\boldsymbol{0}}^k} \sum_{k\in\mathcal{K}} \mathbf{1}_{\mathcal{D}_{\boldsymbol{X}}^k}(\boldsymbol{x}) \widehat{\mathcal{R}}_{\mathrm{S}}^k(\boldsymbol{x}). \tag{1.60}$$

Posterior marginals Utilizing the disjoint sets $\mathbf{u}$ and $\mathbf{v}$ from Eq. (1.54) it is also possible to analytically derive posterior marginal PDFs as

$$\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{X}_{\mathbf{v}}}} \pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x}_{\mathbf{v}} \approx \frac{\pi_{\mathbf{u}}(\boldsymbol{x}_{\mathbf{u}})}{\sum_{k\in\mathcal{K}} \mathcal{V}^k a_{\boldsymbol{0}}^k} \sum_{k\in\mathcal{K}} \mathbf{1}_{\mathcal{D}_{\boldsymbol{X}_{\mathbf{u}}}^k}(\boldsymbol{x}_{\mathbf{u}}) \widehat{\mathcal{R}}_{S,\mathbf{u}}^k(\boldsymbol{x}_{\mathbf{u}}) \mathcal{V}_{\mathbf{v}}^k \tag{1.61}$$

where

$$\widehat{\mathcal{R}}^k_{S,\mathbf{u}}(\boldsymbol{x}_\mathbf{u}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}^k_{\mathbf{v}=0}} a^k_{\boldsymbol{\alpha}}\Psi^k_{\boldsymbol{\alpha}}(\boldsymbol{x}_\mathbf{u}) \quad \text{and} \quad \mathcal{V}^k_\mathbf{v} = \int_{\mathcal{D}^k_{\boldsymbol{X}_\mathbf{v}}} \pi_\mathbf{v}(\boldsymbol{x}_\mathbf{v})\,\mathrm{d}\boldsymbol{x}_\mathbf{v}. \tag{1.62}$$

$\widehat{\mathcal{R}}^k_{S,\mathbf{u}}(\boldsymbol{x}_\mathbf{u})$ is a subexpansion of $\widehat{\mathcal{R}}^k_S(\boldsymbol{x})$ that contains only non-constant polynomials along the dimensions $i \in \mathbf{u}$. Note that, as we assumed that the prior distribution has independent components, the constants $\mathcal{V}^k$ and $\mathcal{V}^k_\mathbf{v}$ are obtained as products of univariate integrals which are available analytically from the prior marginal cumulative distribution functions (CDFs).

Quantities of interest Posterior expectations of a function $h(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}^k} b^k_{\boldsymbol{\alpha}}\Psi^k_{\boldsymbol{\alpha}}(\boldsymbol{x})$ for $k \in \mathcal{K}$ can be approximated by:

$$\begin{aligned}\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] &= \int_{\mathcal{D}_{\boldsymbol{X}}} h(\boldsymbol{x})\pi(\boldsymbol{x}|\mathcal{Y})\,\mathrm{d}\boldsymbol{x} \\ &= \frac{1}{Z}\sum_{k\in\mathcal{K}}\sum_{\boldsymbol{\alpha}\in\mathcal{A}^k} a^k_{\boldsymbol{\alpha}}\int_{\mathcal{D}^k_{\boldsymbol{X}}} h(\boldsymbol{x})\Psi^k_{\boldsymbol{\alpha}}(\boldsymbol{x})\pi(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\ &= \frac{1}{Z}\sum_{k\in\mathcal{K}}\sum_{\boldsymbol{\alpha}\in\mathcal{A}^k} a^k_{\boldsymbol{\alpha}} b^k_{\boldsymbol{\alpha}},\end{aligned} \tag{1.63}$$

where $b^k_{\boldsymbol{\alpha}}$ are the coefficients of the PCE of h in the $\mathrm{card}(\mathcal{K})$ bases $\{\Psi^k_{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha}\in\mathcal{A}^k}$. The same expression can also be used for computing posterior moments like mean, variance or covariance.

These expressions can be seen as a generalization of the ones for SLE detailed in Section 1.4. For a single-level global expansion (*i.e.* $\mathrm{card}(\mathcal{K}) = 1$) and consequently $\mathcal{V}^{(0,1)} = 1$, they are identical.

1.5.1 Sequential partitioning algorithm

The algorithm to construct SSEs implemented in UQLAB is called *sequential partitioning algorithm*. It sequentially partitions selected refinement domains and constructs local expansions of the residual to ultimately produce a likelihood approximation of the form shown in Eq. (1.57). This algorithm is explained in detail in the UQLAB User Manual – Stochastic spectral embedding. In Wagner et al. (2021), modifications to this algorithm were proposed that were shown to improve the SSE approximation accuracy for likelihood functions in SSLE. These modifications pertain to the *partitioning* and *sample enrichment strategies*.

1.5.1.1 Partitioning strategy

The partitioning strategy determines how a selected refinement domain is split. For likelihood functions, it was proposed in Wagner et al. (2021) to pick the split direction along which a

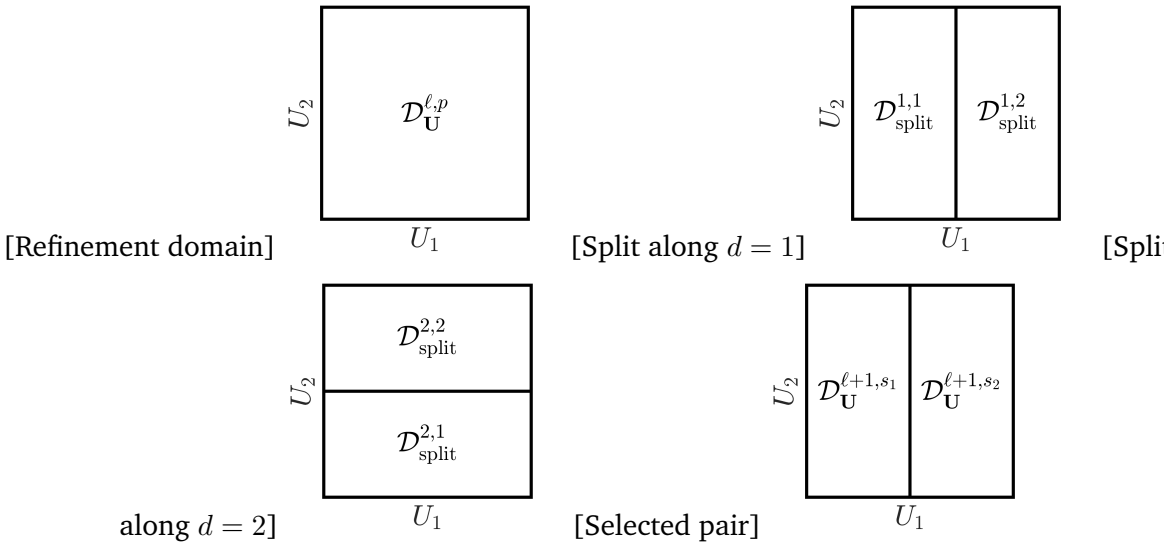


Figure 3: Partitioning strategy for a 2D example visualized in the quantile space $\boldsymbol{U}$. The refinement domain $\mathcal{D}_{\boldsymbol{U}}^{\ell,p}$ is split into two subdomains $\mathcal{D}_{\boldsymbol{U}}^{\ell+1,s_1}$ and $\mathcal{D}_{\boldsymbol{U}}^{\ell+1,s_2}$.

split yields a maximum difference in the residual empirical variance between the two candidate subdomains created by the split. This can easily be visualized with an example given by the $M = 2$ dimensional domain $\mathcal{D}_{\boldsymbol{X}}^{\ell,p}$ in Figure **??**. Assume this subdomain was selected as the refinement domain. To decide along which dimension to split, we construct the M candidate subdomain pairs $\{\mathcal{D}_{\text{split}}^{i,1}, \mathcal{D}_{\text{split}}^{i,2}\}_{i=1,\dots,M}$ and estimate the corresponding $\{E_{\text{split}}^{i}\}_{i=1,\dots,M}$ in those subdomains defined by

$$E_{\text{split}}^{i} \overset{\text{def}}{=} \left| \text{Var}\left[\mathcal{R}^{\ell+1}(\mathcal{X}_{\text{split}}^{i,1})\right] - \text{Var}\left[\mathcal{R}^{\ell+1}(\mathcal{X}_{\text{split}}^{i,2})\right] \right|. \tag{1.64}$$

In this expression, $\mathcal{X}_{\text{split}}^{i,1}$ and $\mathcal{X}_{\text{split}}^{i,2}$ denote subsets of the experimental design $\mathcal{X}$ that lie within the subdomains $\mathcal{D}_{\text{split}}^{i,1}$ and $\mathcal{D}_{\text{split}}^{i,2}$ respectively. The corresponding variances can be estimated with the empirical variance of the residuals in the respective candidate subdomains.

After computing the residual variance differences, the split is carried out along dimension

$$d = \arg \max_{i \in \{1,\dots,M\}} E_{\text{split}}^{i}, \tag{1.65}$$

i.e. to keep the subdomains $\mathcal{D}_{\text{split}}^{d,1}$ and $\mathcal{D}_{\text{split}}^{d,2}$ that introduce the largest difference in variance. For $d = 1$, the resulting split can be seen in Figure **??**.

1.5.1.2 Sample enrichment

Likelihood functions are typically not equally informative at every point of the input space. On the contrary, it is often the case that large input regions are close to zero, while only a small subdomain of the input space yields likelihood responses that are multiple orders of

magnitude larger.

It was shown in Wagner et al. (2021) that it is therefore more efficient in terms of total number of likelihood evaluations, to sequentially sample the experimental design in SSLE (see Section 1.3.3 in UQLAB User Manual – Stochastic spectral embedding). The rationale, is to enrich the experimental design only in domains that have been selected for refinement. Informative regions of the likelihood function (that are more difficult to approximate) will then receive a greater share of likelihood evaluations, because the refinement domain selection in the sequential partitioning algorithm is based on the local approximation error (see Section 1.3.1 in UQLAB User Manual – Stochastic spectral embedding).

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 4. The beam has a known rectangular cross-section of width b and height h and a known span of length L.

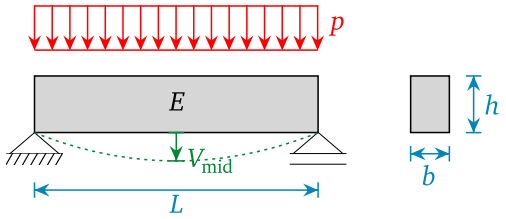


Figure 4: Simple beam bending test.

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 (N/m)	12 000

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:

$$E \sim \mathcal{LN}(\lambda, \zeta), \quad \text{with} \quad \mu_E = 30 \text{ (GPa) and } \sigma_E = 4.5 \text{ (GPa)}. \tag{2.2}$$

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 $\boldsymbol{y}_i \in \mathbb{R}^{N_{\text{out}}}$ 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 = [30 4.5]*1e9 ; % (N/m^2)

PriorOpts.Marginals(5).Name = 'p';
PriorOpts.Marginals(5).Type = 'Constant';
PriorOpts.Marginals(5).Parameters = 12000; % (N/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). 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 $N \times N_{\text{out}}$ 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 $N_{\text{out}} = 1$, 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: Without an 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 $\sigma^2 \sim \mathcal{U}(0, \mu_{\mathcal{Y}}^2)$ 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). By default in UQLAB the first half of the sample points generated by all chains are removed as burn-in (see Section 1.3.5.4) and the empirical parameter mean $\mathbb{E}[\boldsymbol{X}|\mathcal{Y}]$ along with the 5th and 95th

percentile based on the post-processed sample are estimated. Additionally, samples from the prior distribution and from the posterior predictive distribution are drawn (see Section 1.2.6) and stored in the `myBayesianAnalysis` object. For all available post-processing options see Section 2.4.1.6 and Section 3.3.

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):                           00:00:41
   Number of sample points:                       3.00e+04

%------------------- Posterior Marginals
-------------------------------------------------------------------------
| Parameter | Mean    | Std     | (0.05-0.95) Quant.   | Type         |
-------------------------------------------------------------------------
| E         | 2.4e+10 | 1.7e+09 | (2.2e+10 - 2.7e+10) | Model        |
| Sigma2    | 3.7e-06 | 1.1e-05 | (1.3e-07 - 1.6e-05) | Discrepancy |
-------------------------------------------------------------------------


%------------------- Point estimate
----------------------------------------
| Parameter | Mean    | Parameter Type |
----------------------------------------
| E         | 2.4e+10 | Model          |
| Sigma2    | 3.7e-06 | Discrepancy    |
----------------------------------------
```

The results can also be visualized by:

```
uq_display(myBayesianAnalysis)
```

which produces the images in Figure 5.

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 $\sigma^2 = 10^{-7}$ (m^2)):

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = 1e-7; % (m^2)
```

Which is then passed to `BayesOpts` by:

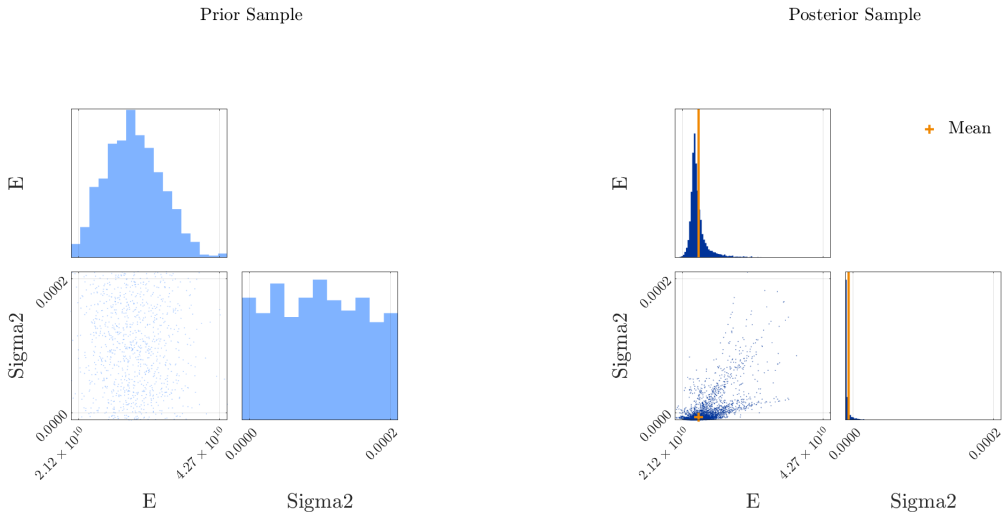
```
BayesOpts.Discrepancy = DiscrepancyOpts;
```

2.2.6.2 Unknown residual variance

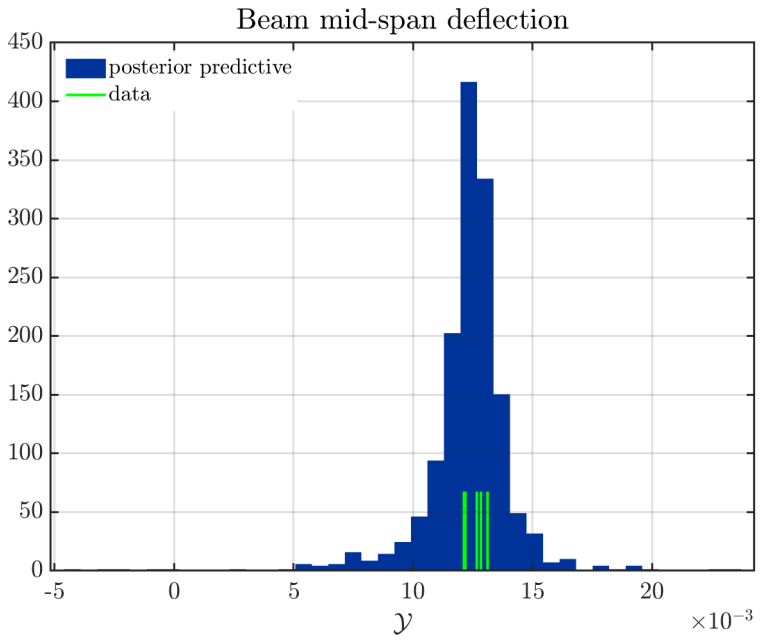
If σ^2 is not known a priori, as detailed in Section 1.2.3, the Bayesian framework can infer the distribution of the discrepancy parameter $\boldsymbol{x}_\varepsilon \equiv \sigma^2$. This requires the initial specification of a prior distribution of the discrepancy parameter $\pi(\sigma^2)$ (see Eq.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



(a) Scatterplots of prior and posterior sample with the posterior mean point estimator from (1.31).



(b) Posterior predictive distribution from (1.28), data, and model at mean prediction obtained by propagating the mean point estimator through the forward model.

Figure 5: Visualize analysis: The results of the Bayesian inverse analysis on the input and the model predictions.

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

Note: Here a uniform prior on σ^2 with bounds $\mathcal{U}(0, \mu^2_{V_{\text{mid}}})$ 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 6.

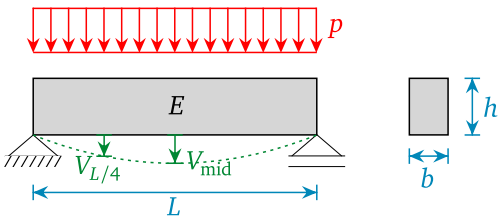


Figure 6: Simple beam bending test.

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 $V_{L/4}$, re-

spectively. 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 Eqs. (2.1) and (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 $N_{\text{out}} = 2$.

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 $N \times N_{\text{out}}$ 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: Without an 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 $\boldsymbol{x}_\varepsilon = (\sigma_1^2, \ldots, \sigma_{N_{\text{out}}}^2)$. The prior distribution is by default taken as independent uniform distributions $\pi(\boldsymbol{x}_\varepsilon) = \prod_{i=1}^{N_{\text{out}}} \pi(\sigma_i^2)$ 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);
```

To generate a prior predictive sample, call:

```
uq_postProcessInversion(myBayesianAnalysis,'priorPredictive',1000);
```

see Section 2.4.1.6 and Section 3.3 for all available post-processing options.

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
                                                  2

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

   Algorithm:                                     AIES
   Duration (HH:MM:SS):                           00:00:47
   Number of sample points:                       3.00e+04

%------------------- Posterior Marginals
-------------------------------------------------------------------------
| Parameter | Mean     | Std      | (0.05-0.95) Quant.   | Type        |
-------------------------------------------------------------------------
| E         | 2.3e+10  | 7e+08    | (2.3e+10 - 2.5e+10)  | Model       |
| Sigma2    | 2e-06    | 6.3e-06  | (4.7e-08 - 9.1e-06)  | Discrepancy |
| Sigma2    | 6e-06    | 1.8e-05  | (1.3e-07 - 3.4e-05)  | Discrepancy |
-------------------------------------------------------------------------

%------------------- Point estimate
---------------------------------------
| Parameter | Mean     | Parameter Type |
---------------------------------------
```

```
| E          | 2.3e+10 | Model          |
| Sigma2     | 2e-06   | Discrepancy    |
| Sigma2     | 6e-06   | Discrepancy    |
-----------------------------------------

%------------------- Correlation matrix (Discrepancy Parameters)
-------------------------------
|         |  Sigma2    Sigma2  |
-------------------------------
| Sigma2 |  1          -0.047  |
| Sigma2 |  -0.047     1       |
-------------------------------
```

and visualized graphically with:

```
uq_display(myBayesianAnalysis)
```

which produces a set of plots similar to those in Figure 7.

2.3.4 Advanced options: discrepancy model

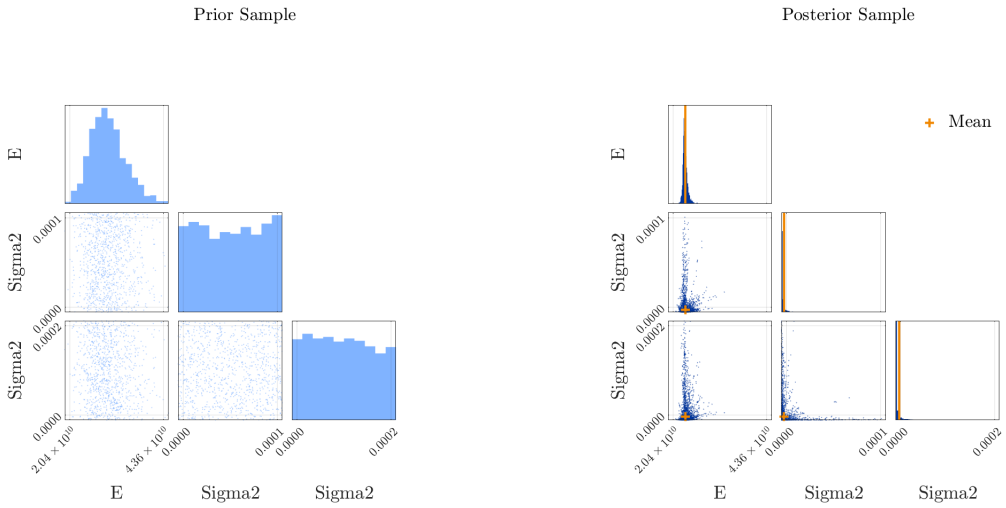
In the case of models with multiple outputs, the full covariance matrix $\mathbf{\Sigma}$ of the residual discrepancy vector $\boldsymbol{\varepsilon} = (\varepsilon_i, \ldots, \varepsilon_{N_{\text{out}}})$ has to be defined for the likelihood function. In this section, all options that are available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

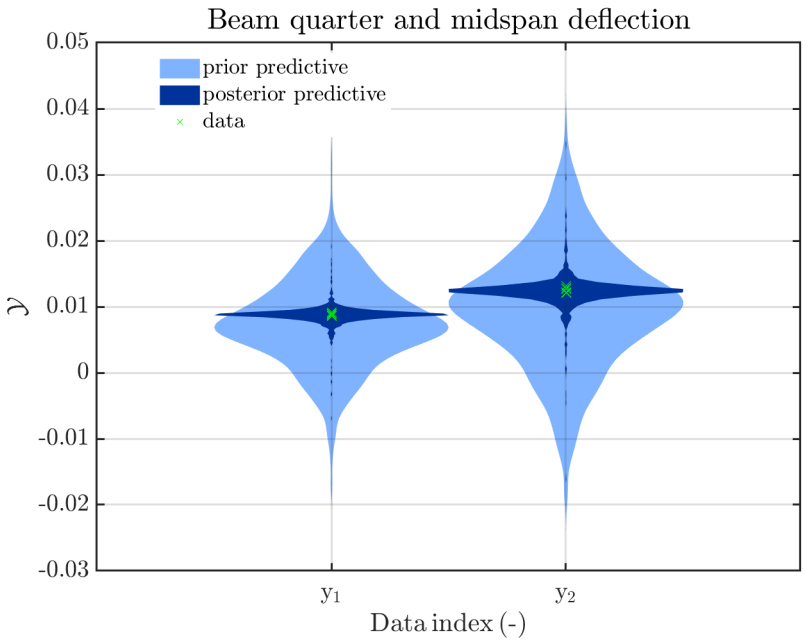
2.3.4.1 Known residual variance

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 $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_{N_{\text{out}}})$. 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.*, $\sigma^2 = 10^{-7}$ (m^2)), this can specified by a `DiscrepancyOpts` structure like



(a) Scatterplots of prior and posterior sample with the posterior mean point estimator from (1.31).



(b) Prior and posterior predictive distributions from (1.27) and (1.28), data, and model at mean prediction obtained by propagating the mean point estimator through the forward model.

Figure 7: Advanced discrepancy options: The results of the Bayesian inverse analysis on the input and the model predictions.

```
DiscrepancyOpts.Type = 'Gaussian';
DiscrepancyOpts.Parameters = 1e-7; % 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-7 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-7  -5e-8;...
                              -5e-8   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]; % (m^2)
myDiscrepancyPrior = uq_createInput(DiscrepancyPriorOpts);

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

Here a uniform prior $\pi(\sigma^2) = \mathcal{U}(0, 10^{-4})$ (m^2) 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 $N_{\text{out}} = 2$, 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];% (m^2)
DiscrepancyPriorOpts.Marginals(2).Name = 'Sigma2_2';
DiscrepancyPriorOpts.Marginals(2).Type = 'Uniform';
DiscrepancyPriorOpts.Marginals(2).Parameters = [0, 1e-4];% (m^2)
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 (see Section 1.2.4).

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 $\sigma_1^2 = 10^{-7}$ (m^2), while for the second residual ε_2 the variance σ_2^2 is unknown and assigned a uniform distribution $\sigma_2^2 \sim \mathcal{U}(0, 10^{-4})$ (m^2). These discrepancy options can be passed to UQLAB by defining $N_{\mathrm{gr}} = 2$ 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-7;% (m^2)

% 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];% (m^2)
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 $i \in \{1, \ldots, N_{\mathrm{out}}\}$ 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 discrepancy 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 $a = 2$.

The Bayesian module offers four different solvers that are described in more detail next:

- MCMC: Section 2.4.1
- SLE: Section 2.4.2
- SSLE: Section 2.4.3
- None: Section 2.4.4

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 solver 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 and Table 12), which can be specified by defining a `myProposal` structure (see also Table 14).

If the algorithm is to use a Gaussian proposal distribution centered at the previous sample (standard random walk algorithm), the `myProposal` structure should only contain one field `.PriorScale` that can, for instance, be set to:

```
myProposal.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 fully specified:

```
myProposal.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 `myProposal` structure can also contain two fields:

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

where `myProposalDistribution` 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 `myProposalDistribution` 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.

Finally, the proposal structure needs to be assigned to the `Solver.MCMC.Proposal` field:

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

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

where the `myProposal` structure 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 `myProposal` before switching to the Gaussian distribution with the empirical covariance matrix as detailed in Eq. (1.34). 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. (1.43). 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 8 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:

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

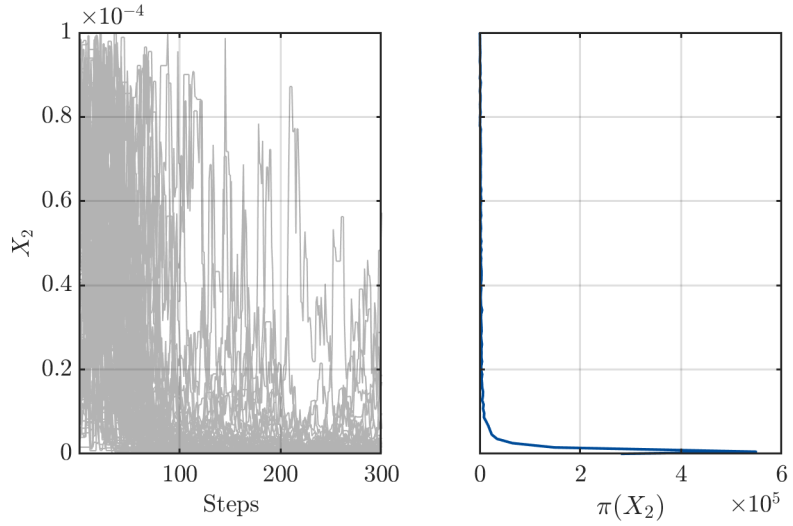


Figure 8: Trace plot and corresponding KDE during execution of the MCMC algorithm.

```
    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 is automatically done with the `uq_postProcessInversionMCMC` function that is called for MCMC analyses by the wrapper function:

```
uq_postProcessInversion(myBayesianAnalysis)
```

Since UQLAB v1.3.0 this function is called automatically after every analysis and 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, the covariance matrix is estimated, and samples are drawn from the prior distribution and posterior predictive distribution.

These post-processing results are then stored in the `.PostProc` structure inside the `.Results` structure. If `uq_postProcessInversion` is called with all possible options (see Section 3.3), it contains the following fields

```
myBayesianAnalysis.Results.PostProc =

struct with fields:

                PostSample: [T"xMxC"] double]
          PostLogLikeliEval: [T"xC"] double]
              PostModel: [1x1 struct]
```

```
                Dependence: [1x1 struct]
               Percentiles: [1x1 struct]
            PostPredSample: [1x1 struct]
           PriorPredSample: [1x1 struct]
               PriorSample: [PxM double]
             PointEstimate: [1x1 struct]
                     MPSRF: double
```

where `T"` is the length of the MCMC chains without the burn in, `C"` is the number of chains excluding the `badChains` and `P` is the number of drawn prior sample points.

For more control over the post-processing operations, it is recommended to adapt the post-processing options to the problem at hand and recall the `uq_postProcessInversion` function after the analysis is complete. For example, to draw $1{,}000$ sample points from the prior predictive distribution, the function needs to be called with:

```
uq_postProcessInversion(myBayesianAnalysis,'priorPredictive', 1000)
```

`uq_postProcessInversion` always operates on the original `Sample` array and overwrites the contents of the respective `PostProc` field when called repeatedly. As an example, after calling

```
uq_postProcessInversion(myBayesianAnalysis,'burnIn', 0.6)
```

`T"` will be 40% of the original T. After calling

```
uq_postProcessInversion(myBayesianAnalysis,'burnIn', 0.7)
```

`T"` will be 30% of the original T, independent of the previous call.

To maintain previously computed content of `PostProc`'s fields, `uq_postProcessInversion` only overwrites fields if the call produces new content or an output is explicitly turned off by the user. This does not happen with the default options of `uq_postProcessInversion` (see Section 3.3). In order to delete a previously computed post-processing result, it needs to be removed explicitly *e.g.*:

```
uq_postProcessInversion(myBayesianAnalysis,'priorPredictive', 0)
```

removes a previously computed prior predictive sample while any other call would leave it untouched.

An exhaustive list of available post-processing options can be found in Section 3.3.

2.4.2 Spectral likelihood expansion

To solve the Bayesian problem with the spectral likelihood expansion (SLE) technique (see Section 1.4), the user should specify the `Solver` structure as follows:

```
Solver.Type = 'SLE'
```

The Bayesian module of UQLAB uses the polynomial chaos expansion module (PCE, UQLAB User Manual – Polynomial Chaos Expansions) to construct the likelihood approximation. Additional options for the PCE can be passed to UQLAB through the following syntax:

```
Solver.SLE.Degree = 0:20
Solver.SLE.ExpDesign.NSamples = 1e4;
```

which specifies a degree adaptive PCE to be computed based on an experimental design of size 10^4. In general, `SLE` accepts all options that can be passed to the `MetaOpts` structure in UQLAB User Manual – Polynomial Chaos Expansions (*e.g.*`TruncOptions`). See Table 18 for more information.

To conduct an SLE-based inverse analysis, the following command should be executed (after assigning the `Solver` structure to the `BayesOpts.Solver` field):

```
uq_createAnalysis(BayesOpts)
```

By default, the SLE-specific post-processing function `uq_postProcessInversionSLE` is called after every SLE analysis by the wrapper function `uq_postProcessInversion`: it computes the Bayesian evidence (see Eq. (1.52)) and the posterior mean (see Eq. (1.56)). These values are stored in the `myBayesianAnalysis.Results.PostProc` field that reads

```
myBayesianAnalysis.Results.PostProc =

struct with fields:

          Evidence: 2.9561e+23
              Mean: [2.2524e+10 5.1988e-06]
     PointEstimate: [1x1 struct]
         Posterior: [function_handle]
```

To evaluate the SLE-based posterior density approximation on 1000 points, the function handle `Posterior` can be used with

```
X = uq_getSample(myBayesianAnalysis.PriorDist, 1000);
postDensity = myBayesianAnalysis.Results.PostProc.Posterior(X);
```

To additionally compute the posterior covariance and correlation matrices, the post-processing function can be called again with the following arguments:

```
uq_postProcessInversion(myBayesianAnalysis, 'dependence', true).
```

The results of the analysis can be inspected with the `uq_print` or the `uq_display` function. To generate the plots shown in Figure 9 the following command should be executed:

```
uq_display(myBayesianAnalysis)
```

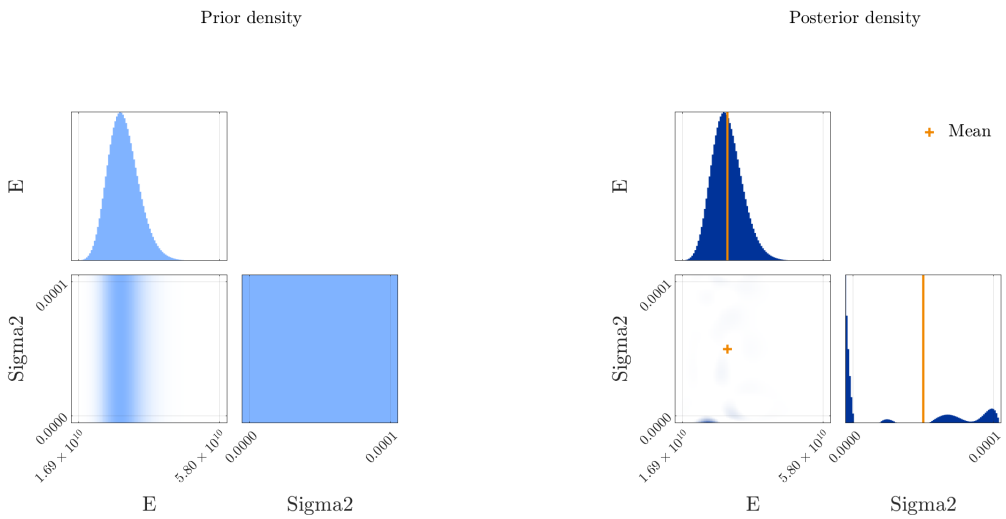


Figure 9: SLE solver - density plots of prior and posterior distribution with the posterior mean point estimator from (1.56).

2.4.3 Stochastic spectral likelihood embedding

To solve the Bayesian problem with the stochastic spectral likelihood embedding (SSLE) technique (see Section 1.5), the user should specify the `Solver` structure as follows:

```
Solver.Type = 'SSLE'
```

The Bayesian module of UQLAB uses the stochastic spectral embedding module (SSE, UQLAB User Manual – Stochastic spectral embedding) to construct the likelihood approximation. Additional options for the SSE can be passed to UQLAB through the following syntax:

```
% Expansion options
Solver.SSLE.ExpOptions.Degree = 0:4;

% Experimental design options
Solver.SSLE.ExpDesign.NSamples = 1000;
Solver.SSLE.ExpDesign.NEnrich = 100;
```

which specifies degree adaptive PCE for the residual expansions with an experimental design of size 1000. By default, the experimental design is added sequentially (see Section 1.5.1.2), where we now specified an enrichment rate of 100 sample points per refinement step. In general, `SSLE` accepts all options that can be passed to the `metaOpts` structure in the UQLAB User Manual – Stochastic spectral embedding (*e.g.*`Partitioning`). See Table 19 for more information.

By default, SSLE uses a partitioning strategy based on the residual variance difference (see Section 1.5.1.1) that is implemented in the function `uq_SSE_partitioning_varDiff`, but other strategies can be selected as well (see Section 3.2.2 of UQLAB User Manual – Stochastic

spectral embedding).

To conduct an SSLE-based inverse analysis, the following command should be executed (after assigning the `Solver` structure to the `BayesOpts.Solver` field):

```
uq_createAnalysis(BayesOpts)
```

By default, the SSLE-specific post-processing function `uq_postProcessInversionSSLE` is called after every SLE analysis by the wrapper function `uq_postProcessInversion`: it computes the Bayesian evidence (see Eq. (1.59)) and the posterior mean (see Eq. (1.63)). These values are stored in the `myBayesianAnalysis.Results.PostProc` field:

```
myBayesianAnalysis.Results.PostProc =

struct with fields:

         Evidence: 2.2327e+23
             Mean: [2.4101e+10 4.6643e-06]
    PointEstimate: [1x1 struct]
        Posterior: [function_handle]
```

To evaluate the SSLE-based posterior density approximation on a sample of size $1,000$ of the prior distribution, the function handle `Posterior` can be used with

```
X = uq_getSample(myBayesianAnalysis.PriorDist, 1000);
postDensity = myBayesianAnalysis.Results.PostProc.Posterior(X);
```

To additionally compute the posterior covariance and correlation matrices, the post-processing function can be called again with the following arguments:

```
uq_postProcessInversion(myBayesianAnalysis, 'dependence', true).
```

The results of the analysis can be inspected with the `uq_print` or the `uq_display` function. To generate the plots shown in Figure 10 the following command should be executed:

```
uq_display(myBayesianAnalysis)
```

2.4.4 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:

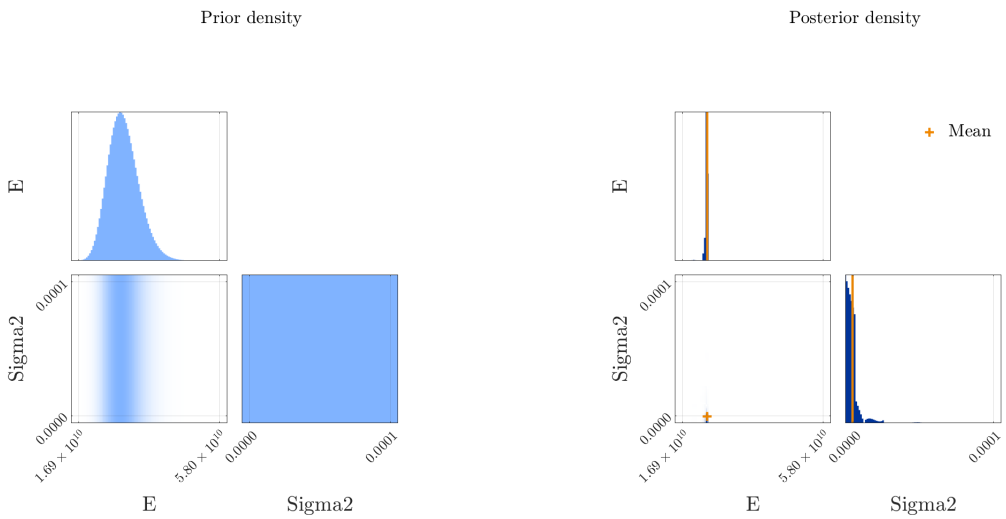


Figure 10: SSLE solver - density plots of prior and posterior distribution with the posterior mean point estimator from (1.56).

```
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 an 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.

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 11.

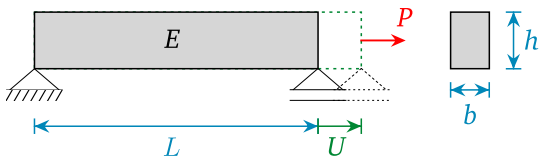


Figure 11: Tensile test.

The nominal value of the load P is $50\ kN$. 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)	0.235	0.236	0.229

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

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

In this case, the measurements were carried out with a measuring device that has a known measurement error distribution of $\varepsilon \sim \mathcal{N}(0, 2 \cdot 10^{-11})$ (m).

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 = 50000; % (N)

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 list the desired parameters properly.

2.5.3 Provide measurements

The results of the tensile test are stored in a vector `U` and are assigned to the second data group `myData(2)`:

```
U = [0.235; 0.236; 0.229]/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]; % (m^2)
myDiscrepancyPrior1 = uq_createInput(DiscrepancyPriorOpts1);

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

% Discrepancy group 2
DiscrepancyOpts(2).Type = 'Gaussian';
DiscrepancyOpts(2).Parameters = 2e-11; % (m^2) known discr. 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 = myPriorDist;
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 the `ForwardModels` structure array, as there are multiple forward models now.

The inverse problem can then be solved by running the analysis:

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

Run post-processing with non-default options by removing 30%, instead of the default 50%, of the initially generated sample points:

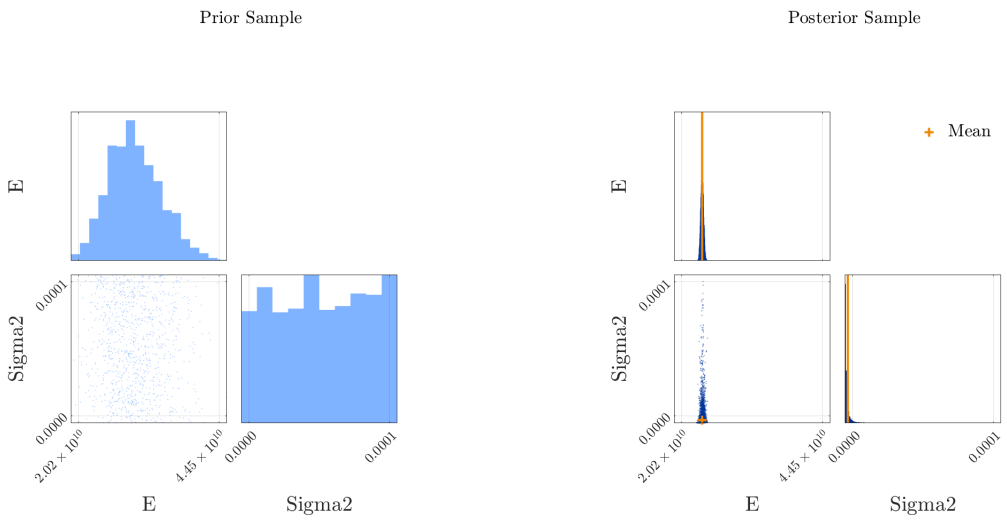
```
uq_postProcessInversion(myBayesianAnalysis, 'burnIn', 0.3);
```

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

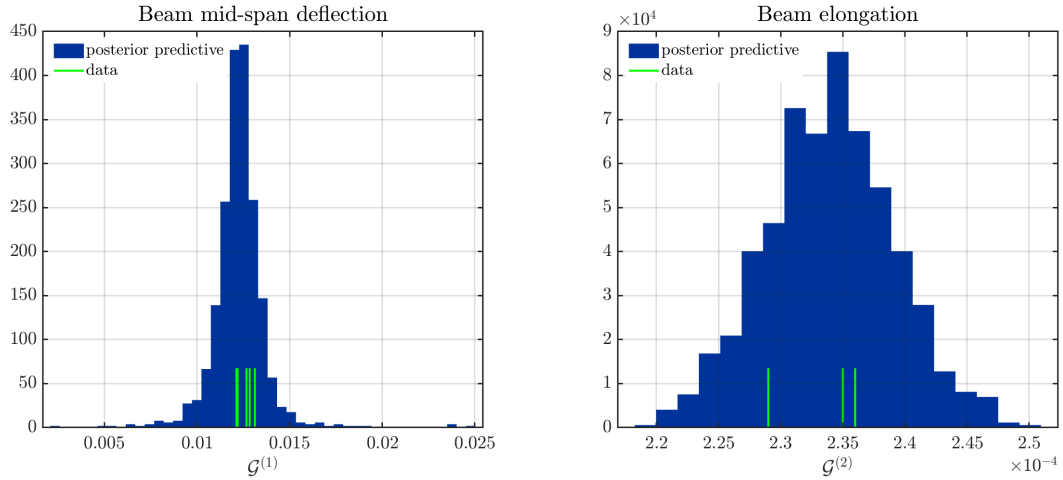
```
uq_print(myBayesianAnalysis)
```

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):                           00:00:55
   Number of sample points:                       3.00e+04

%------------------- Posterior Marginals
-------------------------------------------------------------------------
| Parameter | Mean    | Std     | (0.05-0.95) Quant.  | Type         |
-------------------------------------------------------------------------
| E         | 2.4e+10 | 5.7e+07 | (2.4e+10 - 2.4e+10) | Model        |
| Sigma2    | 2.3e-06 | 5.6e-06 | (2.2e-07 - 8.3e-06) | Discrepancy  |
-------------------------------------------------------------------------

%------------------- Point estimate
----------------------------------------
| Parameter | Mean    | Parameter Type |
----------------------------------------
| E         | 2.4e+10 | Model          |
| Sigma2    | 2.3e-06 | Discrepancy    |
----------------------------------------
```



(a) Scatterplots of prior and posterior sample with the posterior mean point estimator from (1.31).



(b) Posterior predictive distribution from (1.28), data, and model at mean prediction obtained by propagating the mean point estimator through the forward model for both data groups.

Figure 12: Multiple forward models: The results of the Bayesian inverse analysis on the input and the model predictions.

A visualization of the analysis can be produced by:

```
uq_display(myBayesianAnalysis)
```

which produces the images shown in Figure 12.

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`).

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 semantically group configuration options and output quantities. Due to the complexity of the algorithms implemented, it is not uncommon to employ nested structures to fine-tune the inputs and outputs. Throughout this reference guide, a table-based description of the configuration structures is adopted.

The simplest case is given when a field of the structure is a simple value or 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, from left to right, correspond to the name, the 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, a link to a table that describes the available options is provided, 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` and `doubles` mean a real number represented in double precision and a set of such real numbers, respectively.

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 modelling
●	`.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 solvers.
		`'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 structure array of size N_{gr}. The g-th structure in this array defines the g-th data group $\mathcal{G}^{(g)}$ used in solving the inverse problem. $N_{\text{out},g}$ is the number of model outputs related to the current data group (Section 2.3.4.3):

Table 6: `BayesOpts.Data`

●	`.y`	$N \times N_{\text{out},g}$ Double or Arbitrary	The observations $\mathcal{G}^{(g)}$. 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 g-th data group to specific model outputs. See also Section 2.3.4.3.
		$1 \times N_{\text{out},g}$ Integer default: consecutive numbering $1 : N_{\text{out}}$	Vector of model output indices related to the g-th data group. If $N_{\text{mod}} > 1$ the definition below has to be used.
		$2 \times N_{\text{out},g}$ 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 1.2.4 and 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 g-th structure in this array defines the g-th discrepancy group used in solving the inverse problem. $N_{\text{out},g}$ 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
		$1 \times N_{\text{out},g}$ double	Independent discrepancies with individual variances
		$N_{\text{out},g} \times N_{\text{out},g}$ 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 1.2.4 and 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.
		`'SLE'`	Spectral likelihood expansion. See also Section 2.4.2.
		`'SSLE'`	Stochastic spectral likelihood embedding. See also Section 2.4.3.
		`'None'`	Only initialize and provide handles. See also Section 2.4.4.
☐	`.MCMC`	Table 10	Parameters of the MCMC algorithm.
☐	`.SLE`	Table 18	Options for the PCE used in SLE. See UQLAB User Manual – Polynomial Chaos Expansions.
☐	`.SSLE`	Table 19	Options for the SSE used in SSLE. See UQLAB User Manual – Stochastic spectral embedding.

3.1.4.1 Markov chain Monte Carlo

If the solver type is `'MCMC'`, the field `BayesOpts.Solver.MCMC` accepts the additional options listed in Table 10.

Table 10: `BayesOpts.Solver.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: MH, AM: 3,000 HMC, AIES: 300	Number of MCMC iterations T per chain
☐	.Visualize	Table 17	MCMC runtime visualization. See also Section 2.4.1.5
⊞	.NChains	Integer scalar default: MH, AM, HMC: 10 AIES: 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	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.Solver.MCMC (Metropolis-Hastings)

☐	.Proposal	Table 14	Proposal distribution $p(\boldsymbol{x}\|\boldsymbol{x}^{(t)})$

Table 13: BayesOpts.Solver.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 $\mathbf{\Sigma}_p$ for Gaussian proposal centered at $\boldsymbol{x}^{(t)}$.
⊞	`.Cov`	$M \times M$ Double	Full covariance matrix $\mathbf{\Sigma}_p$
⊞	`.Distribution`	INPUT object	Custom proposal distribution. Requires also `.Conditioning` 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 `.Distribution`.

Table 15: `BayesOpts.Solver.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.Solver.MCMC (Affine invariant ensemble sampler)`

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

Table 17: `BayesOpts.Solver.MCMC.Visualize`

☐	`.Parameters`	Integer	Which parameter should be visualized
☐	`.Interval`	Integer scalar	Plot interval

3.1.4.2 Spectral likelihood expansions

If the solver type is `'SLE'`, the field `BayesOpts.Solver.SLE` accepts the additional options listed in Table 18. These options pertain to the properties of the PCE used in SLE and the table lists only the options that are set by default. Additional options can be found in the UQLAB User Manual – Polynomial Chaos Expansions.

Table 18: `BayesOpts.Solver.SLE`

☐	`.MetaType`	String default: `'PCE'`	Type of the expansion. Currently only `'PCE'` is supported.
☐	`.Method`	String default: `'LARS'`	PCE calculation strategy. See the UQLAB User Manual – Polynomial Chaos Expansions for available options.
☐	`.Degree`	Integer array default: `0:15`	Adaptively selected degree of the PCE.
●	`.ExpDesign`	`Structure`	Properties of the experimental design according to Table 11 of the UQLAB User Manual – Polynomial Chaos Expansions.
☐	`.[...]`		Additional options can be selected according to Table 3 of the UQLAB User Manual – Polynomial Chaos Expansions.

3.1.4.3 Stochastic spectral likelihood embedding

If the solver type is `'SSLE'`, the field `BayesOpts.Solver.SSLE` accepts the additional options listed in Table 19. These options pertain to the properties of the SSE used in SSLE and the table lists only the options that are set by default. Additional options can be found in the UQLAB User Manual – Stochastic spectral embedding.

Table 19: `BayesOpts.Solver.SSLE`

☐	`.Partitioning`	Function handle default: `uq_SSE_[...]_varDiff`	Partitioning strategy according to Table 8 of UQLAB User Manual – Stochastic spectral embedding.
●	`.ExpDesign`	Structure default: `.Type = 'Sequential'`	Properties of the experimental design according to Table 18 of the UQLAB User Manual – Stochastic spectral embedding.
☐	`.[...]`		Additional options can be selected according to Table 1 of the UQLAB User Manual – Stochastic spectral embedding.

3.2 Accessing analysis results

Syntax

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

Output

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

Table 20: `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 23	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 21: `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`	Table 25	The post processed MCMC results as generated by `uq_postProcessInversionMCMC` (see Section 3.3.1).

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

Table 22: myBayesianAnalysis.Results (SLE)		
.SLE	PCE uq_model	PCE model of the likelihood function.
.PostProc	Table 28	The results obtained from post-processing the SLE with uq_postProcessInversioSLE (see Section 3.3.2).

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

Table 23: myBayesianAnalysis.Results (SSLE)		
.SSLE	SSE uq_model	SSE model of the likelihood function.
.PostProc	Table 30	The results obtained from post-processing the SSLE with uq_postProcessInversioSSLE (see Section 3.3.3).

3.3 Post-processing results

Syntax

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

Depending on the solver type, different post-processing functions are called, when the analysis is passed to uq_postProcessInversion.:

- Markov chain Monte carlo - Section 3.3.1
- Spectral likelihood expansions - Section 3.3.2
- Stochastic spectral likelihood embedding - Section 3.3.3

3.3.1 Markov chain Monte Carlo

Description

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

uq_postProcessInversionMCMC(myBayesianAnalysis, Name, Value) post-processes the results stored in myBayesianAnalysis.Restults.Sample using additional op-

tions specified by `Name, Value` pairs given in any order. These options are summarized in Table 24.

For a comprehensive discussion of the function logic, see also Section 2.4.1.6.

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

Table 24: `uq_postProcessInversionMCMC(..., 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 0.99	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: $[0.025; 0.975]$	Computes the specified percentiles using the supplied sample.
`'dependence'`	Logical default: true	if `true` the posterior covariance and correlation matrices are computed.
`'badChains'`	Integer array default: []	Removes the specified chains from the sample.
`'prior'`	Integer default: 1,000	Draws a specified number of sample points from the prior distribution.
`'priorPredictive'`	Integer default: 0	Draws a specified number of sample points from the prior predictive distribution, see also Eq. (1.27).
`'posteriorPredictive'`	Integer default: $\min(1{,}000, TC)$	Draws a specified number of sample points from the posterior predictive distribution, see also Eq. (1.28).
`'pointEstimate'`	String or Double array or Cell array default: `'Mean'`	Computes a point estimate.
	`'Mean'`	Computes the empirical mean from the supplied sample.
	`'MAP'`	Returns the point with the maximum posterior probability from the supplied sample.
	`'None'`	Removes possibly existing point estimate.
	Double array	Adds P custom estimators passed as $P \times M$ Double array.

	Cell array	Any combination of the above, except `'None'`, passed inside Cell array.
`'gelmanRubin'`	Logical default: false	if `true` the multivariate potential scale reduction factor $\hat{R}^p$ is computed (see Section 1.3.5.3).

Examples:

The command:

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

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

All post-processing results generated by the `uq_postProcessInversionMCMC` function are stored in the `myBayesianAnalysis.Results.PostProc` structure. The fields of this structure are listed in Table 25.

Table 25: `myBayesianAnalysis.Results.PostProc`

`.ChainsQuality`	Structure	Information about good and bad chains of the MCMC sample specified with the `'badChains'` argument of `uq_postProcessInversion`.
`.PostSample`	$T \times M \times C$ double	Posterior sample after post-processing with `uq_postProcessInversion` after removing bad chains and burn-in.
`.PriorSample`	$N_{\text{prior}} \times M$ double	Prior sample of size N_{prior}.
`.PriorPredSample`	Table 26	Prior predictive sample for each discrepancy group.
`.PostPredSample`	Table 26	Posterior predictive sample for each discrepancy group.
`.PostLogLikeliEval`	$T \times C$ double	Log likelihood evaluations at `.PostSample`.
`.PostModel`	N_{gr} structure array	Forward model evaluations at `.PostModel`.
`.PointEstimate`	Structure	Information related to the point estimate computed with `uq_postProcessInversionMCMC`.
`.Dependence`	Structure	Posterior dependence estimates like the covariance or correlation matrices.
`.Percentiles`	Structure	Percentiles of the posterior marginals.

`.MPSRF`	Structure	Multivariate potential scale reduction factor defined in Section 1.3.5.3.

The `.{Prior,Post}PredSample` field contains a structure array of size N_{gr}. The g-th structure in this array contains the predictive sample for the g-th discrepancy group. N_{pred} is the number of predictive samples:

Table 26: `myBayesianAnalysis.Results.PostProc.{Prior,Post}PredSample`

`.Sample`	$N_{\text{pred}} \times N_{\text{out},g}$	Predictive sample for g-th discrepancy group.
`.ModelEvaluations`	$N_{\text{pred}} \times N_{\text{out},g}$	Model evaluations corresponding to the predictive sample.
`.Discrepancy`	$N_{\text{pred}} \times N_{\text{out},g}$	The discrepancy value corresponding to the predictive sample.

3.3.2 Spectral likelihood expansions

Description

`uq_postProcessInversionSLE(myBayesianAnalysis)` post-processes the results stored in `myBayesianAnalysis.Results.SLE`. By default the evidence and the posterior mean are computed.

`uq_postProcessInversionSLE(myBayesianAnalysis, Name, Value)` post-processes the results stored in `myBayesianAnalysis.Results.SLE` using additional options specified by `Name, Value` pairs given in any order. These options are summarized in Table 29.

Table 27: `uq_postProcessInversionSLE(..., Name, Value)`

`'evidence'`	Logical default: true	Switch to compute the Bayesian evidence.
`'pointEstimate'`	String or Double array or Cell array default: `'Mean'`	Computes a point estimate.
	`'Mean'`	Computes the mean from the SLE.
	`'None'`	Removes possibly existing point estimate.
	Double array	Adds P custom estimators passed as $P \times M$ Double array.
	Cell array	Any combination of the above, except `'None'`, passed inside Cell array.
`'dependence'`	Logical default: false	if `true` the posterior covariance and correlation matrices are computed.

'parameters'	Integer array default: 1:M	Parameters to consider.

Examples:

The command:

```
uq_postProcessInversionSLE(myBayesianAnalysis,...
        'dependence', true, 'parameters', [1 3])
```

computes the posterior covariance for the input parameters x_1 and x_3.

All post-processing results generated by the uq_postProcessInversionSLE function are stored in the myBayesianAnalysis.Results.PostProc structure. The fields of this structure are listed in Table 28.

Table 28: myBayesianAnalysis.Results.PostProc

.Evidence	Double	The Bayesian evidence Z.
.PointEstimate	Structure	Information related to the point estimate computed with uq_postProcessInversionSLE.
.Dependence	Structure	Posterior dependence estimates like the covariance or correlation matrices.

3.3.3 Stochastic spectral likelihood embedding

Description

uq_postProcessInversionSSLE(myBayesianAnalysis) post-processes the results stored in myBayesianAnalysis.Results.SSLE. By default the evidence and the posterior mean are computed.

uq_postProcessInversionSSLE(myBayesianAnalysis, Name, Value) post-processes the results stored in myBayesianAnalysis.Results.SSLE using additional options specified by Name, Value pairs given in any order. These options are summarized in Table 29.

Table 29: uq_postProcessInversionSSLE(..., Name, Value)

'evidence'	Logical default: true	Switch to compute the Bayesian evidence.
'pointEstimate'	String or Double array or Cell array default: 'Mean'	Computes a point estimate.

	'Mean'	Computes the mean from the SSLE.
	'None'	Removes possibly existing point estimate.
	Double array	Adds P custom estimators passed as $P \times M$ Double array.
	Cell array	Any combination of the above, except 'None', passed inside Cell array.
'dependence'	Logical default: false	if true the posterior covariance and correlation matrices are computed.
'parameters'	Integer array default: 1:M	Parameters to consider.

Examples:

The command:

```
uq_postProcessInversionSSLE(myBayesianAnalysis,...
        'dependence', true, 'parameters', [1 3])
```

computes the posterior covariance for the input parameters x_1 and x_3.

All post-processing results generated by the uq_postProcessInversionSSLE function are stored in the myBayesianAnalysis.Results.PostProc structure. The fields of this structure are listed in Table 30.

Table 30: myBayesianAnalysis.Results.PostProc

.Evidence	Double	The Bayesian evidence Z.
.PointEstimate	Structure	Information related to the point estimate computed with uq_postProcessInversionSSLE.
.Dependence	Structure	Posterior dependence estimates like the covariance or correlation matrices.

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_postProcessInversionMCMC function, the post-processed sample is used.

3.4.1 Printing the results: uq_print

Syntax

```
uq_print(myBayesianAnalysis)
```

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)
```

Depending on the solver type, different display functions are called, when the analysis is passed to `uq_display`:

- Markov chain Monte carlo - Section 3.4.2.1
- Spectral likelihood expansions - Section 3.4.2.2
- Stochastic spectral likelihood embedding - Section 3.4.2.3

3.4.2.1 Markov chain Monte Carlo

Description

`uq_display_uq_inversion_MCMC(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_uq_inversion_MCMC(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 31.

Table 31: `uq_display_uq_inversion_MCMC(..., Name, Value)`

`'scatterplot'`	String or Integer default: `'all'` `'all'`	Scatterplots of the posterior and (if available) prior sample. Plots M dimensional scatterplots of the generated samples.

	Integer array	Plots the scatterplot with the specified parameters.
`'predDist'`	Logical	Requires initial call to `uq_postProcessInversionMCMC` to draw prior and/or posterior predictive sample points (see Eq. (1.27) and Eq. (1.28)). •if $N_{\text{out}} = 1$ displays *histogram plots* based on the sample points generated by `uq_postProcessInversionMCMC` and scatterplots of $\mathcal{Y}$ •if $N_{\text{out}} > 1$ displays *violin plots* based on the sample points generated by `uq_postProcessInversionMCMC` and scatterplots of $\mathcal{Y}$
`'trace'`	String or Integer default: `'none'`	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 default: `'none'`	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:

The command:

```
uq_display_uq_inversion_MCMC(myBayesianAnalysis, 'scatterplot', [1 3])
```

displays scatterplots of the first and third parameter x_1 and x_3.

3.4.2.2 Spectral likelihood expansions

Description

`uq_display_uq_inversion_SLE(myBayesianAnalysis)` create a visualization of the Bayesian analysis in object `myBayesianAnalysis`. By default a plot of the posterior density is created.

`uq_display_uq_inversion_SLE(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 32.

Table 32: `uq_display_uq_inversion_SLE(..., Name, Value)`

`'densityplot'`	String or Integer default: `'all'`	Prior and posterior density plots.
	`'all'`	Plots M dimensional density plots.
	Integer array	Plots the density plots with the specified parameters.
`'displaypce'`	Logical	Calls the PCE-specific display function with the constructed SLE object. See UQLAB User Manual – Polynomial Chaos Expansions for more information.

Examples:

The command:

```
uq_display_uq_inversion_SLE(myBayesianAnalysis, 'densityplot', [1 3])
```

displays density plots of the first and third parameter x_1 and x_3.

3.4.2.3 Stochastic spectral likelihood embedding

Description

`uq_display_uq_inversion_SSLE(myBayesianAnalysis)` create a visualization of the Bayesian analysis in object `myBayesianAnalysis`. By default a plot of the posterior density is created.

`uq_display_uq_inversion_SSLE(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 33.

Table 33: `uq_display_uq_inversion_SSLE(..., Name, Value)`

`'densityplot'`	String or Integer default: `'all'`	Prior and posterior density plots.
	`'all'`	Plots M dimensional density plots.
	Integer array	Plots the density plots with the specified parameters.
`'displaysse'`	Logical	Calls the SSE-specific display function with the constructed SLE object. See UQLAB User Manual – Stochastic spectral embedding for more information.

Examples:

The command:

```
uq_display_uq_inversion_SLE(myBayesianAnalysis, 'densityplot', [1 3])
```

displays density plots of the first and third parameter x_1 and x_3.

Acknowledgements

The updates to the `uq_postProcessInversionMCMC` function in UQLAB v1.4.0 were encouraged by Olaf Klein who pointed out some flaws with the function's logic. His contribution is gratefully acknowledged.

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. 14, 41

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. 17

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. 16, 17

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth (1987). Hybrid Monte Carlo. *Physics Letters B 195*(2), 216–222. 12

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. 11

Gelman, A. and D. B. Rubin (1992). Inference from iterative simulation using multiple sequences. *Statistical Science 7*(4), 457–472. 16

Goodman, J. and J. Weare (2010). Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science 5*(1), 65–80. 13, 14, 41

Haario, H., E. Saksman, and J. Tamminen (2001). An adaptive Metropolis algorithm. *Bernoulli 7*(2), 223–242. 11, 12

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. 9

Lüthen, N., S. Marelli, and B. Sudret (2020, September). Automatic selection of basis-adaptive sparse polynomial chaos expansions for engineering applications. 18

Marelli, S., P.-R. Wagner, C. Lataniotis, and B. Sudret (2021). Stochastic spectral embedding. *11*(2), 25–47. 19

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 (2016a). 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. 12

Nagel, J. B. and B. Sudret (2016b). Spectral likelihood expansions for Bayesian inference. *309*, 267–294. 17, 18

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. 12, 13

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. 9

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. 14

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. 48

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

Wagner, P.-R., S. Marelli, and B. Sudret (2021). Bayesian model inversion using stochastic spectral embedding. *436*, 110141. 19, 20, 22

Wand, M. and M. C. Jones (1995). *Kernel smoothing*. Chapman and Hall, Boca Raton. 15

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. 14, 41

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