JoDaKISS | Journal of Data- and Knowledge-integrated Simulation Science

BayesValidRox 2.0.0

Rebecca Kohlhaas[i] Maria Fernanda Morales Oreamuno[ii]
Farid Mohammadi[i] Alina Lacheim[i]

2025

Cite as

R. Kohlhaas, M. F. Morales Oreamuno, F. Mohammadi, and A. Lacheim, "BayesValidRox 2.0.0," *Journal of Data- and Knowledge-integrated Simulation Science*, vol. 1, 2025, DOI:10.46298/jodakiss.15337.

This document is licensed under the terms of the
Creative Commons Attribution 4.0 International License (CC BY 4.0).

[i]Institute for Modelling Hydraulic and Environmental Systems, LH2, University of Stuttgart
[ii]Institute for Modelling Hydraulic and Environmental Systems, LS3, University of Stuttgart

Abstract

BayesValidRox is an open-source Python package that provides methods for surrogate modeling, Bayesian inference and Bayesian multi-model comparison. Release 2.0.0 improves the modularity and uniformity of the package and introduces template classes for surrogate models and methods for generating posterior samples.

1 Background

Uncertainty quantification approaches, such as Bayesian inference, are important in assessing model reliability. In particular, Bayesian calibration, validation, and multi-model comparison provide a systematic approach to evaluating models under uncertainty. To support this, BayesValidRox offers an automated, modular software package for uncertainty-aware Bayesian calibration, validation, and multi-model comparison. However, these methods typically require a large number of model runs to reach statistical convergence, which can be unfeasible for computationally expensive models. To address this challenge, BayesValidRox integrates surrogate models, reducing computational costs while properly accounting for surrogate-induced uncertainties, making it well-suited for complex, high-fidelity simulations.

BayesValidRox has been in development since 2022 at the Institute for Modelling Hydraulic and Environmental Systems at the University of Stuttgart. Its first release, version 0.0.1, was launched in February 2022. The package received a major structure update with version 1.0.0 in February 2024, followed by a second major release, version 2.0.0, in February 2025. Since its initial release, the package has been used in multiple publications [5, 6].

2 Specification

Subject	Computational engineering
Specific subject area	Methods for uncertainty quantification, surrogate modeling, Bayesian model analysis and Bayesian multi-model comparison.
Programming language	Python
Software repository	git.iws.uni-stuttgart.de/inversemodeling/bayesvalidrox
Website	pages.iws.uni-stuttgart.de/inversemodeling/bayesvalidrox/
Documentation	pages.iws.uni-stuttgart.de/inversemodeling/bayesvalidrox/
Related research article	None

3 Value

- BayesValidRox provides an automated workflow for Bayesian calibration, validation and multi-model comparison for computationally expensive models. This is achieved by substituting expensive high-fidelity models with cheaper-to-evaluate surrogate models. The framework takes into account measurement uncertainty, as well as surrogate-induced errors.
- BayesValidRox is designed with a modular structure, especially for surrogate training and evaluation. This simplifies switching between surrogate types in a workflow and allows for the application of techniques such as Principal Component Analysis or sequential training on arbitrary surrogates. Similarly, surrogate models that are not yet implemented in BayesValidRox can be coupled to the framework with the help of a metamodel template class.
- BayesValidRox allows for integration with external and Python-based models via a PyLink class, which uses wrapper functions and file executables to run any base model. This makes the framework adaptable to a wide range of computational applications.
- BayesValidRox offers a broad collection of methods for sequential training of surrogate models, including both variance-based [1] and Bayesian approaches [8].
- Choosing the Engine class as the central focus point of BayesValidRox allows for the application of Bayesian methods for validation, inference and model comparison both with a surrogate model and the original model.
- BayesValidRox is implemented in Python 3.10 with an object-oriented structure. Its availability as a Python package via pip makes it easy to install. The surrogate training and multiple parts of the Bayesian analysis support parallelization.

4 Code Structure

BayesValidRox is developed in a GitLab repository and follows a corresponding state-of-the-art layout for open-source projects in terms of source code and supplementary files, such as licensing information, contribution guidelines, etc. In terms of code, the most relevant folders are:

- `src\bayesvalidrox`: All source files for the package.
- `tests`: Unit tests used in GitLab's continuous integration.
- `examples`: Well documented example applications.

More details on the folder structure can be found in the online documentation at https://pages.iws.uni-stuttgart.de/inversemodeling/bayesvalidrox/packagedescription.html.

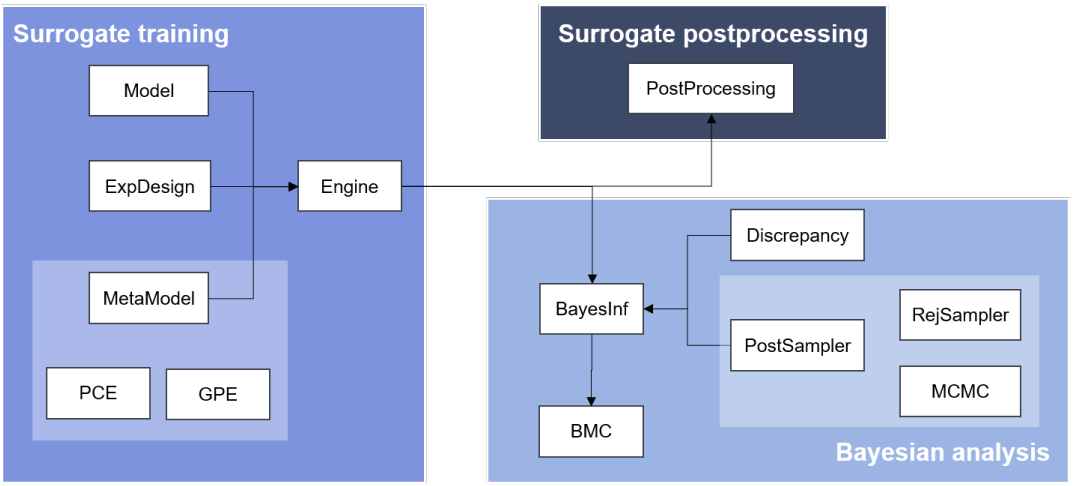

Figure 1: Workflow for BayesValidRox

5 Design and Methodology

This section walks through the expected workflow for BayesValidRox in a general manner. The user guide on the documentation website gives more detailed information on the specific class structure that is relevant to each section, as well as the settings and options available for each class.

A schematic representation of the workflow for BayesValidRox is given in Figure 1. We split the capabilities of BayesValidRox into three categories, surrogate construction, surrogate post-processing and Bayesian analysis.

Surrogate construction refers more specifically to the construction of an `Engine` object from a model interface, description of the experimental design and surrogate model object. The `Engine` class acts as a manager to train and evaluate the chosen surrogate. If no surrogate is given, the full-fidelity model is run directly. The training data for the surrogate can be either given by the user or generated from the defined input parameter space via the class `ExpDesign`, with subsequent model evaluations. The training framework is written to support arbitrary surrogate models of a structure, which is described by a `MetaModel` template class. BayesValidRox includes extensive and well-tested implementations of both the Polynomial Chaos Emulator (PCE) [12], as well as the arbitrary PCE [7] with a broad variety of Bayesian, sparse and non-sparse training options, and a Scikit-Learn [9] based implementation of Gaussian Process Emulators (GPE) [13].

Sequential training is available with multiple options, split into methods for exploration- and exploitation-based strategies for selecting additional training samples. Variance-based approaches [10] can be applied for surrogate models that provide approximation uncertainty (typically assumed Gaussian) along with their predictions. Bayesian active learning [8] strategies can be used to refine the surrogate model for inference purposes when a reference observation is available .

BayesValidRox provides post-processing tools for the trained surrogate models, allowing for surrogate perfomance evaluation. If validation data is available, different criteria, including moment calculation, are available to assess model predictive quality. Additionally, for PCE-based surrogates, sensitivity analysis can be performed via the calculation of Sobol indices from the polynomial coefficients.

The third module of BayesValidRox is its collection of tools for Bayesian model analysis and comparison. These methods can be applied to any `Engine` object, both with a trained surrogate model or with the original model directly, incorporating available reference observations, such as measurement data. Measurement uncertainty can be included in the analysis with the help of the `Discrepancy` class.

The `BayesInf` class supports Bayesian parameter inference and Bayesian validation via the calculation of the Bayesian Model Evidence (BME). For inference, posterior samples can be generated with either rejection sampling [3] or Markov-Chain Monte Carlo (MCMC) sampling. If multiple competing models are given, Bayesian multi-model comparison can be performed with the class `BMC` via pairwise comparison of the model BMEs, the calculation of model weights or the generation of a confusion matrix [2, 11, 4, 5]. For a detailed description of the methods we refer to the work in [5].

6 Limitations

The limitations described here are specific to the release 2.0.0 of BayesValidRox. As the package originates from a research group specializing in PCE-based surrogates, the sensitivity analysis (Sobol indices) and some options for sequential training are specifically tailored for PCE surrogate models and are not yet applicable to arbitrary surrogate models, including the implemented Gaussian Process Emulators (GPEs). Moreover, the methods for Bayesian analysis do not support user-defined likelihood functions but use a pre-defined Gaussian likelihood. Furthermore, measurement uncertainty is assumed to follow a Gaussian distribution, which aligns with common probabilistic modeling practices.

7 Technical Validation

BayesValidRox uses GitLab's continuous integration framework for unit testing of the implemented features. Pipelines are run after every commit and before any merge. For release 2.0.0, 280 tests were run in each pipeline, resulting in a test coverage of the source code of about 50%.

CC BY

8 Usage Notes

BayesValidRox can be installed as a Python package via pip and runs on Windows, Linux, and Mac operating systems. Release 2.0.0 is built on Python 3.10.

Installation instructions are given on both the repository and the website. The website further provides a detailed walk-through of one example for building, training and evaluating a surrogate model, as well as performing Bayesian inference. The user guide section on the website gives insight into the overall code structure and introduces each section of the framework with UML diagrams and brief code examples.

Acknowledgment

The development of BayesValidRox is primarily funded by the Collaborative Research Centre SFB 1313, Project Number 327154368. Additional funding was received from the Bundesgesellschaft für Entlagerung as part of the project URS and the Cluster of Excellence "Data-Integrated Simulation Science" (EXC 2075 – 390740016).

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Beck and S. Guillas. Sequential design with mutual information for computer experiments (MICE): Emulation of a tsunami model. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):739–766, 2016. doi:10.1137/140989613.

[2] D. Draper. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):45–97, 1995. doi:10.1111/j.2517-6161.1995.tb02015.x.

[3] J. M. Hammersley. Monte Carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86(3):844–874, 1960. doi:10.1111/j.1749-6632.1960.tb42846.x.

[4] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: a tutorial. *Statistical Science*, 14(4):382–401, 1999. doi:10.1214/ss/1009212519.

[5] F. Mohammadi. *A surrogate-assisted Bayesian framework for uncertainty-aware validation benchmarks*. PhD thesis, University of Stuttgart, Stuttgart, Germany, 2023. doi:10.18419/opus-15402.

[6] F. Mohammadi, E. Eggenweiler, B. Flemisch, S. Oladyshkin, I. Rybak, M. Schneider, and K. Weishaupt. A surrogate-assisted uncertainty-aware bayesian validation framework and its application to coupling free flow and porous-medium flow. *Computational Geosciences*, 27(4):663–686, 2023. doi:10.1007/s10596-023-10228-z.

[7] S. Oladyshkin and W. Nowak. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering & System Safety*, 106:179–190, 2012. doi:10.1016/j.ress.2012.05.002.

[8] S. Oladyshkin, F. Mohammadi, I. Kroeker, and W. Nowak. Bayesian3 active learning for the gaussian process emulator using information theory. *Entropy*, 22(8), 2020. doi:10.3390/e22080890.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[10] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409 – 423, 1989. doi:10.1214/ss/1177012413.

[11] A. Schöniger, W. A. Illman, T. Wöhling, and W. Nowak. Finding the right balance between groundwater model complexity and experimental effort via Bayesian model selection. *Journal of Hydrology*, 531:96–110, 2015. doi:10.1016/j.jhydrol.2015.07.047.

[12] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4): 897–936, 1938. doi:10.2307/2371268.

[13] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.