Proc. of the European Safety and Reliability Conference (ESREL 2026)
Edited by José C. Matos, Paulo B. Lourenço, Michael Beer, Daniel Oliveira,
Edoardo Patelli, Hélder S. Sousa, Rui A. Silva and Monica Santamaria
 Published by Research Publishing, Singapore.
doi: 10.3850/ESREL2026061419_esrel26-p26140-cd

A Python library for stochastic model updating with black-box models using `PyUncertainNumber` - a case study with the NASA UQ challenge

Yu Chen

Department of Mechanical and Aerospace Engineering, University of Liverpool, UK.
E-mail: yuchen2@liverpool.ac.uk

Roberto Rocchetta, Lorenzo Nespoli, Vasco Medici

Università della Svizzera Italiana, SUPSI, Mendrisio, CH.
E-mail: [roberto.rocchetta],[lorenzo.nespoli],[vasco.medici]@supsi.ch

Marco de Angelis, Dawid Ochnio, Edoardo Patelli

University of Strathclyde, Glasgow, UK. E-mail: [marco.de-angelis],[dawid.ochnio.2020],[edoardo.patelli]

Model updating plays a vital part in enhancing the credibility of computational models, especially for safety-critical engineering applications subject to polymorphic uncertainties. A long-standing objective in engineering is to reduce discrepancies arising from various sources of uncertainties and approximations between the simulation model and the corresponding physical systems. This paper presents both likelihood-based and likelihood-free methods for stochastic model updating using `PyUncertainNumber`, with the particular focus on reducing epistemic uncertainty. The developed library allows for a user-friendly calling signature and features scalability with black-box models. To demonstrate its efficacy and applicability, these methods are applied to the 2025 NASA and DNV Challenge on uncertainty quantification.

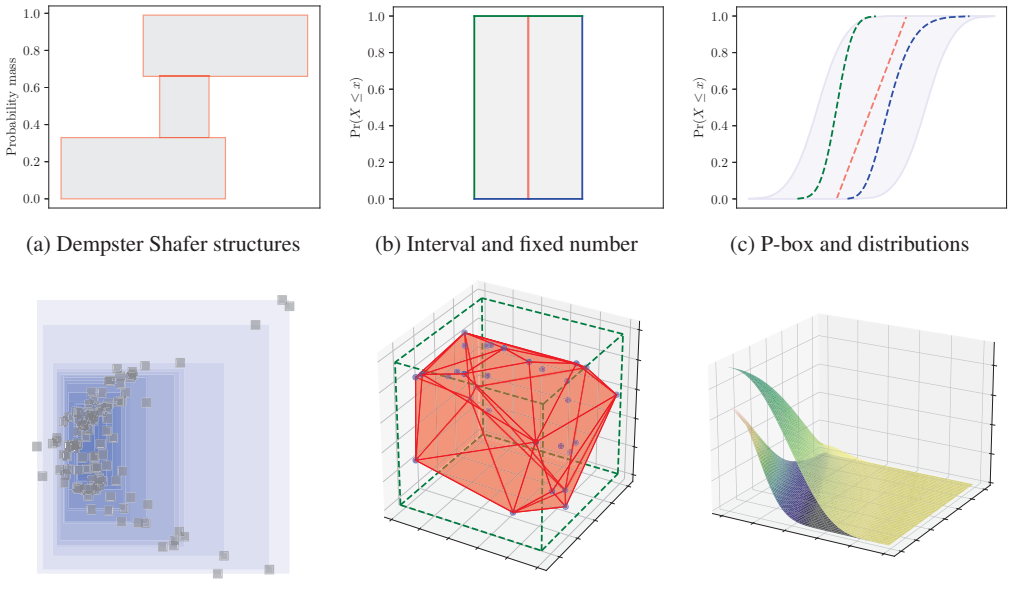
Keywords: stochastic model updating, uncertain numbers, black box model, probability box, uncertainty quantification.

1. Introduction

Mathematical models are often proposed in scientific computations to simulate the behaviour of complex engineered or natural systems, typically under unknown external (operational or environmental) conditions (Oberkampf et al., 2004; Gray et al., 2022). This leads to a long-standing goal of improving the mathematical model to better reflect the physical system under limited observations (Rocchetta et al., 2018; Chen et al., 2026). Model updating, also termed as *model calibration*, represents the process of adjusting physical or non-physical parameters in the computational model to improve agreement with experimental results (Bi et al., 2023).

However the computational environment is often surrounded by a variety forms of uncertainty. What compounded the situation is that empirical information, whether expert knowledge or empirical data, are commonly sparse (Chen et al., 2023). As a result, additional experiments may be required to reduce the epistemic uncertainty in the computational workflow. It therefore comes the pursuit to reliably utilise the additional experiments, which are often expensive, to update the model setting in the face of uncertainty, such that model simulations agree better with observations.

With the increasing recognition of the differentiation of *aleatory* and *epistemic* uncertainty, the engineering community has increasingly realised the significance to appropriately address different kinds of uncertainty in safety-critical systems, especially when quantitative data is either very sparse or prohibitively expensive to collect (Ching and Chen, 2007; Ching and Beck, 2004; Bi et al., 2019). Abundant advancements have been made in the subject of *stochastic model updating*, see Bi et al. (2023) for a comprehensive review. This research goal has also been echoed in the NASA challenges (Gray et al., 2022) in recent years to unite a community of computational engineer-



(a) Dempster Shafer structures (b) Interval and fixed number (c) P-box and distributions

(d) Nested DSS (e) Convex hull and hyperrectangle (f) Bivariate p-box

Fig. 1.: Illustration of uncertainty models

ing and physics towards improved credibility of model simulations, decisions and designs.

Given its importance, an accessible toolkit with the state-of-the-art capability of non-deterministic model updating is beneficial in facilitating the enhancement of model credibility for real-world engineering applications. As such, this paper presents `PyUncertainNumber`, a Python package that provides both likelihood-based and likelihood-free approaches for model updating, with the particular focus on reducing epistemic uncertainty.

2. Uncertainty models to appropriately characterise polymorphic uncertainty

Complex systems are subject to polymorphic uncertainty, requiring appropriate representation and characterisation of the various forms of uncertainty. For instance, aleatory variables are modelled by probability distributions, while epistemic uncertainty is represented by intervals or other bounded sets which does not specify likelihood or belief on any values within. Particularly, Ferson and Hajagos (2004) introduced the concept of *uncertain numbers* which stand for a generalised representation that unifies several uncertainty constructs including intervals, probability distributions, probability boxes (p-boxes) and Dempster-Shafer structures (DSS), plus real numbers. This framework enables expressive uncertainty representations at different imprecision through a unified construct.

The Python library `PyUncertainNumber` serves as a user-friendly toolkit that implements the framework of uncertain numbers, allowing for a closed computation ecosystem whereby trustworthy computations, whether intrusively or non-intrusively, can be conducted (Chen and Ferson, 2025).

Conveniently, this library has native supports for a wide spectrum of uncertainty models (see Fig. 1). The main idea is for the uncertainty models to appropriately represent uncertainties depending on the available empirical information without introducing unjustified assumptions. These instances of uncertainty models can be explicitly constructed based on characterising empirical information or implicitly created during the

model calibration process.

Fig. 1 displays several examples of uncertainty models supported in `PyUncertainNumber`, and the code block below illustrates how different uncertainty models can be easily constructed using the `pba` API (Application Programming Interface).

```
from pyuncertainnumber import pba

# interval vectors are also supported
i = pba.I(1, 2)

# distribution instances can serve as
    prior distributions for calibration
d = pba.D("gaussian", (6, 2))

# degenerate pbox will be a distribution
p = pba.normal([2, 3], 0.5)

# where intervals are to be specified,
    interval objects can also be used
dss = pba.DSS(
    intervals=[[1, 5], [3, 6]],
    masses=(0.5, 0.5)
    )
```

3. Non-deterministic calibration of black-box models

3.1. *Likelihood-based: Transitional Markov Chain Monte Carlo (TMCMC) technique*

Bayesian model updating plays a significant role in *stochastic model updating* based on its established posterior updating mechanism. In cases of intractable likelihood, ABC (Approximate Bayesian Computation) procedures are proposed to bypass the evaluation of likelihood (Robert, 2016), and pseudo-likelihood functions have been proposed based on various stochastic distances between experimental measurements and model simulations, see Bi et al. (2023) for a review.

Notably, TMCMC algorithm (Ching and Chen, 2007) has gained popularity in recent years due to its versatility for high-dimensional, multimodal, highly peaked PDFs. In its core entails an iterative procedure of weighting, resampling, and MCMC perturbation in a sequence of stages, guided by a tempering parameter β_j, to gradually approximate the target posterior (Lye et al., 2021; Bi et al., 2023), as shown below:

$$P^j \propto P(\mathcal{D}|\boldsymbol{\theta})^{\beta_j} \cdot P(\boldsymbol{\theta}) \tag{1}$$

where P^j denotes the tempered posterior at iteration stage j, $P(\mathcal{D}|\boldsymbol{\theta})$ and $P(\boldsymbol{\theta})$ represent the likelihood and prior distribution respectively. As shown below, `PyUncertainNumber` provides an accessible API which sketches out the main components of the TMCMC algorithm. Conveniently, prior distributions are indeed the `Distribution` instances constructed using the `pba` API.

```
# exemplar prior distributions
parameters = [
  pba.D("uniform", (0.8, 2.2)),
  pba.D("uniform", (0.4, 1.2))
  ]

t = TMCMC(
  N,                # number of particles
  parameters,       # prior distributions
  names,            # name tags
  log_likelihood  # callable, closed-form
     or approximate likelihood
  mutation_step,  # maximum mutation step
)

trace = t.run()    # return the trace
```

3.2. *Likelihood-free: K-Nearest Neighbors filtering approach*

K-Nearest Neighbors (KNN) filtering is a novel data-driven calibration method based on a non-parametric kernel average smoother (Hastie et al., 2001), whereby posterior parameter samples $\mathcal{D}_\xi^{KNN}$ corresponding to a certain design parameter ξ can be filtered using some certain norm $\|\cdot\|$ as the discrepancy measure $\delta(\theta, \xi)$, which reads:

$$\mathcal{D}_\xi^{KNN} \leftarrow \arg\min_{\theta \in \mathcal{D}^{\text{sim}}} \|\delta(\theta, \xi)\| \tag{2}$$

where $\delta(\theta, \xi)$ denotes a discrepancy function based on some salient features conditioned on a certain design ξ. $\mathcal{D}^{\text{sim}}$ denotes a simulation dataset obtained from exploration of the input space.

Intuitively, it could be conceptually seen as a dictionary mapping between parameters and data. Given empirical observation, the most likely parameters, as if nearest neighbours to the ground truth, are found through the defined norm. For this

to work, it is typically needed for the parameter space to be explored first by, for instance, uniform sampling, followed by the KNN filtration process as below:

```
from pyuncertainnumber.calibration import
     KNNCalibrator

k = KNNCalibrator(
  knn=500,        # number of neighbors
  evaluate_model=True
  )

k.setup(
  model,          # simulation model
  theta_sampler,  # callable, sampler
  xi_list,        # list of designs
  n_samples       # number of particles
  )

# run calibration
post_single = k.calibrate(
    observations,   # empirical data
    combine="stack" # pooled kNN
    )

# posterior samples
theta_post_single = post_single["theta"]
```

3.3. *Epistemic filtering approach*

Epistemic filtering is a straight. forward filtering-based method that constructs the level sets for epistemic parameters modelled by some bounded sets with a discrepancy measure δ. It is defined as below where $x_e \in \mathbb{R}^{n_e}$ denotes a normalised epistemic domain of dimension n_e.

$$E_\alpha = \{x_e \in [0,1] : \delta(x_e) \leq \alpha\} \tag{3}$$

E_α can be approximated by a convex hull (see Fig. 1e), and choices of δ include common norms such as infinity norm $\|x\|_\infty$ or Euclidean norm $\|x\|_2$. This approach is also referred to as "iterative filtering" due to the fact that once α is given as a vector, the algorithm will iterate over each element, and implicitly conduct resampling in order to ensure a thorough exploration of the epistemic space.

```
from pyuncertainnumber.calibration import
     EpistemicFilter

ef = EpistemicFilter(
    xe_samples,         # Epistemic
    samples to be filtered
    discrepancy_scores # associated
    discrepancy values
)

filtered_xe, hull, low_bound, upper_bound
     = ef.filter(threshold)
# a tuple of returns including the
    filtered epistemic samples, the
    convex hull, plus the hyperrectangle
    bounds
```

3.4. *Data peeling approach*

The data peeling algorithm (de Angelis et al., 2023) constructs a hierarchy of nested enclosing regions, which serve as calibrated consonant Dempster-Shafer structures whose basic beliefs are set equal to the scenario lower probabilities. The algorithm solves an optimisation problem to determine the smallest possible region or set that contains all data points, see Fig. 1d. The points that lie on the boundary and mathematically define this minimal region are treated as *support vectors*.

Once identified, these boundary points are iteratively removed from the dataset (*peeling*), while the process is repeated on the reduced set, producing progressively smaller, mutually nested layers. The final result is a full sequence of nested level sets that encode the depth structure of the data. Posterior samples are finally generated from a mixture distribution within the credal set identified by the calibrated consonant structure.

```
from pyuncertainnumber.calibration import
     data_peeling as dp

# X is the problem-specific dataset
a, b = dp.data_peeling_algorithm(
    X,
    tol=0.01)
# a: a list of subindices corresponding
    to the support vectors
# b: a list of enclosing sets (boxes by
    default)

f, p = dp.peeling_to_structure(
    a,
    b,
    kind='scenario',
    beta=0.01)
# f: a structure containing projections
# p: a list of lower probability, one for
     each level
```

4. Application example: NASA UQ challenge 2025

The NASA and DNV Challenge on Optimization under Uncertainty considers a generic physical system whose state is described by an input vector $\boldsymbol{x}$ and a corresponding time-varying response $\boldsymbol{Y} \in \mathbb{R}^{T \times n_y}$. The input vector is consisted of parameters of various nature $\boldsymbol{x} = (\boldsymbol{x_a}, \boldsymbol{x}_e, \boldsymbol{x}_c, \omega)$ where $\boldsymbol{x_a} \sim f(a)$ denotes aleatory variables with unknown distribution, $\boldsymbol{x}_e \in E$ denotes epistemic variables that have fixed-but-unknown values, and modelled by some bounded set. $\boldsymbol{x}_c$ and ω are design variables are random seed respectively, see (Agrell et al., 2024) for additional details about the dimensions of the problem statement.

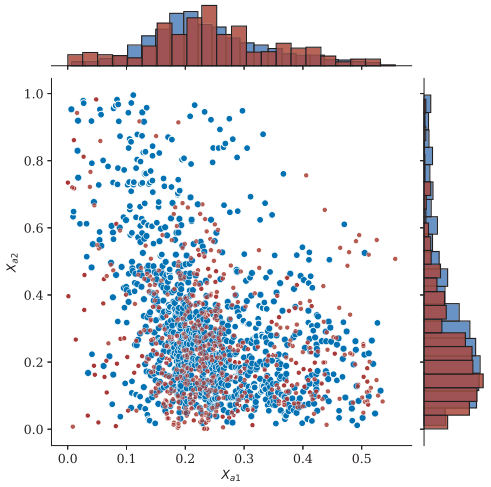


Fig. 2.: Comparison of the posterior X_a samples between KNN (brown) and data peeling (blue) techniques

A black-box model is provided for performing simulations. For the purpose of this paper, we present results regarding merely to the first part of the challenge problem: calibration for the above-mentioned uncertain parameters given empirical observations. Specifically, the results in this section are on the basis of merely the base design provided by the challenge, i.e. $\boldsymbol{x}_c = (0.533, 0.666, 0.5)$. Readers are referred to (Rocchetta et al., 2025) for a comprehensive treatment of the challenge, which further explores solutions as to optimisation tasks and data querying strategies.

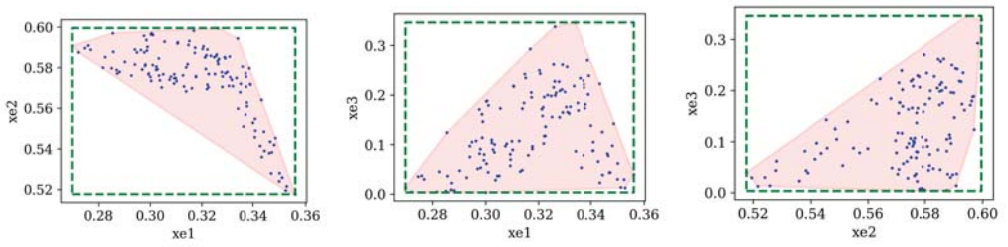
In this study, both parametric and non-parametric calibration assessments are conducted using the methods in the last section. As observed in Rocchetta et al. (2025), the ratios $r_{ij}^{I_2} = y_i/y_j, i, j \in I_2$ between the responses in $I_2 = \{y_4, y_5, y_6\}$ is invariant with the aleatory space $(\boldsymbol{x}_a, \omega)$ but only changes with $\boldsymbol{x}_e$ given a fixed design $\boldsymbol{x}_c$. This provides salient features amenable to the calibration techniques.

Fig. 2 shows the posterior samples for of X_a for the KNN and data peeling, which are non-parametric approaches. It should be noted that data peeling in fact yields imprecise structures and hence these posterior particles are sampled uniformly in the focal elements.

In comparison, in a parametric manner, a second-order distribution for $\boldsymbol{x}_a \in \mathbb{R}^2$, parameterised by Beta marginals $X_{ai} \sim \mathcal{B}(\alpha_i, \beta_i)$ and a Gaussian copula ρ, is employed to characterize the aleatory variables. This therefore leads to an enriched epistemic space besides the original epistemic variables, i.e. $E = \{\alpha_1, \beta_1, \alpha_2, \beta_2, \rho, X_{e1}, X_{e2}, X_{e3}\}$. A multivariate imprecise structure (e.g. multivariate p-box) can be obtained by expressing the epistemic uncertainty in intervals. Fig. 4 shows the marginal p-box for X_{a1} which clearly manifests a reduction of the epistemic uncertainty in E between the prior and the posterior after model updating on the basis of empirical data associated with the base design.

Fig. 5 further shows the joint cumulative distribution function (CDF) of the bivariate p-box X_a where the upper and lower surface are respectively depicted in green and in blue.

Filtered epistemic space of $\boldsymbol{x}_e$ using the feature ratios can be seen in Fig. 3, in which both the convex hull and the enveloping hyperrectangle intervals are shown. It is apparent that a significant reduction has been accomplished from the original unit space $\boldsymbol{x}_e \sim [0, 1]^3$.



(a) X_{e_1} vs. X_{e_2} (b) X_{e_1} vs. X_{e_3} (c) X_{e_2} vs. X_{e_3}

Fig. 3.: Iterative filtering on epistemic variables $\boldsymbol{x}_e$

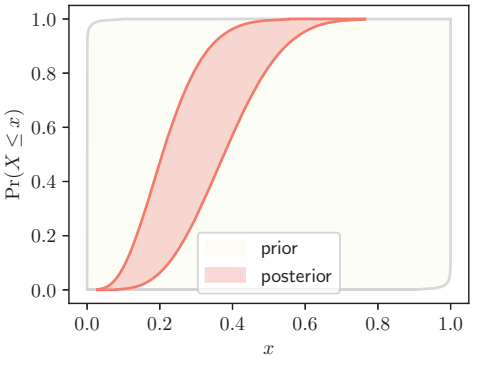


Fig. 4.: Prior and posterior p-box for the imprecise Beta uncertainty model X_{a1}

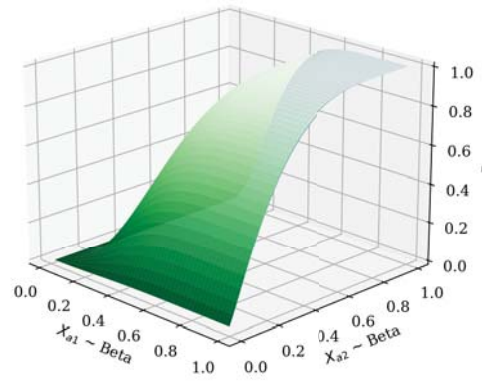


Fig. 5.: Joint cumulative distribution function for the imprecise bivariate p-box $\boldsymbol{x}_a$

5. Conclusion

The developed Python library facilitates trustworthy management of uncertainty through faithful uncertainty representations.

Unlike other established Python-based general purpose uncertainty quantification (UQ) tools that primarily focus on probabilistic modelling, `PyUncertainNumber`, which is underpinned on the framework of *uncertain numbers*, however, relaxes such precision and is capable of explicitly addresses both aleatory and epistemic uncertainty using appropriate uncertainty models without making unjustified assumptions.

This study illustrates its capability of *stochastic model updating* using both likelihood-based and likelihood-free techniques, making it a robust framework for addressing real-world UQ challenges subject to polymorphic uncertainties. These techniques are demonstrated with the NASA UQ challenge and the results suggest great performance in reducing epistemic uncertainty and great applicability in handling black-box models.

6. Reproducibility

The source code for calibration capability is in the Python library `PyUncertainNumber` whose GitHub repository can be accessed using `https://github.com/leslieDLcy/PyUncertainNumber`. Illustrative hands-on examples, and additional details regarding the use the calibration methods are provided in the documentation `https://pyuncertainnumber.readthedocs.io/en/latest/examples/index.html`.

7. Future work and potential limitations

The framework of uncertain numbers allows for a closed computation ecosystem whereby trustworthy computations can be conducted in a rigorous manner. As such, it is planned for `PyUncertainNumber` to enable capabilities across the typical uncertainty analysis pipeline, encompassing uncertainty characterisation, aggregation, propagation, model updating, and applications including reliability analysis and optimisation under uncertainty. Note that the APIs may undergone slight changes in the upcoming versions as the package is under active development. The version for this paper is `V0.1.15`.

This paper is focused on the implementation of state-of-the-art stochastic calibration methodologies. Readers are encouraged to consult the relevant academic literature for further details on the potential limitations of each respective method.

Acknowledgement

The work leading to these results received funding through the UK project Development of Advanced Wing Solutions 2 (DAWS2). The DAWS2 project is supported by the Aerospace Technology Institute (ATI) Programme, a joint government and industry investment to maintain and grow the UK's competitive position in civil aerospace design and manufacture. The programme, delivered through a partnership between ATI, Department for Business and Trade (DBT) and Innovate UK, addresses technology, capability and supply chain challenges.

References

Agrell, C., L. G. Crespo, V. Flovik, E. Vanem, and S. Kenny (2024). The nasa and dnv challenge on optimization under uncertainty.

Bi, S., M. Beer, S. Cogan, and J. Mottershead (2023). Stochastic model updating with uncertainty quantification: an overview and tutorial. *Mechanical Systems and Signal Processing 204*, 110784.

Bi, S., M. Broggi, and M. Beer (2019). The role of the bhattacharyya distance in stochastic model updating. *Mechanical Systems and Signal Processing 117*, 437–452.

Chen, Y. and S. Ferson (2025). Imprecise uncertainty management with uncertain numbers to facilitate trustworthy computations. *Python in Science Conference, 2025*.

Chen, Y., I. Ioannou, and S. Ferson (2026). Efficient interval-based uncertainty quantification for model validation and predictive capability. In *AIAA SCITECH 2026 Forum*, pp. 0296.

Chen, Y., E. Patelli, B. Edwards, and M. Beer (2023). A bayesian augmented-learning framework for spectral uncertainty quantification of incomplete records of stochastic processes. *Mechanical Systems and Signal Processing*.

Ching, J. and Y.-C. Chen (2007). Transitional markov chain monte carlo method for bayesian model updating, model class selection, and model averaging. *Journal of engineering mechanics 133*(7), 816–832.

Ching, J.-y. and J. L. Beck (2004). Bayesian analysis of the phase ii iasc–asce structural health monitoring experimental benchmark data. *Journal of Engineering Mechanics 130*(10), 1233–1244.

de Angelis, M., A. Gray, S. Ferson, and E. Patelli (2023). Robust online updating of a digital twin with imprecise probability. *Mechanical Systems and Signal Processing 186*, 109877.

Ferson, S. and J. G. Hajagos (2004). Arithmetic with uncertain numbers: rigorous and (often) best possible answers. *Reliability Engineering & System Safety 85*(1-3), 135–152.

Gray, A., A. Wimbush, M. de Angelis, P. O. Hristov, D. Calleja, E. Miralles-Dolz, and R. Rocchetta (2022). From inference to design: A comprehensive framework for uncertainty quantification in engineering with limited information. *Mechanical Systems and Signal Processing 165*, 108210.

Hastie, T., J. Friedman, and R. Tibshirani (2001). Kernel methods. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pp. 165–192. Springer.

Lye, A., A. Cicirello, and E. Patelli (2021). Sampling methods for solving bayesian model updating problems: A tutorial. *Mechanical Systems and Signal Processing 159*, 107760.

Oberkampf, W. L., T. G. Trucano, and C. Hirsch (2004). Verification, validation, and predictive capability in computational engineering and

physics. *Appl. Mech. Rev. 57*(5), 345–384.

Robert, C. P. (2016). Approximate bayesian computation: a survey on recent results. In *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*, pp. 185–205. Springer.

Rocchetta, R., M. Broggi, and E. Patelli (2018). Do we have enough data? robust reliability via uncertainty quantification. *Applied Mathematical Modelling 54*, 710–721.

Rocchetta, R., L. Nespoli, V. Medici, Y. Chen, M. de Angelis, D. Ochnio, E. Patelli, and E. Smith (2025). An integrated uncertainty quantification and optimization for solving the 2025 nasa-dnv challenge. In *Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference*, Stavanger, Norway. Research Publishing, Singapore.