Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
©2025 ESREL SRA-E 2025 Organizers. *Published by* Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P7698-cd

Uncertainty-Aware Optimization in Engineered Systems via Gradient Boosting and Differential Evolution

Dibakar Roy Sarkar

Civil and Systems Engineering, Johns Hopkins University, USA. E-mail: droysar1@jhu.edu

Sukanta Basu

Atmospheric Sciences Research Center, University at Albany, U.S.A. E-mail: sbasu@albany.edu

Lance Manuel

Civil, Architectural, and Environmental Engineering, The University of Texas at Austin, U.S.A.
E-mail: lmanuel@mail.utexas.edu

Somdatta Goswami

Civil and Systems Engineering, Johns Hopkins University, USA. E-mail: sgoswam4@jhu.edu

In engineered systems where safety and performance objectives are critical, accounting for various uncertainties-aleatoric and epistemic-is essential. Data sets offering useful information for decision-making can be expensive to collect and, hence, sparse. Distinction between all the underlying stochasticity through efficient data-driven approaches must be informed by selective strategies. In this work, we document our efforts in addressing a set of challenge problems posed by DNV and NASA personnel that require identifying the character and structure of the underlying stochasticity and then addressing reliability, performance, and optimization under uncertainty Agrell et al. (2025). We employ LightGBM for regression tasks, FLAML for hyperparameter tuning, and Differential Evolution for robust optimization using limited test data and simulation results on key response time series.

Keywords: machine learning, uncertainty quantification, feature engineering, optimization.

1. Introduction

Engineered systems in critical domains require robust performance despite significant uncertainties. These uncertainties—both aleatory and epistemic — challenge design optimization and reliability assessment, especially when data are sparse. This paper integrates machine learning (ML) with uncertainty quantification (UQ) and optimization to address NASA and DNV UQ challenge problems Agrell et al. (2025). Our approach characterizes stochasticity, optimizes performance, and ensures reliability using differential evolution and constraint-based methods. Through regression models and adaptive sampling, we demonstrate how limited data can effectively inform engineering decisions for systems operating under uncertainty.

2. Strategy and Solution for UQ Problem #1.1

In order to address the UQ challenge problem Agrell et al. (2025) as posed, we have reformulated it as a tabular regression task. In the scientific literature and ML competitions, decision tree-based ML models are commonly employed for tackling tabular regression problems, leveraging a variety of statistical strategies. For example, Random Forest (Breiman, 2001) and Extremely Randomized Trees (Geurts et al., 2006) rely on a bagging or bootstrap aggregating approach, whereas XGBoost (Chen and Guestrin, 2016; Wade, 2020) and LightGBM (Machado et al., 2019) use gradient boosting (Freund and Schapire, 1999; Friedman, 2002). As reported in this paper, our numerical experiments utilize the Light Gradient Boosting Machine (LightGBM) model.

Our proposed ML approach is summarized in

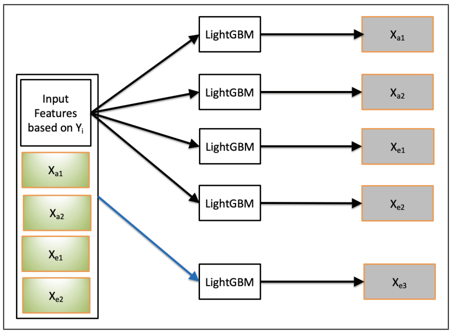

Fig. 1. Proposed ML framework to solve UQ problem #1.

Fig. 1. For a selected combination of control parameters (say, X_{c1} = 0.533, X_{c2} = 0.666, X_{c3} = 0.500), we generate training data of size N using the computational model provided by the challenge organizers. The aleatoric parameters (X_{a1}, X_{a2}) and epistemic parameters (X_{e1}, X_{e2}, X_{e3}) are randomly drawn from uniform distributions while the seed variable s assumes random integer values. The resulting training set contains N samples of multivariate response time series, $Y_j^{(k)}$, where j ranges from 1 to 6 and k varies from 1 to N. We use $N = 10^6$.

From each response time series (of length 60), we extract more than one hundred input features (including moments, Hurst exponents, etc.). These features are then used to predict the X_a and X_e parameters using the LightGBM model. In essence, we employ LightGBM to "reverse engineer" the underlying parameters of the computational model based on the simulated response time series. Note that to estimate X_{e3}, we also include estimated values of X_{a1}, X_{a2}, X_{e1}, and X_{e2} as additional input features.

Technical details on the proposed ML approach are elaborated upon in the following subsections.

2.1. *Training Data Generation*

As mentioned earlier, for each selected combination of X_c, we create a specific training set, denoted $\mathcal{D}^c$. The specific X_c combinations we employed are listed in Table 1. The first combination, X_{c1} = 0.533, X_{c2} = 0.666, and X_{c3} = 0.500, is suggested by the challenge organizers and is treated as our baseline case, while the remaining combinations were proposed by us. We should point out that the selection of most of the combinations (with the exceptions of Runs 6 and 7) we use is not arbitrary. We first generated a global training set, $\mathcal{D}^g$, of one million random samples of X_c, X_a, X_e, and s and, then, run the computational model with these inputs. This global training set allowed us to discover intrinsic traits in the response time series as well as identify preferred (though not necessarily optimal) X_c combinations as discussed in the following subsection.

2.2. *Exploratory Data Analysis*

By visual inspection, we have discovered several intriguing relationships between the system parameters and certain statistics of the response time series. For instance, the relationship between X_{a1} and $\overline{Y}_1$ exhibits a pattern similar to a wind turbine power curve (not shown). For brevity, we do not elaborate on these relationships in this paper. Instead, we focus on a key aspect of the response time series that proved critical to the specific ML approach we adopted.

In Fig. 2, we present pairwise scatter plots involving response variables Y_4, Y_5, and Y_6 for the baseline case. Interestingly, the individual samples indicate that the ratios of these responses, Y_5 to Y_4 and Y_6 to Y_4, remain constant, although this ratio varies across the different random samples for the baseline case. Similar trends are also evident in other datasets, $\mathcal{D}^c$. For all the samples, we performed linear regressions as follows: $Y_j^{(k)} = r_{ij}^{(k)} Y_i^{(k)} + c_{ij}^{(k)}$, where i and j were chosen to vary from 4 to 6. The superscript k ranges from 1 to the sample size, N. The resulting slopes $r_{ij}^{(k)}$ and intercepts $c_{ij}^{(k)}$ are used as input features for the LightGBM model. These estimated slopes, r_{ij}, also guided our selection of preferred X_c ranges, as illustrated in Fig. 3. In the top panel, the slope variation is shown as a function of X_{e1} and X_{e2}, and reveals a noisy relationship corroborated by the low Pearson correlation coefficients. The baseline case is within this selected X_c range. In contrast, the bottom panels indicate stronger correlations between the parameters and the slopes. Consequently, we expect that any training sets, $\mathcal{D}^c$, generated for X_c combinations

Table 1. R2 scores from the holdout set. Sample size = 10^6. The lower scores are italicized.

Simulation	X_c	X_{a1}	X_{a2}	X_{e1}	X_{e2}	X_{e3}
Baseline	(0.533, 0.666, 0.500)	0.9996	0.9940	*0.9278*	*0.9204*	0.9995
Run 1	(0.050, 0.850, 0.850)	0.9997	0.9940	0.9970	*0.7612*	0.9999
Run 2	(0.950, 0.550, 0.950)	0.9997	0.9941	0.9947	0.9979	0.9990
Run 3	(0.850, 0.250, 0.750)	0.9997	0.9940	0.9948	*0.6939*	0.9999
Run 4	(0.150, 0.950, 0.450)	0.9997	0.9939	0.9942	*0.6835*	0.9999
Run 5	(0.350, 0.050, 0.150)	0.997	0.9988	*0.8678*	0.9870	*0.9099*
Run 6	(0.670, 0.870, 0.870)	0.9997	0.9940	*0.9732*	*0.3725*	0.9999
Run 7	(0.500, 0.850, 0.850)	0.9997	0.9938	*0.9619*	0.9865	0.9999

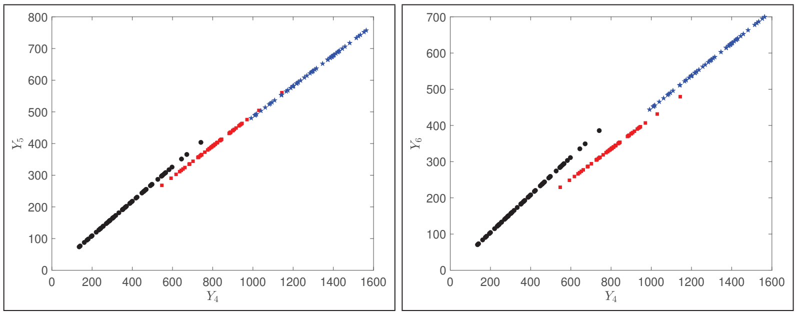

Fig. 2. Scatter plots of Y_5-vs.Y_4 (left panel) and Y_6-vs.Y_4 (right panel). Colored symbols represent distinct samples for the baseline simulation case with X_{c1} = 0.533, X_{c2} = 0.666, and X_{c3} = 0.500.

in these ranges will yield more robust parameter estimations. Cross-validation statistics reported in Table 1 support this expectation.

2.3. *Feature Engineering*

In addition to the ratios, $r_{ij}^{(k)}$ and $c_{ij}^{(k)}$, we have computed numerous features based on the response time series. These include: (a) central moments, (b) spectral moments, (c) min-max values, (d) zero crossings, (e) autocorrelation functions, (f) Hurst exponents, (g) various measures of entropy (*e.g.*, permutation entropy), and (h) various measures of dimensions (*e.g.*, Petrosian fractal dimension). In total, the training set comprises 106 such input features.

2.4. *Strategy for Data Splitting*

In ML, it is standard practice to divide the dataset into training, validation, and holdout (test) subsets. In this work, we reserved 10% of each $\mathcal{D}^c$ dataset as a holdout set. The remaining data were randomly split into training and validation sets using an 80/20 ratio. For each split, we trained one LightGBM model. We then randomized the split and trained a new model, repeating this process M times. The median of the M resulting predictions is used to compute the $R2$ scores (coefficient of determination) reported in Table 1. This resampling strategy is essential for ensuring prediction robustness, especially given the high dimensionality and complexity of the physical system.

2.5. *Hyperparameter Tuning*

The LightGBM model includes several hyperparameters (*e.g.*, number of trees, tree depth) that must be carefully tuned to achieve high predictive performance. Common approaches for hyperparameter optimization include random search and grid search (Géron, 2022). However, these methods are often computationally expensive and may require significant hardware resources.

As a more efficient alternative, we employed FLAML (Fast and Lightweight AutoML Library), developed by Microsoft (Wang et al., 2021), for

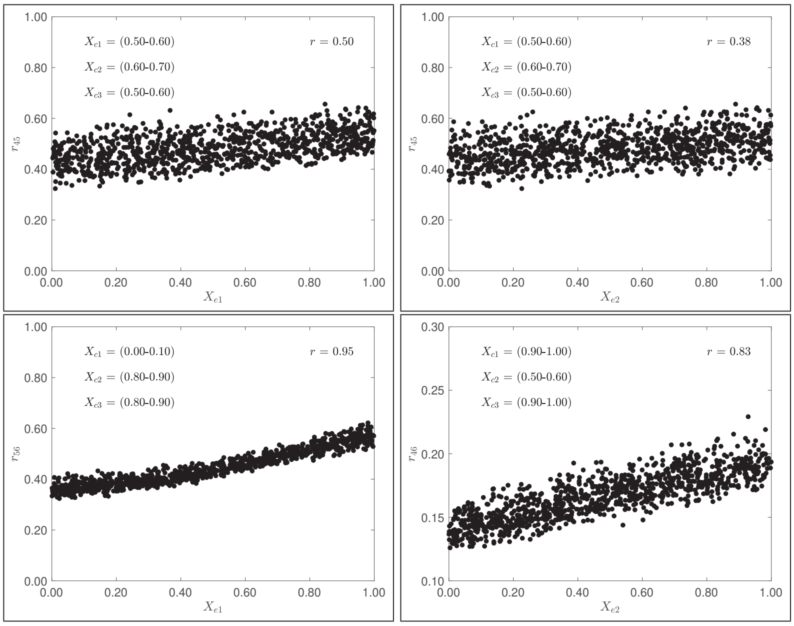

Fig. 3. Variation in slopes, $r_{ij}^{(k)}$, as functions of X_{e1} (left panel) and X_{e2} (right panel). The ranges selected for the X_c parameters are indicated in the top-left corner of each figure, while the estimated Pearson's correlation coefficient (r) values in each case are indicated in the top-right corner of each figure. Note the different y-axis range for the bottom-right X_{e2} plot.

this UQ challenge. Unlike grid search, FLAML takes the available computational time as an input and aims to identify optimal hyperparameters within the specified time budget.

In the analysis, we used $M = 25$ for each $\mathcal{D}^c$ dataset. Given that there are 8 $\mathcal{D}^c$ sets and 5 parameters, this resulted in training a total of $25 \times 8 \times 5 = 1000$ separate LightGBM models. Each model was tuned for a wall-clock time of 10 minutes on a DELL Precision 7960 workstation equipped with an Intel Xeon W9-3495X processor (56 cores, 105 MB cache, base frequency of 1.9 GHz, up to 4.8 GHz Turbo, 350 W). The total computational cost is approximately 170 hours.

2.6. *Results*

Based on the cross-validation statistics reported in Table 1, we conclude that the proposed ML approach can reliably estimate X_{a1}, X_{a2}, and X_{e1} across most of the $\mathcal{D}^c$ datasets. However, accurate prediction of X_{e2} is achieved only in Runs 2, 5, and 7. Since the prediction of X_{e3} depends on all the other estimated parameters (see Fig. 1), the relatively low cross-validation scores for X_{e3} should be interpreted with caution.

Estimates of the X_a and X_e parameters are shown in Fig. 4. The distribution of X_{a1} exhibits a shape that one might interpret as qualitatively similar to a Weibull distribution. For a given run, the estimated values of X_e remain approximately constant. The optimal ranges of estimated X_e parameters are as follows: 0.322-0.323 for X_{e1}, 0.588-0.592 for X_{e2} and 0.525-0.5659 for X_{e3}. Given this high degree of parameter stability, we limit our sampling to just 10 representative values within these narrow ranges for subsequent problem-solving.

To enhance computational efficiency while maintaining the distribution of the parameters, we implement stratified sampling to extract a reduced set (n=100) of representative samples from the estimated X_a distribution. This methodology divides sorted data into equal-sized groups (strata), and then samples from each stratum proportionally, thereby preserving critical distributional

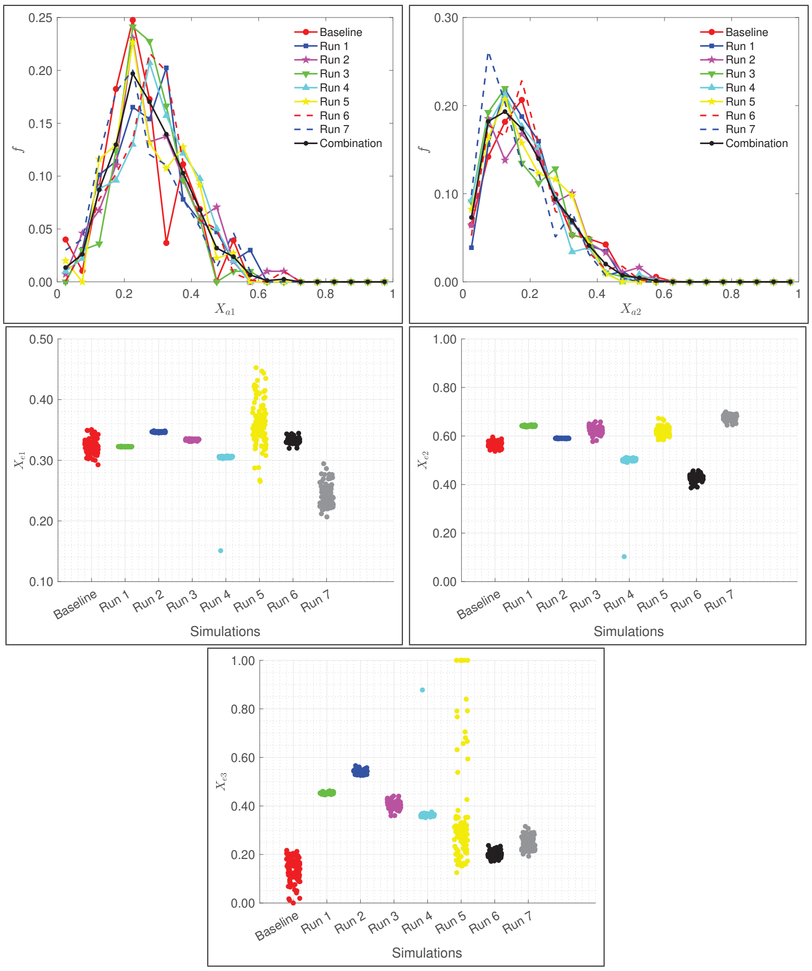

Fig. 4. Top panel: estimated normalized histograms of X_{a1} (left) and X_{a2} (right). Middle and bottom panels: swarm charts of estimated epistemic parameters.

characteristics across the entire parameter space. Kolmogorov-Smirnov tests confirm the statistical validity of this approach, with resulting p-values approaching unity, indicating excellent agreement between the original and sampled distributions, and these reduced sets of X_a are used in the further problems.

3. Strategy and Solution for UQ Problem #1.3

Let $y(X_a, X_e, X_c, s, t)$ be a component of $\mathbf{Y}(X_a, X_e, X_c, s)$. The probabilities of exceeding bounds are:

$$p_u(X_e, X_c, u) = \mathbb{P}\left[\max_{t\in[0,1]} y(X_a, X_e, X_c, s, t) > u\right]$$

$$p_l(X_e, X_c, l) = \mathbb{P}\left[\min_{t\in[0,1]} y(X_a, X_e, X_c, s, t) < l\right]$$

The prediction interval limits at confidence level α are:

$$\tilde{y}(X_c, \alpha) = \inf\left\{u : \max_{X_e\in E} p_u(X_e, X_c, u) \leq 1-\alpha\right\}$$

$$\tilde{y}(X_c, \alpha) = \sup\left\{l : \max_{X_e\in E} p_l(X_e, X_c, l) \leq 1-\alpha\right\}$$

To quantify uncertainty in our model, we employed a Monte Carlo approach by sampling 100 aleatory uncertainty variables $\mathbf{X}_a$ and 10 epistemic uncertainty variables $\mathbf{X}_e$. For each response component, upper and lower bounds were determined at specified confidence levels, α.

The upper bound $\bar{y}(\mathbf{X}_c, \alpha)$ was calculated by first selecting 5,000 potential threshold values, u, uniformly distributed between the minimum and maximum values of the component observed in $\mathcal{Y}(\mathbf{X}_a, \mathbf{X}_e, \mathbf{X}_c, s, t)$. For each threshold, u, we computed the probability $p_u(\mathbf{X}_a, \mathbf{X}_e, u)$ as the proportion of cases where the maximum component value exceeds u across all the sampled $\mathbf{X}_e$. The upper bound $\bar{y}(\mathbf{X}_c, \alpha)$ was then defined as the infimum of u such that the maximum probability does not exceed $1 - \alpha$.

Similarly, the lower bound $\underline{y}(\mathbf{X}_c, \alpha)$ was determined as the supremum of l such that the maximum probability does not exceed $1 - \alpha$.

3.1. *Results*

Results summarizing upper and lower bound estimates for the different response components are presented in Table 2.

4. Strategy and Solution for UQ Problem #2.1

Consider the objective function below:

$$J(\mathbf{X}_a, \mathbf{X}_c) = \int_0^1 \mathbb{E}\left[\sum_{i \in \mathbf{I}_1} y_i(\mathbf{X}_a, \mathbf{X}_e, \mathbf{X}_c, s, t)\right] dt,$$

where $\mathbb{E}[\cdot]$ is the expected value operator with respect to $\mathbf{X}_a$ and s. This objective function is the expected total contribution from the system outputs in $\mathbf{I}_1$ for the chosen $\mathbf{X}_c$. $\mathbf{I}_1$ is defined based on the set of the first 3 components y_1, y_2 and y_3. In this performance-based design problem, we are seeking to estimate the optimal $\mathbf{X}_c^\star$ that maximizes $\min_{\mathbf{X}_e \in E} J(\mathbf{X}_e, \mathbf{X}_c)$.

The performance-based design optimization approach employs differential evolution Storn and Price (1997) to determine the control variables $\mathbf{X}_c$ that maximize performance under worst-case epistemic uncertainty. The problem is formulated to find the maximum of the minimum value of $J(\mathbf{X}_e, X_c)$ across all epistemic uncertainties X_e for each control setting $\mathbf{X}_c$, representing a robust optimization strategy in the presence of uncertainty.

The optimization process employs differential evolution with carefully selected parameters (strategy='rand1bin', mutation=0.8, recombination=0.7, population=15) to balance exploration and exploitation. Within the objective function, the forward model simulation evaluates performance values, with the algorithm identifying the worst-case (minimum) performance across all epistemic scenarios for each candidate control setting. To improve computational efficiency, we implemented a two-level caching system that stores all evaluations in a standard cache while maintaining a separate high-fidelity cache for more detailed assessments. Critical epistemic scenarios that consistently lead to poor performance are tracked and prioritized to focus computational resources where they matter most. Additionally, aleatory samples are periodically refreshed to ensure comprehensive coverage of the aleatory space and to prevent optimization bias.

4.1. *Results*

The Differential Evolution algorithm converged after 10 iterations, yielding optimal control variables $\mathbf{X}_c = [0.1285, 0.4006, 0.6076]$. This configuration maximizes the objective function $J(\mathbf{X}_e, \mathbf{X}_c)$ with a value of 7.7572, as illustrated in Figure 5.

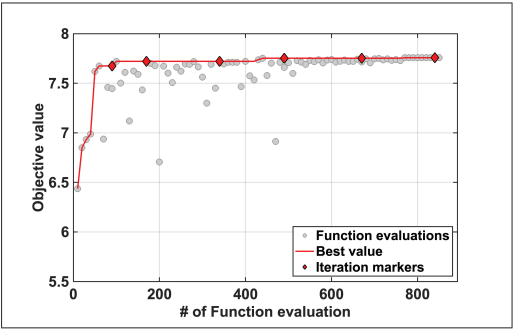

Fig. 5. Performance-based optimization. The iteration markers show iterations 1, 3, 7, 10, 14, and 16 iteration respectively.

Table 2. Upper and Lower Bounds for Different Components at $\alpha = 0.999$ and $\alpha = 0.95$

	$\alpha = 0.999$		$\alpha = 0.95$	
Components	$\underline{y}$	$\overline{y}$	$\underline{y}$	$\overline{y}$
y_1	0	3.35	0.0006	3.35
y_2	0	3.35	0	3.35
y_3	0	3.35	0	3.35
y_4	1.67	1914.44	22.33	1310.65
y_5	1.09	934.12	11.54	638.66
y_6	1.08	805.89	8.96	492.59

5. Strategy and Solution for UQ Problem #2.2

For responses in $I_2 = \{y_4, y_5, y_6\}$, we define limit state functions:

$$g_i(X, s) = c_i - \max_{t \in [0,1]} y_i(X_a, X_e, X_c, s, t), \quad i \in I_2$$

where $c_i = [2750, 2000, 1000]$ respectively. Failure occurs when $g_i(X, s) < 0$. Individual failure probabilities are:

$$pof_i(X_e, X_c) = P[g_i(X_a, X_e, X_c, s) < 0]$$

with system failure probability:

$$pof_{sys}(X_e, X_c) = P\left[\min_{i \in I_2} g_i(X_a, X_e, X_c, s) < 0\right]$$

Our objective is to find X_c minimizing $\max_{X_e \in E} pof_{sys}(X_e, X_c)$.

We considered a similar sampling strategy as mentioned in Section 4. For this problem, we are seeking to minimize pof_{sys} in the outer loop among the maximum pof_{sys} values in the epistemic set E.

5.1. *Results*

The Differential Evolution algorithm converged after minimal iterations to yield optimal control variables $X_c = [0.3735, 0.2502, 0.5868]$. This configuration achieves a zero failure probability, representing an ideal design solution. As illustrated in Figure 6, our findings reveal that the majority of sampled control parameter configurations result in safe designs.

This robust performance can be attributed to our sensitivity analysis, which identified a critical threshold within the parameter X_{a1} specifically, the probability of failure increases significantly when X_{a1} exceeds 0.7. The estimated distribution of X_{a1} shows minimal probability mass in this critical region, explaining the predominance of safe designs across our parameter space. Notably, our analysis indicates that the failure probability demonstrates low sensitivity to both epistemic variables and seed parameters.

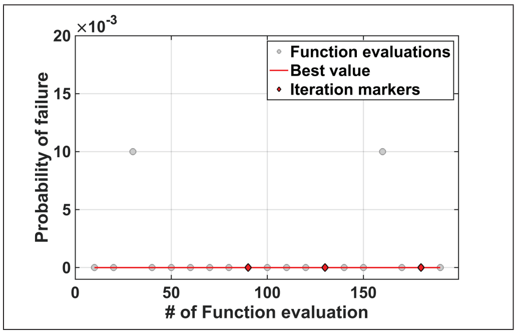

Fig. 6. Reliability-based optimization

6. Strategy and Solution for UQ Problem #2.3

We are seeking to estimate the optimal $\mathbf{X}_c^\star$ that maximizes $\min_{\mathbf{X}_e \in E} J(\mathbf{X}_e, \mathbf{X}_c)$ that maximizes the above objective under the constraint $\max_{X_e \in E} pof_{sys}(X_e, X_c) \leq \epsilon$. We consider a case with ϵ equal to $1e^{-3}$.

The sampling strategy followed the approach detailed in Section 4. To handle constraints, we implemented a penalty method that augmented the objective function by adding a penalty factor multiplied by the difference between the worst-case probability of failure and ϵ. A substantial penalty factor of 10^6 was applied to strongly discourage infeasible solutions.

6.1. *Results*

Our optimization yielded an optimal $\mathbf{X}_c$ value of $[0.1321, 0.3360, 0.6124]$ with an objective function value of 7.729 and a system failure probability of zero. The reported results are obtained after optimization was run for 71.42 hours and the results are presented in Figure 7.

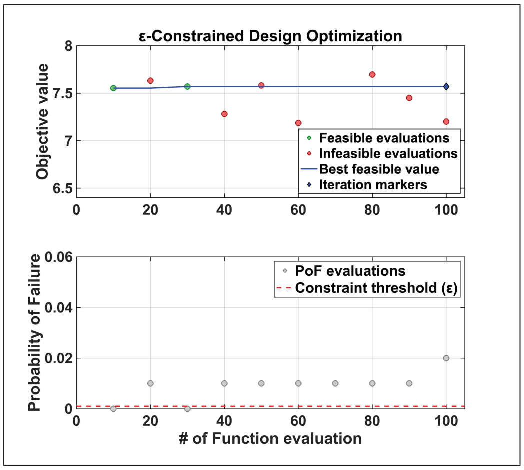

Fig. 7. ϵ-constrained based optimization

7. Concluding Remarks

This paper explores an ML approach to UQ, parameter estimation, and optimization for engineered systems under uncertainty. We employ LightGBM for regression tasks, using over 100 engineered features, including spectral moments, entropy, and fractal dimensions from multivariate time-series data. FLAML is used for efficient hyperparameter tuning under strict compute constraints. Stratified sampling and resampling strategies ensure statistical robustness and distribution preservation. Optimization problems are solved using Differential Evolution, targeting performance maximization and failure probability minimization under epistemic uncertainty. Constraint handling is achieved via penalty methods. The methodology demonstrates how data-efficient ML models, combined with advanced sampling and optimization strategies, can solve complex reliability and performance problems in simulation-heavy environments that are critical to safety.

References

Agrell, C., L. G. Crespo, V. Flovik, E. Vanem, and S. Kenny (2025). The NASA and DNV Challenge on Optimization Under Uncertainty.

Breiman, L. (2001). Random Forests. *Machine learning 45*, 5–32.

Chen, T. and C. Guestrin (2016, August). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM Sig KDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California, pp. 785–794.

Freund, Y. and R. Schapire (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence 14*, 771–780.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis 38*, 367–378.

Géron, A. (2022). *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow* (3 ed.). O'Reilly Media, Inc.

Geurts, P., D. Ernst, and L. Wehenkel (2006). Extremely Randomized Trees. *Machine learning 63*, 3–42.

Machado, M. R., S. Karray, and I. T. de Sousa (2019). LightGBM: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry. In *14th International Conference on Computer Science & Education (ICCSE)*, pp. 1111–1116. IEEE.

Storn, R. and K. Price (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization 11*, 341–359.

Wade, C. (2020). *Hands-on Gradient Boosting with XGBoost and scikit-learn*. Packt Publishing Ltd.

Wang, C., Q. Wu, M. Weimer, and E. Zhu (2021). FLAML: A Fast and Lightweight AutoML Library. *Proceedings of Machine Learning and Systems 3*, 434–447.