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-P9896-cd

Tackling the NASA and DNV 2025 UQ challenge : an Approximate Bayesian Computation framework for surrogate-based optimization under uncertainty

Gatien Chopard
DTU Wind, Technical University of Denmark, Denmark. E-mail: gatch@dtu.dk

Teodor Åstrand
DTU Wind, Technical University of Denmark, Denmark. E-mail: tobas@dtu.dk

Nikolay Dimitrov
DTU Wind, Technical University of Denmark, Denmark.

Performing an optimization task on a complex system can be challenging when input variables are not completely known and when there is inherent randomness in the system response. The first step is therefore to gather information to reduce these uncertainties. In this paper, we use a rejection algorithm based on approximate Bayesian computation to infer input distributions from a limited number of output observations. We define informative metrics to estimate the likelihood of each input variable using a computer model of the real system and variance-based sensitivity analysis. Further, we identify optimal control parameters accounting for both performance and probabilistic constraints. We utilize neural network surrogates to efficiently approximate key relationships, evaluate failure probability, and enable gradient-based optimization approaches.

Keywords: Stochastic model updating, Approximate Bayesian computation, rejection sampling, Surrogate modelling, Uncertainty quantification.

1. Introduction

1.1. *Challenge overview*

This paper presents our answer to the 2025 UQ-challenge on optimization under uncertainty proposed by NASA and DNV. A detailed description can be found in Agrell et al. (2024). The goal of the challenge is to stimulate the development of methods to perform optimization tasks for complex systems with critical applications where inputs are not completely known. It is divided in two problems : Problem 1 aims at defining an uncertainty model on the system input, then building prediction intervals of the system output; Problem 2 consists in optimizing system's control based on an objective function, under constraints on the system output.

1.2. *System and variables definition*

We study a real system whose response $\mathbf{Y}$ depends on an input vector $\mathbf{X} \in [0,1]^8$, composed of :

- A two-dimensional aleatory variable $\mathbf{X}_a = (X_{a_1}, X_{a_2})$ that can be described by a multivariate random variable.
- A three-dimensional epistemic variable $\mathbf{X}_e = (X_{e_1}, X_{e_2}, X_{e_3})$ whose true value X_e^* is unknown.
- A three-dimensional control variable $\mathbf{X}_c = (X_{c_1}, X_{c_2}, X_{c_3})$ that can be chosen when manipulating the system, contrary to $\mathbf{X}_a$ and $\mathbf{X}_e$.

The model response also depends on an aleatory variable ω representing seed-to-seed uncertainty. The response itself is a six-dimensional time series $(\mathbf{y}_1, \ldots, \mathbf{y}_6)$, with time $t \in [0,1]$. The time series are discretized in $m = 60$ time steps such that the response for output dimension i is $\mathbf{y}_i = y_i^{(1)}, \ldots, y_i^{(m)}$, with $t_1 = 0$ and $t_m = 1$. In order to answer the challenge, we get a budget of $N = 10$ samples of system responses $\mathbf{Y}$, for which we can choose the control variable $\mathbf{X}_c$. When calling the real system, $\mathbf{X}_a$ is sampled from its unknown distribution and $\mathbf{X}_e$ takes its true and unknown value $\mathbf{X}_e^*$. A number of $K = 100$ repetitions are obtained for each sampled realization of the control variable, which results in a total of $N \times K = 1000$ system outputs. In the next sections, $j = 1, \dots, N$ will index the $\mathbf{X}_c$ values, $k = 1, \dots, K$ the repetitions and $i = 1, \dots, 6$ the output dimension considered. As an example, $\mathbf{y}_i^{j,k}$ is the i-th dimension time series of the k-th repetition for the j-th $\mathbf{X}_c$ values. In addition to the real system, we also have at our disposal a computer model reproducing the system's response. The model can be used without limits by specifying $\mathbf{X}_a$, $\mathbf{X}_e$, $\mathbf{X}_c$ as well as the random seed in the form of an integer s.

1.3. *Approach*

We will first study the computer model's behavior in section 2. Then, aleatory variable distribution and epistemic variable set inference methodology is discussed in section 4, then put into practice with real system data in section 5. To finish with, we perform an optimization under constraints in section 6.

2. Model behavior understanding

The first step into the challenge is to assess qualitatively and quantitatively the computer model behavior, that is, the output characteristics and how it is influenced by input variables. We identified that the first three dimensions $\mathbf{y}_1$, $\mathbf{y}_2$ and $\mathbf{y}_3$ are bounded (between 0 and 3.35). They can either be a constant equal to those bounds, or a more noisy time series in between. On the other hand, $\mathbf{y}_4$, $\mathbf{y}_5$ and $\mathbf{y}_6$ are also positive, but not bounded, and can reach high values.

We aim at quantifying the different effects of inputs on output variance by estimating the Sobol' indices (Sobol, 2001). For each output dimension and each input variable, the first order and total order Sobol indices are estimated. Both the indices for the output's mean and standard deviation are estimated using the method described in Saltelli et al. (2008). The Sobol's indices for the output mean show that it is mostly driven by X_{a_1}, especially for the first three dimensions y_1, y_2 and y_3 : Its first order indices are $[0.95, 0.88, 0.96, 0.86, 0.71, 0.99]$ and total order indices are $[1.0, 0.98, 1.0, 0.93, 0.84, 1.0]$. The control variables have a small effect on y_4 and y_5, while no $\mathbf{X}_e$ contribution was captured. Figure 1 presents the Sobol's indices for the output standard deviation. The variance contributions for y_1, y_2 and y_3 are much more distributed, and apart from X_{a_1}, consist mostly of interaction effects. On the other hand, the variance of y_4, y_5 and y_6 is mainly explained by X_{a_1} and X_{a_2}, with small effects of $\mathbf{X}_c$ and no effect of $\mathbf{X}_e$.

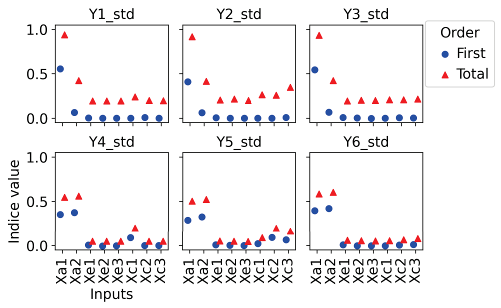

Fig. 1. Sobol' indices of the output standard deviation, estimated with a total of 10^5 simulations.

3. Validation tool by mockup system simulator

The real system observations don't allow for a full assessment of our methods. For instance, we cannot compare the estimated $\mathbf{X}_a$ distribution or $\mathbf{X}_e$ set to their true values because only the system output is observed. Therefore, we use what we call a "mockup system" : we arbitrarily assign a "true" $\mathbf{X}_a$ distribution and $\mathbf{X}_e$ "true" values $\mathbf{X}_e^*$, then run the computer model on $N = 10$ controls and $K = 100$ repetitions for each. The resulting output is treated as a real system observation. That way, we can compare our inference with the known distributions and values that were used to generate the mockup system output. Another benefit is that the mockup system allows to try developed methods on several input configurations and therefore test their robustness.

4. Uncertainty model

4.1. *Approximate Bayesian Computation*

Now that we have a better idea of the model's behavior, we tackle the first challenge's problem, that is, $\mathbf{X}_a$ distribution and $\mathbf{X}_e$ set inference. In order to do so, we choose to use Bayesian sampling methods which are adapted to this configuration where we have noisy and limited data. Denoting by θ, either $\mathbf{X}_a$ distribution parameter or $\mathbf{X}_e$ values, our goal is to estimate a posterior distribution

$$\mathrm{P}(\theta|\mathbf{Y}_{obs}) \propto \mathrm{P}(\mathbf{Y}_{obs}|\theta)\,\mathrm{P}(\theta), \tag{1}$$

where $\mathrm{P}(\theta|\mathbf{Y}_{obs})$ is the posterior distribution of θ having observed $\mathbf{Y}_{obs}$, $\mathrm{P}(\mathbf{Y}_{obs}|\theta)$ is the likelihood of observations given θ and $\mathrm{P}(\theta)$ is the prior distribution of θ. However, due to the black box nature of the system, it is difficult to estimate the likelihood in our case. That's why we lean towards Approximate Bayesian Computation (ABC) methods. The main idea of ABC is to find parameter values for which simulated data $\mathbf{Y}_{sim}$ is close to observed data $\mathbf{Y}_{obs}$, using a distance metric d that is informative on θ(Bi et al. (2022)). We perform the inference of θ posterior using a version of the rejection sampling algorithm(Lintusaari et al. (2017)) : The distance is computed for a large number of candidate parameters θ^*, then the $\alpha\%$ for which it is the lowest are accepted, where α is the acceptance rate.

4.2. *Definition of distance metrics*

Each input is assigned the summary statistics for which its total Sobol' indices are the highest. For instance, the output mean will be used for X_{a_1} inference, while the standard deviation will be used for X_{a_2} and $\mathbf{X}_e$. Additionally, we can take advantage of the multi-dimensionality of the output by picking the relevant dimensions for a given input. As an example, for $\mathbf{X}_{a_2}$ inference we could focus on the standard deviation of y_4, y_5, and y_6 that are the most informative. We then define a distance between simulations and observations which is the mean Bhattacharyya distance between summary statistic distributions. The Bhattacharyya distance is a quantity measuring the overlap between two distributions of density p and q :

$$\mathrm{d_B}(p,q) = -log \int_{\mathcal{X}} \sqrt{p(x)q(x)}dx. \tag{2}$$

For each $\mathbf{X}_c$ value and each output dimension, the summary statistic distribution is computed from the K repetitions thanks to a kernel density estimation :

$$\hat{s}_i^j = \mathrm{kde}\left(\{s\left(y_i^{j,k}\right)\}_{k=1,\dots,K}\right). \tag{3}$$

where s denotes the mean or standard deviation, and $\hat{s}_i^j (j = 1,\dots,N, i \in I)$ are density functions. Then, we compute the Bhattacharyya distance of $\hat{s}_i^j$ between observations and simulations :

$$d_i^j = \mathrm{d_B}\left(\hat{s}_{sim,i}^j, \hat{s}_{obs,i}^j\right). \tag{4}$$

The final distance is the mean of these Bhattacharyya distances over the selected output dimensions and different $\mathbf{X}_c$ values :

$$\mathrm{d}_1(\mathbf{Y}_{obs}, \mathbf{Y}_{sim}|\theta) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{|I|}\sum_{i\in I} d_j^i, \tag{5}$$

where I is the subset of informative output dimensions considered for the inferred input.

The output mean Sobol' indices of figure **??** show that X_{a_1} can be estimated quite independently due to its dominating marginal effect. Moreover, figure 1 shows that if X_{a_1} is known, then the difference in standard deviation between observed and simulated output for the last three dimensions could be attributed to X_{a_2}. These observations entice us to first perform an X_{a_1} inference, then an X_{a_2} inference, and finish with a joint inference of $\mathbf{X}_e$ components.

4.3. *Aleatory variable inference methodology*

We start by investigating $\mathbf{X}_a$ inference. We choose uniform distribution as candidate distributions, which means that $\theta = (a, b)$, where a and b are the bounds of a uniform distribution. The rejection algorithm therefore produces a set of accepted uniform distributions that we average to get the estimated distributions. Neural network or polynomial chaos expansion (pce) surrogates of the output mean and standard deviation were used to save computation time. Figure 2 presents the results of an X_{a_1} and X_{a_2} inference in three mockup system configurations. When the true distribution is a beta, the estimated distribution is able to match the density function shape quite accurately, following the tail of the distribution. However, when the true

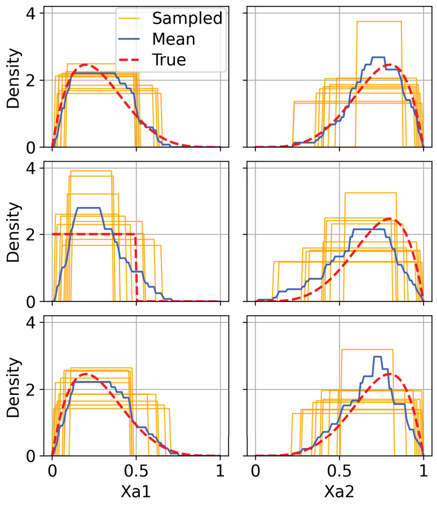

Fig. 2. Inference of X_{a_1} and X_{a_2} in three configurations. The distribution of X_{a_1}, X_{a_2} and $\mathbf{X}_e^*$ are : (*top*) : Beta(2,5), Beta(5,2), $[0.5, 0.5, 0.5]$; (*middle*) : Uniform(0,0.5), Beta(5,2), $[0.5, 0.5, 0.5]$; (*Bottom*) : Beta(2,5), Beta(5,2), $[0.1, 0.8, 0.4]$. The rejection algorithm was run with $n = 200$ iterations, acceptance rate of 0.1 and $K = 100$ repetitions for each control.

distribution is a uniform, the inferred distribution fails to capture the rectangular shape of the uniform density function, yielding a more triangular shape and heavier tails. The inaccuracy in $\widehat{X}_{a_1}$ estimation has an impact on $\widehat{X}_{a_2}$, which is worse for output 2. Lastly, the inference quality is similar between output 1 and output 3, showing that $\mathbf{X}_a$ inference is not drastically affected by assuming an incorrect $\mathbf{X}_e$. Lastly, we learned after further investigation that the inference quality remains constant when adding new control variables for both X_{a_1} and X_{a_2}. This means that we can get a satisfying $\mathbf{X}_a$ inference from the initial baseline design.

4.4. *Epistemic variable inference methodology*

Having tested our methodology for $\mathbf{X}_a$ distribution inference, we can now aim at estimating $\mathbf{X}_e$ set E. The choice of the distance metric is not as straightforward because the effect of $\mathbf{X}_e$ on the output is smaller. The goal is to find metrics for which the distribution when $\mathbf{X}_e$ is correct is very different compared to when $\mathbf{X}_e$ is random, that is, wrong. For that matter, we found that sampling the controls leading to the highest output variance was an effective strategy. We investigate two metrics, based on the Sobol' indices. The first one is the mean Bhattacharyya distance between observed and simulated output standard deviation distribution; the second one is the same but only with y_5.

The inference quality is quantified by estimating the mean average error with several mockup system true $\mathbf{X}_e$ values. We found that, while X_{e_1} and X_{e_2} inferences are generally better than random, X_{e_3} inference is close to random values, or worse. Therefore, we will adopt a conservative approach for our $\mathbf{X}_e$ inference.

5. Inference on the real system

5.1. *Choice of control variables*

Besides the baseline design, we sample our $N = 10$ real system observations as follows :

- One observation is chosen at the center of the safe zone estimated in section 6.
- Two observations are chosen at optimal points under two probability of failure levels, as defined in section 6.
- Three points are chosen to maximize the variance of y_1, y_2 and y_3, respectively.
- The four remaining points maximize the variance of all output dimensions : For each output dimension, the output variance is estimated for a large number of controls and the resulting vector is sorted in decreasing order. We then add the variance ranks for all output dimensions and select the four controls with the lowest ranks.

While sampling at the optimal points will enable to compare the real objective function value to the value estimated with the computer model, the points of high variance are useful for $\mathbf{X}_e$ inference. Choosing points of high variance for single output dimensions allows a better input space exploration, as some of the global high-variance points are close to each other.

5.2. *Inference of aleatory variable*

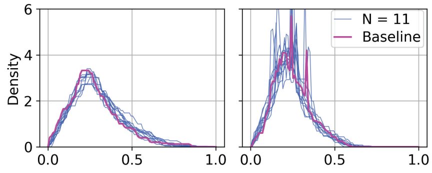

Fig. 3. Inference of X_{a_1} and X_{a_2} on one observation of the real system (magenta) vs 10 repetitions of an inference on $N = 11$ observations, using the Bhattacharyya distance metric. For each repetition, the algorithm was run with $n = 200$ iterations and the acceptance rate set to 0.1. Surrogates are used with $K = 100$ repetitions for the each control variable.

We first perform an inference of X_{a_1} and X_{a_2} on one real system observation, which is the baseline design. The inferred distributions are used to determine the controls of highest output variance, that are then sampled from the real system. Figure 3 compares the inference on the baseline design to repeated inferences on all our $N = 11$ observations. The distribution of $\widehat{X}_{a_1}$ is centered around 0.2, which is where the switch between y_1, y_2 and y_3 low and high values happens. Despite not being able to compare the estimated distributions with true distributions, it is at least coherent with the system's behavior. Moreover, the inference made on the baseline design is indeed similar to inferences made on 11 observations.

5.3. *Inference of epistemic variable*

We perform the $\mathbf{X}_e$ inference on the real system using the methodology and metrics described in subsection 4.4, with $n_{xe} = 400$, and $K = 100$ repetitions for each control variable. For each distance metric, the 10% of the $\mathbf{X}_e$ values with the lowest distance are represented on the histograms of figure 4. Both metrics kept more X_{e_1} and X_{e_2} values on the left hand side on the [0,1] interval, especially the y_5 standard deviation metric which did not keep any X_{e_1} or X_{e_2} past ≈ 0.6. Inferred values are more spread for X_{e_3}. Despite the narrow results of the y_5 standard deviation metric, our

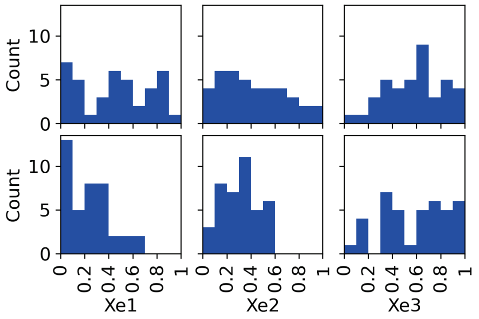

Fig. 4. Histogram of inferred $\mathbf{X}_e$ values for two different metrics. *Top* : Standard deviation Bhattacharyya distance accross all output dimension. *Bottom* : Standard deviation Bhattacharyya distance of y_5. The distances were computed on $n_{xe} = 400$ $\mathbf{X}_e$ values, and only the best 10% are kept.

lack of certainty towards the quality of this metric entice us to use the following conservative approach : For each $\mathbf{X}_e$ dimension, the lower bound of the interval is the minimum value sampled between the two standard deviation metrics, and the upper bound is the maximum value sampled between the two. The resulting $\mathbf{X}_e$ set is $E_1 = [[0.0015, 0.94], [0.00036, 0.97], [0.033, 0.98]]$, which is close to being the [0,1] interval. As a comparison, the resulting $\mathbf{X}_e$ set if we were to trust the y_5 standard deviation metric would be $E_2 = [[0.0015, 0.67], [0.026, 0.58], [0.086, 0.98]]$. More samples would be needed for more accurate bound estimations, and to take into account the distribution of the inferred values inside the bounds. Additionally, one could think of improving the rejection algorithm, for example by implementing a MCMC algorithm, although this would require more tuning.

5.4. *Prediction intervals*

The last part of problem 1 consists of computing prediction intervals of the output for the inferred model and the baseline design $\mathbf{X}_c^{'} = [0.533, 0.666, 0.5]$. To that end, twelve pce surrogates were trained to predict the maximum and minimum of each output dimension over the time series. Seed to seed uncertainty was taken into account by running the computer model $K = 10$

times for each training input. Then, the training output is the mean of the maximums or minimums over the K repetitions.

6. Optimization

6.1. *Visualizing the constraints*

We proceed with our inferred distributions of X_{a1} and X_{a2}. However, without an inferred guess of X_e for the optimization task. To gain an initial insight of the problem and build a foundation for what approaches we want to select for the task, visualization of the optimization problem was employed. We aim to determine the optimal control parameter X_c for a constraints-free performance-based design and a constrained design. This is done by solving the following optimization problem:

$$\max_{X_c} \min_{X_e \in E} J(X_e, X_c) \tag{6}$$

where the objective function is given by the expected contribution from $I_1 = \{y_1, y_2, y_3\}$ as:

$$J(X_e, X_c) = \int_0^1 \sum_{i \in I_1} \mathbb{E}[y_i(X_a, X_e, X_c, s, t)]\, dt. \tag{7}$$

For the constrained design, a constraint describing the probability of failure based on a limit state:

$$g_i(X, s) = c_i - \max_{0 \le t \le 1} |y_i(X, s, t)|, \quad \text{for all } i \in I_2 \tag{8}$$

is introduced, where system failure occurs for $g_i(X, s) < 0$. We create scatter plots describing the relationship between X and $\max |y_i(X, s, t)|$ for $i \in I_2$ over the interval $t \in [0, 1]$, and include the constraints c_i. In figure 5 we observe that certain values of X_{c1} and X_{c2} appear promising to fulfill c_1 and c_2.

To enhance our understanding of the safe X_c domain, we created filtered 3D scatter plots that include only points satisfying all constraints, as shown in the right plot in figure 6, and points violating constraints in the left plot of figure 6. For each point in the X_c space, we sample $n = 100$ points where failure of one sample would lead to the filtering of the corresponding X_c point.

Through this, we aim to develop an initial understanding of a region where the probability of system failure is low. As an estimate, we introduce bounds for X_c as seen in figure 6 where 97% of the points within satisfy the constraints.

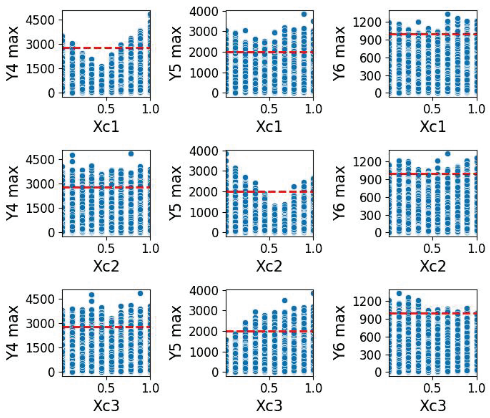

Fig. 5. Scatter plot depicting the relationship of X_c and $\max |y_i(X, s, t)|$ for $I_2 = \{y_4, y_5, y_6\}$, with data from the local model.

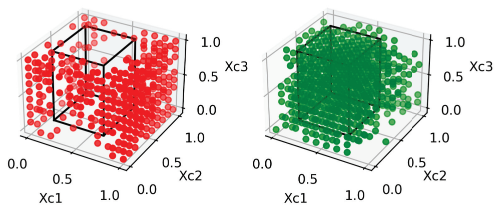

Fig. 6. The left plot displays points that violate constraints, while the right plot shows points that satisfy constraints. Bounds approximating a region exhibiting a lower probability of system failure are included as a wire frame, where the range of X_{c1} is given by $[0.1, 0.65]$, X_{c2} by $[0.25, 0.85]$ and X_{c3} by $[0.0, 1.0]$

6.2. *Neural network surrogate optimization*

To find optimal solutions for the performance-based and constrained designs, we employ a neural network surrogate to approximate relationships of interest. By utilizing these surrogates, we gain the ability to perform a large number of evaluations and leverage gradient-based strategies. Our approach revolves around training surrogates based on data from the simulator, where we only utilize the relationships we find useful for the optimization. This lets us train simpler surrogates compared to fully recreating the simulator. As a baseline for the surrogates we employ, we use a three-layer feed-forward neural network. Further, we would like to highlight that the results in this section have been updated due to modifications in our methodology.

Our primary surrogate was trained to predict the relation between the input parameters X_a, X_e, and X_c and the objective. The training data was sampled from the simulator where X_a was sampled from the inferred distributions, and X_e and X_c were uniformly sampled between 0 and 1. Furthermore, a total of $n = 100000$ samples were drawn, and a random seed was used for each sample. We consider this our objective surrogate, and it achieves a R^2 score of 0.99 on a test set, and the objective has been rescaled to the interval [0,10].

Our secondary surrogate modeled the relationship between the same inputs as our primary surrogate and $max|y_i(X, s, t)|$ for I_2. With this max value surrogate, we allow for fast evaluation of the probability of failure for a given input, and it reaches a R^2 score of 0.99 on a test set.

With our objective surrogate, we can now do more affordable sampling. We use this surrogate to approximate the inner minimization over $X_e \in E$ in equation 6. For each sampled value of X_c, we generate multiple X_e samples, and for each X_e, we sample multiple realizations of X_a. For a given X_e, we compute the mean of the surrogate objective across X_a. Among the sampled X_e values, we identify the one corresponding to the lowest 5th percentile with respect to the objective. Through this process, we get an estimate of the worst-case performance for each sampled X_c. With this data, we now address the outer maximization in equation 6.

6.2.1. *Performance-based optimization*

The inspection of the left plot of figure 7 shows a complex landscape of the data generated for $\min_{X_e \in E} J(X_e, X_c)$ using our objective surrogate. From the slice present, we can identify areas with high and low values, however, the data appears noisy, where peaks and valleys can lead us to local optimums.

A new surrogate is trained to map the design variables X_c and the worst-case scenario data we generated from our objective surrogate. The primary purpose of this new surrogate is to capture the underlying trend of the data and provide a smoother representation. This surrogate achieves an R^2 score of 0.74 with respect to the data it approximates, and its ability to capture underlying trends can be studied in 7. Furthermore, with a smoother representation of what we want to maximize, we now cater to gradient-based approaches to efficiently search for optimum candidates.

Further, through the same sampling methods we used to gather data for $\min_{X_e \in E} J(X_e, X_c)$, we now search for $\max_{X_e \in E} J(X_e, X_c)$ for our candidate optimal control parameters giving us the the estimated range $J_{\text{range}}(X_c)$. However, for the optimization, we only consider $\min_{X_e \in E} J(X_e, X_c)$ as this is our objective.

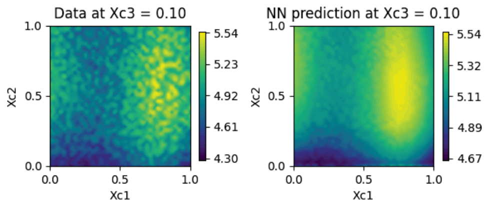

Fig. 7. Sliced contour plots showing gathered data of $\min_{X_e \in E} J(X_e, X_c)$ from our objective surrogate (left) and a surrogate trained on to predict $\min_{X_e \in E} J(X_e, X_c)$ from X_c (right).

Three optimization algorithms, L-BFGS-B, TNC, and slsqp from scipy Virtanen et al. and SciPy 1.0 Contributors (2020), were used to perform the optimization. To mitigate the risk of converging to local minima, each algorithm was initialized from multiple starting points in the X_c space. Through this approach, we obtain the unconstrained optimal X_c seen in table 1.

6.2.2. *Constrained optimization*

For the constrained optimization, we search for optimums under the constraint:

$$\max_{X_e \in E} P_{f,\text{sys}}(X_e, X_c) \leq \epsilon \quad (9)$$

We investigate two levels of allowed probability of system failure $\epsilon = 10^-3$ and $\epsilon = 10^-4$, where the systems failure probability is defined as:

$$P_{f,sys}(X_e, X_c) = P\left[\min_{i \in I_2} g_i(X, s) < 0\right] \quad (10)$$

We employ the same approach as for the performance-based optimization, however, now with an added evaluation step for the found X_c candidates, where the failure rate of each optimum is evaluated from $n = 1000000$ samples using our max value surrogate.

To encourage optimum candidates that pass the constraints, we consider the bounds from figure 6 as bounds for the initial start points of the optimizers. As the constraints are only addressed after the optimizer has converged, we strive to avoid finding optimums where the probability of failure is high. By setting bounds for the initial guesses and not setting strict bounds for the optimizer, we still allow X_c candidates outside our defined bounds. Through this approach, we obtain the following constrained optimal X_c seen in table 1.

Table 1. Best result considering all algorithms used and their respective starting points for the performance-based (first) and constrained optimization. The objective $J(X_e, X_C)$ is scaled to the interval [0,10], and the lower bound of $J_{\text{range}}(X_C)$ is the maximized objective.

X_c	$J_{\text{range}}(X_C)$	ϵ
$[0.740, 0.453, 0.664]$	6.054 - 6.677	none
$[0.637, 0.738, 0.645]$	6.005 - 6.719	10^-3
$[0.576, 0.629, 0.669]$	5.964 - 6.604	10^-4

To further investigate the probability of system failure, we aim to approximate its range by sampling our max value surrogate for multiple X_e and X_a. After $n = 10000$ X_e samples with $n = 100000$ X_a samples, we obtain the range $[2.5 \times 10^{-4},\ 8.5 \times 10^{-4}]$ for X_c = [0.637, 0.738, 0.645] and $[7 \times 10^{-5},\ 4.6 \times 10^{-4}]$ for X_c = [0.576, 0.629, 0.669]. From this, we observe that our X_C optimum for $\epsilon = 10^-3$ does not exceed the limit for failure probability considering the approximated range. However, for the optimum X_C for $\epsilon = 10^-4$, even though this candidate exhibits a lower probability of system failure, its upper bound surpasses our desired limit of $\epsilon = 10^-4$.

Furthermore, we observe that our optimal candidates, both the performance-based and constrained, fall within or near our approximated safe zone from figure 6. Hence, there could be a potential overlap with the domains of interest regarding the two optimization problems.

Acknowledgement

The efforts of the DTU team are supported by the IntelliWind project, funded through the EU MSCA Doctoral Networks programme with grant number 101168725. We also thank Azelice Ludot, Moritz Gräfe and Tuhfe Göçmen from DTU Wind for their valuable inputs throughout the project.

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., K. He, Y. Zhao, D. Moens, M. Beer, and J. Zhang (2022). Towards the NASA UQ challenge 2019: Systematically forward and inverse approaches for uncertainty propagation and quantification. *Mechanical Systems and Signal Processing 165*, 108387.

Lintusaari, J., M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander (2017). Fundamentals and recent developments in approximate bayesian computation. *Systematic biology 66*(1), e66–e82.

Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons.

Sobol, I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation 55*(1-3), 271–280.

Virtanen et al. and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17*, 261–272.