Robustimizer

A graphical user interface application for efficient uncertainty quantification, robust optimization, and reliability-based optimization of processes and designs

Nejadseyfi, Omid

DOI
10.1016/j.softx.2025.102077

Publication date
2025

Document Version
Final published version

Published in
SoftwareX

SoftwareX 30 (2025) 102077

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Robustimizer: A graphical user interface application for efficient uncertainty quantification, robust optimization, and reliability-based optimization of processes and designs

Omid Nejadseyfi

Computational Design and Mechanics, Precision and Microsystems Engineering Department, Mechanical Engineering Faculty, Delft University of Technology, Mekelweg 2, 2628 CD, Delft, The Netherlands

ARTICLE INFO

Keywords:
Uncertainty quantification
Robust optimization
Reliability-based optimization
Surrogate model

ABSTRACT

The primary goal of this work is to provide easy-to-use and cutting-edge optimization software designed to handle uncertainties, intended for use in research and education. Robustimizer offers efficient uncertainty quantification through exact analytic formulas using specific surrogate models, such as Gaussian Processes. Moreover, it supports integration with other software packages, and automatic updating of initial design space through exploration–exploitation techniques, among other features. This software has proven its value in sustainable manufacturing, where optimizing processes to reduce environmental impact while managing uncertainties is critical. In this article, the Robustimizer graphical user interface is introduced as a domain-independent optimization tool for surrogate-model-based robust and reliability-based design or process optimization.

Code metadata

Current code version	V2024.1.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00586
Permanent link to Reproducible Capsule	N/A
Legal Code License	GPL 3
Code versioning system used	git
Software code languages, tools, and services used	MATLAB
Compilation requirements, operating environments & dependencies	MATLAB App Designer
If available Link to developer documentation/manual	https://github.com/onejadseyfi/Robustimizer/tree/main/documentation
Support email for questions	info@robustimizer.com

1. Motivation and significance

One-stage processes are common across many disciplines, including engineering, physics, biology and economics. These processes can be viewed as input–output systems. For example, in power generation, adjusting input variables such as fuel type, temperature, and pressure affects the efficiency and output capacity of the plant. Similarly, in manufacturing and design engineering, modifying parameters related to material properties, dimensions, and tolerances can significantly impact the functionality and durability of a product. Due to the presence of numerous input variables, design or process optimization methods are typically employed to obtain optimal output and meet requirements or constraints.

In practice, some input variables are easy to control and are referred to as design variables. However, some input variables are either uncontrollable or very expensive to control. Such variables, which are stochastic in nature, are referred to as noise or uncertain variables. When uncertain variables are present, the response is no longer deterministic. Both robust and reliability-based design optimization methods address optimization in the presence of uncertainties.

Computer models or costly experiments are often used for optimizing designs or processes. The presence of uncertainties introduces

E-mail address: o.nejadseyfi@tudelft.nl.

https://doi.org/10.1016/j.softx.2025.102077
Received 3 November 2024; Received in revised form 26 January 2025; Accepted 28 January 2025
Available online 15 February 2025

challenges in optimization. One major challenge is predicting the effects of uncertain input on the uncertainty of the response (uncertainty quantification). Furthermore, optimization under uncertainty becomes even more challenging when the process is nonlinear, when only partial knowledge of the uncertainty is available, and when correlations exist among the variables. Optimizing a process or design under uncertainty using computer models requires numerous evaluations, and these models are typically computationally expensive. Experimental approaches for optimization are also very costly. Often, an approximate model known as a surrogate model, is built from the limited number of computer simulations or experiments. Once the surrogate model is constructed, assessing the response for an unsimulated or unmeasured point becomes more cost-effective than running simulations or performing experiments. The surrogate model can then be used to represent the approximate relationship between input and output, and to perform uncertainty quantification and optimization.

Despite their potential, robust and reliability-based optimization remain limited to specific research areas. One of the main reasons hindering the practical application of these techniques is that existing resources are not easy to use. Additionally, building a surrogate model and using approximation techniques, such as Monte Carlo simulations for uncertainty quantification, are computationally demanding. By developing a graphical user interface an easy-to-use and cutting-edge optimization software is provided with the convenience of a self-guided interface. Additionally, the focus is on improving the efficiency and accuracy of uncertainty quantification through surrogate models by implementing analytic formulations [1].

The main focus of Robustimizer is on one-stage processes, a different class of problems than those addressed, for instance, in ROME (Robust Optimization Made Easy) [2], RSOME (Robust Stochastic Optimization Made Easy) [3], and ROmodel [4]. ROME, RSOME, and ROmodel include a decision variable that can be adjusted once uncertainty is revealed, and are sometimes referred to as distributionally robust optimization. These methods are common in control engineering, with their formulation focusing on the expected value or the mean of a response. However, Robustimizer is applicable to one-shot problems, where there is no intermediate decision variable to adjust (sometimes referred to as offline quality control). This approach is rooted in Taguchi robust design [5], which aims to identify the input variables for which the objective function is least sensitive to given uncertainties. Moreover, Robustimizer focuses on surrogate-model-based uncertainty quantification, where expensive model evaluations are replaced by a surrogate model. Additionally, Robustimizer provides analytical expressions for uncertainty quantification, robustness calculation, and reliability estimation. For this purpose, not only the mean but also higher-order moments, such as standard deviation and skewness, are considered. Lastly, all these features are incorporated into an easy-to-use user interface (UI), which is not offered by existing tools.

In the following sections, Robustimizer's potential as a valuable tool for diverse research and educational applications is highlighted. While it has already been employed to enhance sustainability in the manufacturing sector [6], its versatility opens up new avenues for optimizing processes and designs, leading to greater efficiency and effectiveness across various domains.

2. Software description

2.1. Software architecture

The architecture of the Robustimizer software is primarily based on the *clean architecture* principles [7]. The application is divided into several layers, each with its own responsibilities and dependencies as shown in the *onion* diagram of Fig. 1. The layers are designed to be loosely coupled, allowing for flexibility, testability, and ease of maintenance. The user interface (UI) layer is responsible for presenting information to the user and handling user interactions. The main UI is created using MATLAB App Designer. The UI layer communicates with the use cases layer to perform the necessary operations in response to user input. The use cases layer contains the application's logic and use cases. It orchestrates the interactions between the data model and the calculations performed in the kernel, as well as providing validation and error handling for the application's functionalities. The inner layer which is the domain layer, contains the data model representing the structure and relationships of the application's data, as well as the *kernel* calculations and algorithms. It must be noted that the dependencies between the layers are unidirectional, with each layer depending only on the layers inside it. This allows for a clear separation of concerns and makes it easier to test and maintain the application. In particular, the use cases layer is designed to be independent of the UI layer, allowing it to be tested in isolation. The attentive reader may have noticed that the *interface adapters* layer from the original clean architecture diagram [7] is not explicitly depicted here. This simplified approach is intentionally used, where the UI layer directly interacts with the use cases layer.

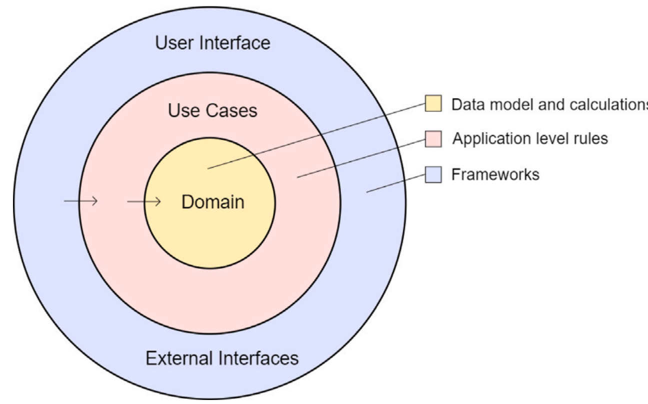

Fig. 1. Various layers in Robustimizer software.

Fig. 2 illustrates the step-by-step workflow in Robustimizer, which is linked to the tabs in the main window. The UI of Robustimizer consists of six tabs. The user begins to formulate the robust optimization problem and proceeds in the following order:

- Tab 1 initiating the problem by defining design variables, noise variables and responses.
- Tab 2 creating design of experiments (DOE) and importing the results of model evaluation
- Tab 3 surrogate modeling and validating.
- Tab 4 defining optimization settings, algorithms, uncertainty quantification method, objective function and reliability constraints.
- Tab 5 robust optimization
- Tab 6 enhancing the surrogate model by exploration and exploitation

While using the application, the user receives guidance through pop-up windows, context-sensitive help, and error-based assistance.

2.2. Software functionalities

In the following sections, a concise summary of the software's functionalities is presented. The reader is referred to the online documentation for detailed information.

2.2.1. Initializing problem

Robust optimization begins by defining the number of design variables, noise variables, and responses. The user specifies the number of design variables, their names, and the upper and lower bounds. For the independent noise variables, the user enters the number of

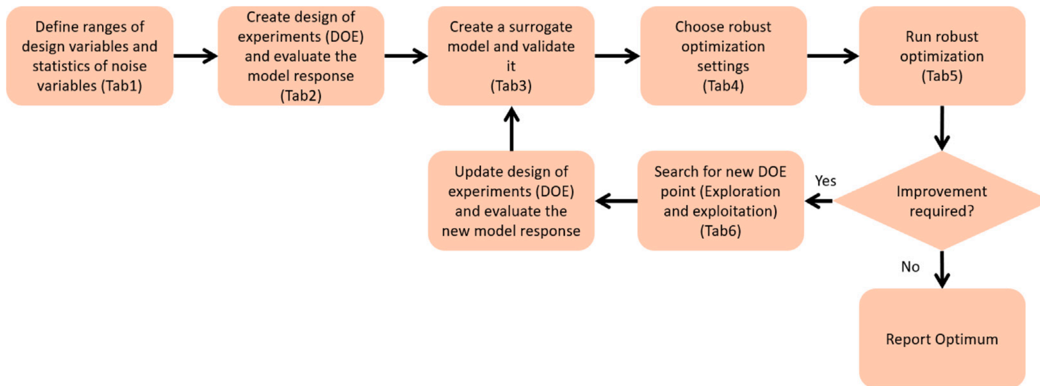

Fig. 2. Flowchart of robust optimization using Robustimizer.

noise variables, their names, and the corresponding mean and standard deviation. Robustimizer not only supports independent noise variables but also performs principal component analysis (PCA) on correlated noise data imported from measurements. In the case of correlated noise data, rather than filling in the statistics in the table, the input data must be provided by the user as a tab-separated text file, where the number of tabs corresponds to the number of noise variables. The number of rows indicates the number of data points.

The correlations among noise variables are then evaluated using covariance PCA [8]. The purpose of PCA is to achieve an uncorrelated subspace, potentially reduce the parameter space and avoid sampling of physically-unlikely parameter combinations when the DOE is created.

Robustimizer requires a single output as the main response. If additional responses are present, the user can either combine them (e.g. weighted sum) to form a single output or treat them as constraints during optimization.

2.2.2. Making a DOE and model evaluation

In this tab the user creates a DOE and imports the results of the model evaluation. Robustimizer supports not only standard methods, such as Latin hypercube sampling (LHS) or factorial design, but also offers the flexibility to import user-defined DOEs as a text file. In this case, a tab-separated file must be provided, where the number of rows corresponds to the number of DOE points and the number of columns corresponds to the total number of variables.

The model (sometimes referred to as a black-box function) is then evaluated at each DOE point. The response can come from a computer simulation, an analytical model evaluation, or an experimental measurement.

Process responses can be imported into the application using two different methods. First, the response can be evaluated outside Robustimizer and imported via a tab-separated text file. Alternatively, an executable can be selected that reads the DOE points from a text file named *in.txt*, evaluates the model, and writes the response to another text file named *out.txt*. This feature offers a significant advantage, as it not only allows the automatic import of process model simulation results into the software but also enables the iterative addition of new DOE points during later optimization stages.

2.2.3. Surrogate modeling

Gaussian Processes (GPs) are widely used for surrogate modeling and are therefore implemented in Robustimizer to describe the response of the process model. The DACE toolbox is used for this purpose, and readers are referred to [9] for more information. Using the *Surrogate Model* tab in Robustimizer, the user fits, plots, and validates the surrogate model. Validation techniques help ensure that the surrogate models accurately represent the underlying system or process. Robustimizer supports leave-one-out cross-validation (LOOCV), in which one data point is left out while the surrogate model is trained on the remaining data points. This process is repeated for each data point in the dataset, and the results are averaged to provide the final accuracy estimate.

2.2.4. Optimization settings

The optimization method, uncertainty quantification approach, objective function, and reliability constraints can be adjusted in this tab, with the relevant input fields being enabled or disabled accordingly. Two uncertainty quantification methods are available in Robustimizer. The analytical method is the default option, as it provides a more accurate and faster evaluation of uncertainties. However, Monte Carlo sampling is also available, allowing the user to specify sample sizes as needed or import user-defined samples.

The objective function to minimize can be selected in different ways under the *Objective Function for Main Response* panel. For non-normally distributed responses, the accuracy of the predictions can be improved by accounting for the skewness [10]. If there are constraints, the user can adjust each constraint separately. For non-normally distributed constraint responses, it is also possible to improve reliability predictions by considering the skewness of the response.

2.2.5. Robust optimization

After providing necessary input in the previous steps, the user proceeds with robust optimization. By pressing the *Perform optimization* button, the optimization starts, and once the results are generated, they are shown in the text box. To visualize the scatter of each response at the predicted optimum and to calculate the response statistics, the user performs a Monte Carlo analysis at the optimum under the *Visualize scatter on optimum* panel.

2.2.6. Exploration and exploitation

Surrogate model optima may not align with the true optimum due to sparse sampling around the predicted optimum or underexplored regions from the initial DOE design. To improve accuracy, Robustimizer uses the expected improvement method [11,12] as a default option, which balances local and global search for new infill points.

Adding new infill points is performed either manually or automatically. If the user defines an executable in the second tab, the automatic option is recommended. In this case, by specifying the number of new DOE points, Robustimizer automatically finds the best point, adds it to the existing DOE, runs the executable, extracts the results, fits a new surrogate model, finds the robust optimum, and repeats this procedure. The new DOE point and the optimization results are displayed in the text boxes after each step.

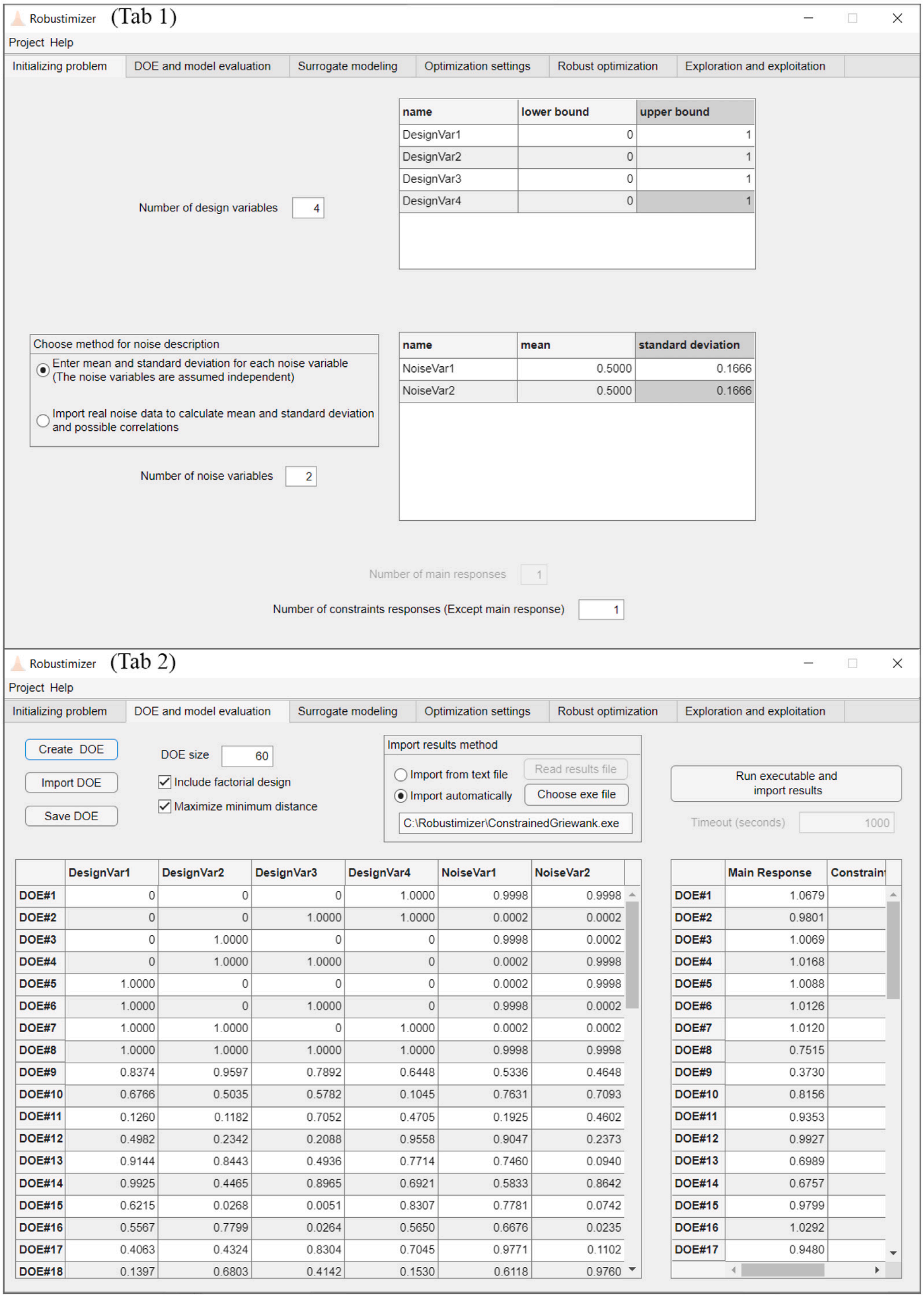

Fig. 3. Illustrative example using Robustimizer (Tabs 1 and 2).

3. Illustrative examples

3.1. Mathematical function

In this example, two mathematical expressions are used as the process model. This example serves merely as an illustrative case. It is important to note that the underlying input-output relationships in real-world scenarios are often unknown and that is the main reason to build a surrogate model. The modified Griewank test function in Eq. (1) is used as the main response and the Robustimizer test function in Eq. (2) is used as the constraint response. Both functions have six inputs with four of these inputs considered as design variables and two as uncorrelated noise variables. An executable file is provided in the repository that evaluates both functions with specific inputs. The ranges of design variables and the statistics of the noise variables are listed in Tables 1 and 2, respectively.

$$f(x) = \beta_1 \left(\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \right) + \beta_2 \tag{1}$$

$$g(x) = \left(x_1 x_4 + \frac{1}{x_2 + \frac{1}{x_3 + \frac{0.1}{x_4 + 0.5}}}\right)^2 - 2\cos(\pi x_2 (x_5 + 0.95 x_6)) + \frac{0.3}{e^{(3x_6 + 0.2x_1)}} \tag{2}$$

Table 1
Ranges of design variables.

Variable	Lower bound	Upper bound
Design Var. 1 (x_1)	0	1
Design Var. 2 (x_2)	0	1
Design Var. 3 (x_3)	0	1
Design Var. 4 (x_4)	0	1

Table 2
Statistics of noise variables.

Variable	Mean	Standard deviation
Noise Var. 1 (x_5)	0.5	0.1666
Noise Var. 2 (x_6)	0.5	0.1666

This data can be entered in the first tab of Robustimizer as shown in Fig. 3. In the second tab, a DOE is generated by pressing the *create DOE* button with the selected options. Then the executable file is selected to evaluate the responses on the DOE points (Fig. 3). In the third tab, the surrogate model is fitted, visualized, and validated as shown in Fig. 4. The responses are plotted as a function of only two variables, with the remaining variables set to nominal values. The user can adjust the plot using interactive buttons on the plot, as well as sliders and the drop-down menu in the *Plot surrogate model* panel. After fitting the surrogate model, the user specifies the settings shown in Fig. 4. Robustimizer offers various robust formulations and reliability-based constraint calculations [13–15]. In this example the following robust optimization problem with a reliability-based constraint is solved:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mu_f(\mathbf{x}) + 3\sigma_f(\mathbf{x}) \\ \text{subject to} \quad & \mu_g(\mathbf{x}) + 3\sigma_g(\mathbf{x}) \leq 1 \\ & \mathbf{lb} < \mathbf{x} < \mathbf{ub} \end{aligned} \tag{3}$$

in which $\mathbf{x}$ is the vector of design variables as shown in Table 1. After selecting relevant settings, the user proceeds with solving the robust optimization problem in Tab 5 which is shown in Fig. 5. The results of the robust optimal design points are displayed in this tab. Uncertainty quantification which is an essential ingredient in robust optimization is performed and the statistical moments are displayed alongside the histograms of the response probability distribution as shown in Fig. 5.

Furthermore, the user can optionally improve the surrogate model by adaptively adding extra DOE points in the last tab as shown in Fig. 5. In this case the required number of new DOE points is entered, and Robustimizer automatically repeats the robust optimization steps.

3.2. Linking robustimizer to finite element analysis packages

Many renowned commercial and open-source software tools, such as Abaqus, Ansys, COMSOL, and Altair HyperWorks for finite element analysis (FEA), OpenFOAM, Fluent, and STAR-CCM+ for computational fluid dynamics (CFD), and LAMMPS for molecular dynamics simulations, can be run from the command line. These tools allow users to modify and process simulation files without opening their respective UIs, making them ideal for automation and batch processing workflows such as those implemented using Robustimizer. This example shows the procedure to link Robustimizer with the commercial FEA package COMSOL Multiphysics.

3.2.1. Problem description

As an illustrative example, Fig. 6-a shows a bracket made of Aluminum 6061-T6 with a given ellipsoidal service hole. The bracket is fixed at the top edge, and a distributed load of F is applied in the $-Y$ direction as shown in Fig. 6-b. The designer can optimize the location of the center of the ellipsoid, x, to minimize the deflection of the bracket, δ, while ensuring that the stress in the bracket does not exceed the yield stress, such that $S_{\max} < S_Y$. However, in practical scenarios, uncertainties are often associated with several parameters. In this example, two sources of uncertainty are considered: one related to the use case and the other to manufacturing. During the use of the bracket, the applied load may deviate slightly from its nominal direction, with a standard deviation of $\sigma_{\theta_1} = 2$ degrees. Additionally, during manufacturing, misalignment may cause the ellipsoid to be slightly rotated, with a standard deviation of $\sigma_{\theta_2} = 1$ degree, as shown in Fig. 6-c. Under these conditions, the robust optimization problem is defined as:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mu_\delta(x) + 3\sigma_\delta(x) \\ \text{subject to} \quad & \mu_{S_{\max}}(x) + 3\sigma_{S_{\max}}(x) \leq S_Y \\ & 16 < x < 24 \ \ [\text{mm}] \\ & \theta_1 \sim \mathcal{N}(0,2) \ \ [\text{deg}] \\ & \theta_2 \sim \mathcal{N}(0,1) \ \ [\text{deg}] \end{aligned} \tag{4}$$

3.2.2. Prepare FE analysis via an executable file

To solve the optimization problem, the maximum stress and deflection must be obtained for every DOE point. In this section, instead of the cumbersome process of manually changing parameters in the simulation software and importing the results using a text file, Robustimizer is linked to COMSOL Multiphysics using an executable file. The following workflow is used to create this executable file.

- Open and read the contents of a file named *in.txt*, which contains the DOE points saved in a tab-separated format. The number of rows corresponds to the number of DOE points, and the number of columns corresponds to the total number of design and noise variables.
- Assign the parameters in each row to the corresponding parameters in the simulation software.
- Run the simulation for one DOE point, extract the results, and proceed to the next DOE point.
- After completing all simulations, write the results to a tab-separated file named *out.txt*, where the number of columns corresponds to the number of responses defined in Tab 1.
- Compile the script into an executable file with the .exe extension.
- Test the executable independently of Robustimizer by placing a test input file named *in.txt* in the same folder. Run the executable file from the command line, wait for the simulations to complete, and verify that the *out.txt* file is generated. This procedure must be tested in two cases, one with multiple entry rows and another with a single entry row (one DOE point).

Although each commercial or open-source software has its own unique requirements, the overall workflow remains consistent. To create the script and provide the executable to Robustimizer, Python is used to follow the procedure outlined above. To run COMSOL simulations within this Python script, the following command is used:

comsolbatch -inputfile RobustBracket.mph -pname x_{center}, θ_1, θ_2 *-plist* ... "DOE$_{i1}$ [mm]", "DOE$_{i2}$ [deg]", "DOE$_{i3}$ [deg]" *-methodcall methodcall1 -nosave* in which DOE$_{i1}$ to DOE$_{i3}$ are the values that are read from the *in.txt* file. Finally, the script is compiled into an executable using *PyInstaller*. More details and tutorials can be found in the online repository.

To run the *comsolbatch* command, certain considerations must be addressed. In this example, the bracket is parametrically modeled in COMSOL. An evaluation group is created to write the maximum displacement and stress to a temporary file after each simulation. A *method* is then created using the Application Builder of COMSOL and is added to the *Global Definitions*. The following lines are required in the *method* to run the study and evaluate the results:

model.study("std1").run();
model.result().evaluationGroup("eg1").run();

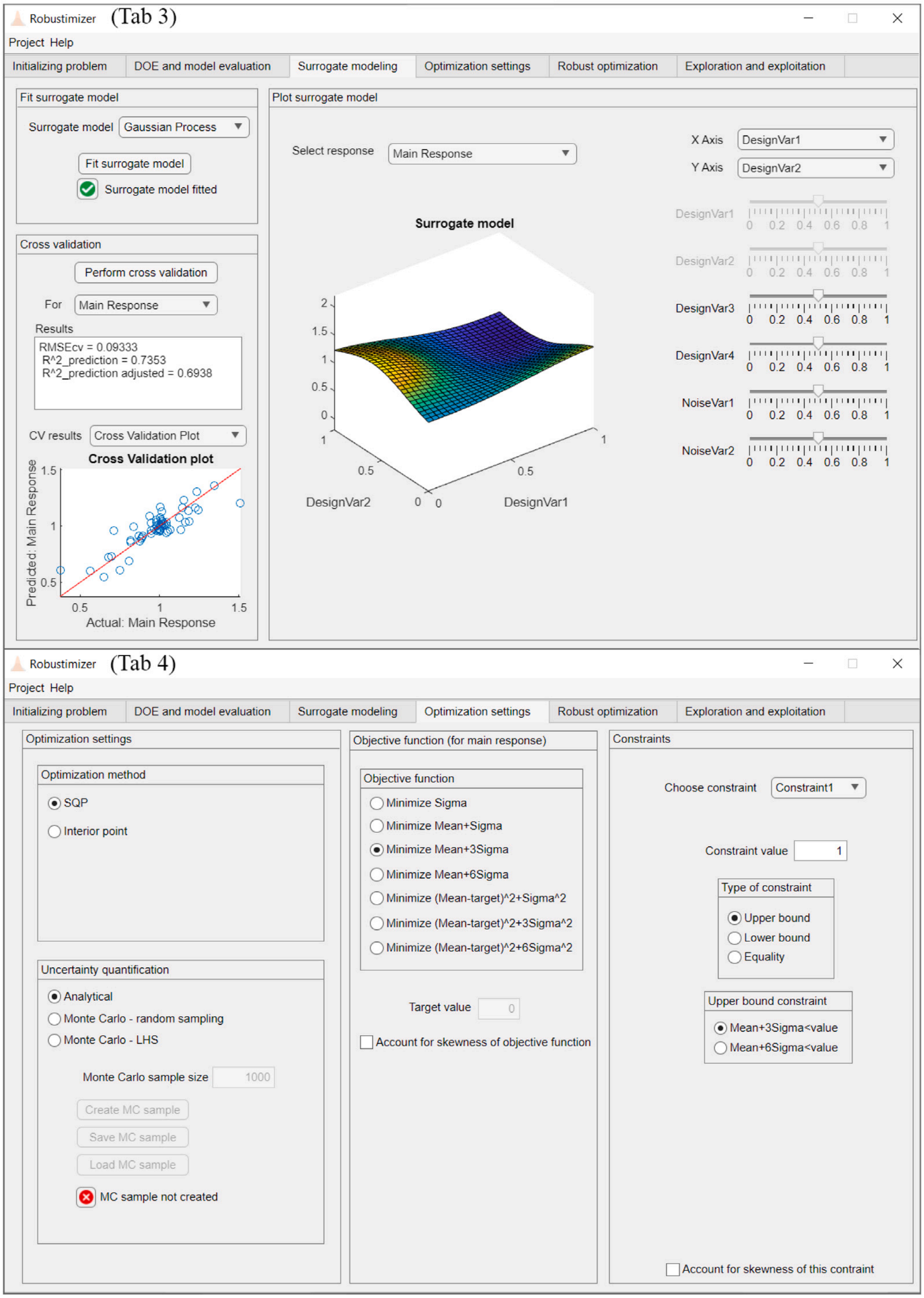

Fig. 4. Illustrative example using Robustimizer (Tabs 3 and 4).

Additionally, for writing to disk via a *method*, the security settings in COMSOL preferences must allow file system access for all files.

3.2.3. Running robustimizer

After providing the input in Tab 1 of Robustimizer, the user generates a DOE with 20 points using LHS. In the second tab, the user provides the executable file created in the previous section and clicks the *Run script and import results* button. Robustimizer then waits for the file *out.txt* to appear in the same folder and imports all results at once into the corresponding table. The next steps are similar to the previous example. A supplementary video is provided for the interested readers to explain the step by step process. Starting with a DOE of 20 points and automatically adding 2 additional points in Tab 6, the expected improvement falls below 0.5%, and the robust optimum is determined to be at $x = 19.5$ [mm] while the deterministic optimum is at $x = 18.7$ [mm].

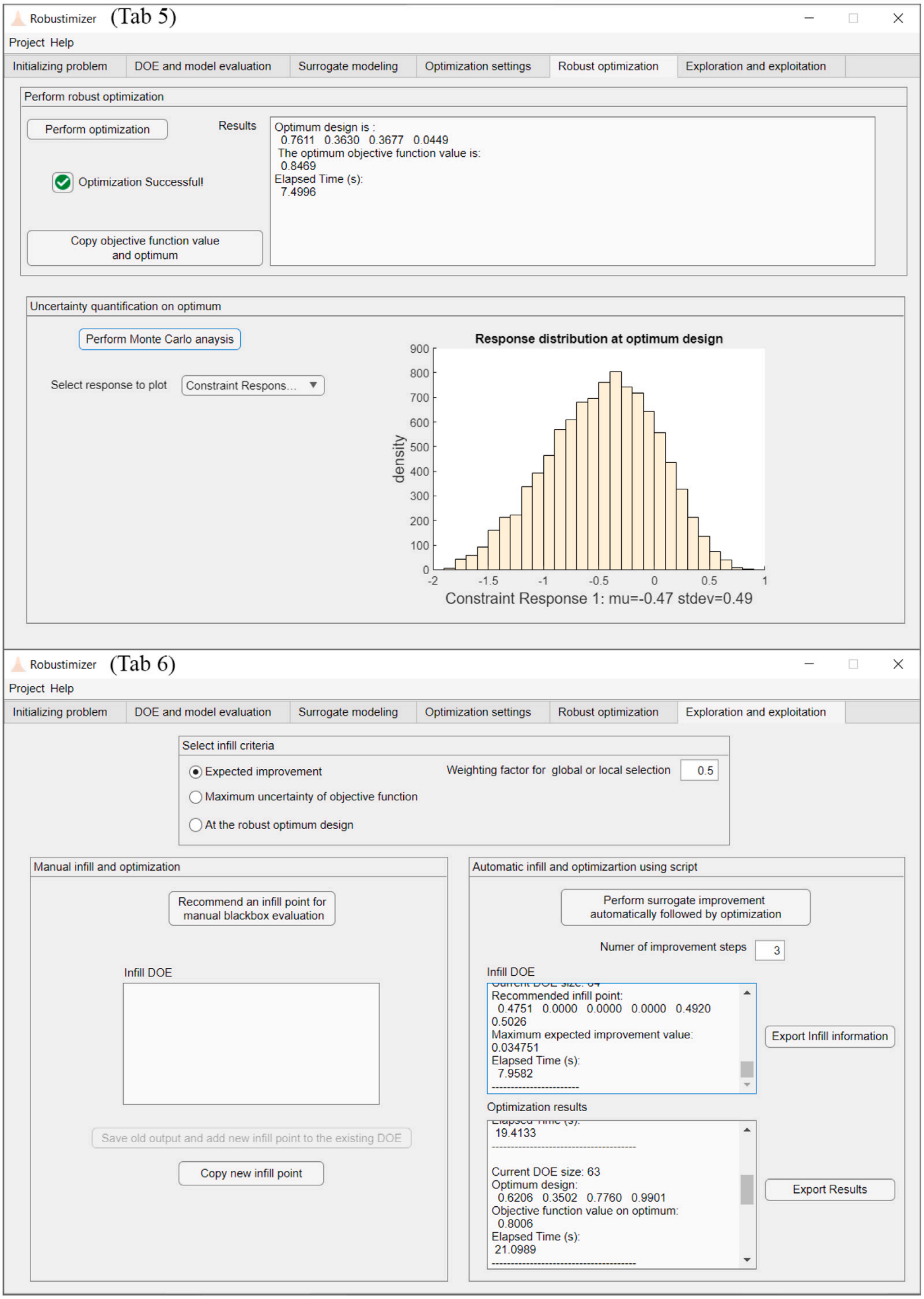

Fig. 5. Illustrative example using Robustimizer (Tabs 5 and 6).

4. Impact

Robustimizer offers a new tool for studying stochastic systems and processes in many disciplines. It has demonstrated its capability to reduce manufacturing waste and contribute to sustainable, zero-defect manufacturing, not only in conventional forming processes but also in additive manufacturing [6,16]. However, its application is not limited to manufacturing. Its capabilities extend to fields such as biotechnology, environmental sustainability, and health informatics, among others [17–21]. Investigating how different types of uncertainties affect optimization strategies in these fields could open new horizons in both science and technology.

The software significantly advances the pursuit of existing research questions by offering more efficient and accurate algorithms [1] for solving optimization problems under uncertainty while providing a user-friendly interface that simplifies experimentation with different scenarios and parameters. Robustimizer enables detailed analysis and evaluation of optimization results to assess their robustness and reliability. Integration with user-defined data and communication with external software help researchers incorporate the software into current

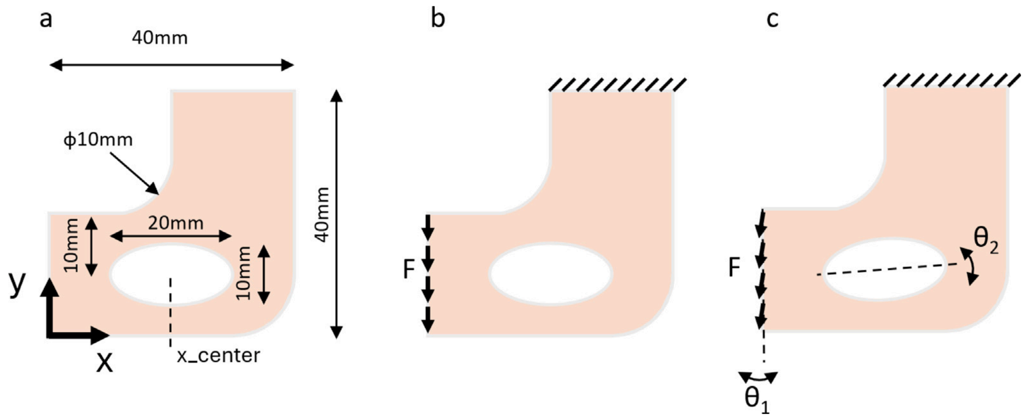

Fig. 6. Dimensions of the bracket (a), the nominal (deterministic) condition (b), and a scenario in which loading and orientation of the hole vary slightly from the nominal case (c).

projects and expand their research.

Additionally, it is important to highlight that there are currently no easy-to-use tools that offer comprehensive and standardized methods for conducting optimization under uncertainty. This gap underscores the distinctive value of Robustimizer, which enables users to perform robust optimization more effectively and efficiently.

Despite its many advanced features, there is still significant potential for expanding Robustimizer. Future developments will focus on incorporating other types of surrogate models such as radial basis function networks, and neural networks [22,23], facilitating integration with existing simulation packages, enabling inverse robust optimization [24], and expanding methods for exploration and exploitation.

5. Conclusions

In this work Robustimizer, a general-purpose application for optimization under uncertainty, is introduced. The major features of Robustimizer are communicating with external software through user-defined executable, efficient and accurate uncertainty quantification using analytical methods for both robust optimization and reliability calculations, and automated updating of the surrogate model. Moreover, it is capable of accounting for higher-order moments of both the objective function and constraints to achieve more accurate measures for robustness and reliability. Robustimizer also considers correlations between noise variables and accepts tailored inputs from users for creating DOE or conducting Monte Carlo analysis. These functionalities are discussed, and two examples are presented to demonstrate its capabilities in solving problems involving uncertainty.

Declaration of competing interest

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.

Acknowledgments

The author would like to greatly thank Xander Burgerhout for providing invaluable insights, thoughtful feedback, and exceptional programming support.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2025.102077.

References

[1] Nejadseyfi O, Geijselaers HJM, van den Boogaard AH. Robust optimization based on analytical evaluation of uncertainty propagation. Eng Optim 2018;51(9):1581–603.
[2] Goh J, Sim M. Robust optimization made easy with ROME. Oper Res 2011;59(4):973–85.
[3] Chen Z, Xiong P. RSOME in Python: An open-source package for robust stochastic optimization made easy. INFORMS J Comput 2023;35(4):717–24.
[4] Wiebe J, Misener R. Romodel: modeling robust optimization problems in pyomo. Optim Eng 2022;23(4):1873–94.
[5] Dellino G, Kleijnen JP, Meloni C. Robust optimization in simulation: Taguchi and response surface methodology. Int J Prod Econ 2010;125(1):52–9.
[6] Nejadseyfi O, Geijselaers H, Atzema E, Abspoel M, van den Boogaard A. Accounting for non-normal distribution of input variables and their correlations in robust optimization. Optim Eng 2022;23(4):1803–29.
[7] Martin RC. Clean architecture:A craftsman's guide to software structure and design. 2017.
[8] Wiebenga J, Atzema E, An Y, Vegter H, van den Boogaard AH. Effect of material scatter on the plastic behavior and stretchability in sheet metal forming. J Mater Process Technol 2014;214(2):238–52.
[9] Lophaven SN, Nielsen HB, Søndergaard J, et al. DACE: a Matlab kriging toolbox. vol. 2, Citeseer; 2002.
[10] Nejadseyfi O, Geijselaers HJ, van den Boogaard AH. Evaluation and assessment of non-normal output during robust optimization. Struct Multidiscip Optim 2019;59:2063–76.
[11] Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. J Global Optim 1998;13:455–92.
[12] ur Rehman S, Langelaar M, van Keulen F. Efficient kriging-based robust optimization of unconstrained problems. J Comput Sci 2014;5(6):872–81.
[13] Koch PN, Yang R-J, Gu L. Design for six sigma through robust optimization. Struct Multidiscip Optim 2004;26:235–48.
[14] Fang J, Sun G, Qiu N, Kim NH, Li Q. On design optimization for structural crashworthiness and its state of the art. Struct Multidiscip Optim 2017;55:1091–119.
[15] Meng D, Yang H, Yang S, Zhang Y, Jesus AMD, Correia J, Fazeres-Ferradosa T, Macek W, Branco R, Zhu S-P. Kriging-assisted hybrid reliability design and optimization of offshore wind turbine support structure based on a portfolio allocation strategy. Ocean Eng 2024;295:116842.
[16] Nejadseyfi O. Sustainable manufacturing via robust optimization and tailored scatter. In: 9th European congress on computational methods in applied sciences and engineering. 2024.
[17] Meng D, Yang S, De Jesus AM, Fazeres-Ferradosa T, Zhu S-P. A novel hybrid adaptive kriging and water cycle algorithm for reliability-based design and optimization strategy: Application in offshore wind turbine monopile. Comput Methods Appl Mech Engrg 2023;412:116083.
[18] Meng D, Yang S, de Jesus AM, Zhu S-P. A novel kriging-model-assisted reliability-based multidisciplinary design optimization strategy and its application in the offshore wind turbine tower. Renew Energy 2023;203:407–20.
[19] Darvazeh SS, Mooseloo FM, Gholian-Jouybari F, Amiri M, Bonakdari H, Hajiaghaei-Keshteli M. Data-driven robust optimization to design an integrated sustainable forest biomass-to-electricity network under disjunctive uncertainties. Appl Energy 2024;356:122404.
[20] Zhang X. A review of the robust optimization process and advances with Monte Carlo in the proton therapy management of head and neck tumors. Int J Part Ther 2021;8(1):14–24.

[21] Pham T, Hoang T, Tran X, Fetni S, Duchêne L, Tran HS, Habraken A. A framework for the robust optimization under uncertainty in additive manufacturing. J Manuf Process 2023;103:53–63.
[22] Alizadeh R, Allen JK, Mistree F. Managing computational complexity using surrogate models: a critical review. Res Eng Des 2020;31(3):275–98.
[23] Yang B, Cheng C, Wang X, Bai S, Long K. Robust reliability-based topology optimization for stress-constrained continuum structures using polynomial chaos expansion. Struct Multidiscip Optim 2023;66(4):88.
[24] Nejadseyfi O, Geijselaers H, Atzema E, Abspoel M, Van Den Boogaard A. From specified product tolerance to acceptable material and process scatter: an inverse robust optimization approach. Int J Mater Form 2020;13:467–78.