Active-Learning Reliability Analysis of Automotive Structures Based on Multi-Software Interaction in the MATLAB Environment

Junfeng Wang 1,2, Jiqing Chen 1,2, Yuqi Zhang 1,2, Fengchong Lan 1,2 and Yunjiao Zhou 1,2,*

1 School of Mechanical & Automotive Engineering, South China University of Technology, Guangzhou 510641, China; w920133787@outlook.com (J.W.); chjq@scut.edu.cn (J.C.); mezhangyuqi@mail.scut.edu.cn (Y.Z.); fclan@scut.edu.cn (F.L.)
2 Guangdong Provincial Key Laboratory of Automotive Engineering, Guangzhou 510641, China
* Correspondence: mezhouyj@scut.edu.cn

Abstract: The reliability design of automotive structures is characterized by numerous variables and implicit responses. The traditional design of experiments for metamodel construction often requires manual adjustment of model parameters and extensive finite element analysis, resulting in inefficiency. To address these issues, active learning-based reliability methods are effective solutions. This study proposes an active-learning reliability analysis method based on multi-software interaction. Firstly, through secondary development of different software and MATLAB (version 2023a)'s batch processing function, a multi-software interactive reliability analysis method is developed, achieving automation in structural parametric design, finite element analysis and post-processing. This provides a more efficient and convenient platform for the implementation of active learning. Secondly, the polynomial chaos–kriging (PCK) active-learning method is introduced, combining the advantages of polynomial chaos expansion (PCE) and kriging. The PCK method captures the global behavior of the computational model using regression-based PCE and local variations using interpolation-based kriging. This metamodel is constructed with fewer training samples, effectively replacing the real multi-dimensional implicit response relations, thereby improving the efficiency of modeling and reliability analysis. Finally, the specific implementation scheme is detailed. The accuracy and efficiency of the proposed method are verified by a reliability engineering example of body-in-white bending and torsional stiffness.

Keywords: reliability analysis; active learning; polynomial chaos–Kriging; multi-software interaction; multi-dimensional implicit response

1. Introduction

Due to the continuous development of automotive technology, structural systems are gradually being integrated. Uncertainty issues have also become increasingly complex and diverse, involving multiple covariates and implicit responses. A common approach is to establish a surrogate model through the design of experiments (DOE) for reliability analysis and optimal design.

To improve efficiency, on the one hand, it is necessary to establish an automated analysis platform to achieve tasks such as model parameterization, solving and post-processing, thereby addressing the problem of inefficiency associated with traditional manual operation. Currently, a variety of software is available for parametric design. For the study of some microscopic stochastic structures, Chen et al. [1] reconstructed the catalyst layer of a proton exchange membrane fuel cell (including non-precious metal catalyst or platinum) with a high-resolution porous structure. Liu et al. [2] parameterized random reconstruction of the fuel cell gas diffusion layer by an ANSYS-SpaceClaim script, improving the efficiency of the repetitive operations. Additionally, for larger structures...
with more fixed shape characteristics, scholars have conducted research on optimization techniques for structural parameters. Mogullapally et al. [3] employed the HyperStudy (version 2017.2.0) tool and the ANSYS (version 18.1) solver to optimize the weight of the leaf disc. Zhao et al. [4] proposed a parametric modeling method based on the secondary development of Abaqus and designed a plug-in tool for cell structure simulation analysis through a graphical user interface. Wu et al. [5] used SolidWorks and ANSYS Workbench software for secondary development and compiled parametric design software using C# to integrate 3D model design, engineering drawing generation and finite element safety analysis. In addition, LS-OPT (version 6.0) [6] and Isight [7] are also often used to build optimization platforms for structural performance analysis and lightweight design. It is evident that the current commonly used software has parametric design and optimization capabilities. In practical application, suitable methods and software should be selected according to the design requirements.

On the other hand, metamodeling methods are used for reliability analysis. For complex engineering problems, there may be complex nonlinear relationships between the design variables and the responses, which are not explicitly expressed and are black-box problems, or the performance (limit state) functions involve expensive computational problems. Therefore, metamodeling methods are used in practical engineering, and the basic idea is to construct approximate (surrogate) models, i.e., to establish approximate mapping relationships between input variables and output responses by DOE, which play a role in computationally efficient and predictable responses at unknown points. Commonly used surrogate models include the response surface model [8], polynomial chaos expansions (PCEs) [9], kriging [10,11], polynomial chaos–kriging (PCK) [12], machine learning [13,14], support vector machines [15,16] and radial-basis functions [17]. Once the surrogate model has been constructed, reliability methods can be used to calculate the failure probability.

In order to reduce the training sample size, active-learning metamodeling methods have been proposed, such as the active-learning reliability method, which combines kriging and Monte Carlo simulation (AK–MC) [18]; the candidate sample set is a large number of samples generated by the MC method, and the evaluation of the learning function occurs in each loop, which affects the computational efficiency. Therefore, some scholars have replaced MC methods with importance sampling (IS) [19], subset simulation (SS) [20] and directional sampling (DS) [21] to form more efficient kriging-based active-learning methods. PCK can also be constructed as an active-learning metamodel [22], which combines the strengths of PCE and kriging, where PCE provides an efficient method for dealing with variability and uncertainty in input parameters, and kriging provides flexibility for modeling complex nonlinear relationships. Therefore, PCK is used in this study to construct an active-learning metamodel.

From the above two aspects, this study proposes an active-learning reliability analysis method based on multi-software interaction, illustrated by conducting bending and torsional stiffness reliability analysis of a body-in-white (BIW) structure. Firstly, the finite element model (FEM) of the BIW is introduced, including design variables, constraints and loads. Secondly, based on the characteristics of the model, a multi-software interactive analysis platform is built in the MATLAB environment, and automated DOE is implemented through secondary development. Finally, the active learning PCK method is introduced to adaptively construct surrogate models and perform the reliability analysis, aiming to improve the efficiency and accuracy of metamodeling and reliability analysis.

2. Bending and Torsional Stiffness Analysis Model of BIW

The bending and torsional stiffness of the BIW have an important impact on vehicle performance, such as NVH (noise, vibration, and harshness), safety, handling performance and stability. They also affect the degree of lightweighting of the body, which are key parameters in the development process and represent the core competitiveness of the vehicle. The model is from ref. [23], and the pre-processing process of CAE meshing,
material properties and constraint boundaries of the model is performed using Hypermesh (version 2019) software, which has about 390,000 cells. The thicknesses of 14 key sheet metal components are considered as the design variables. The FEM and component numbers are shown in Figure 1. The paired components in Figure 1 are annotated on only one side, and the initial values and ranges of the design variables are shown in Table 1.

![Figure 1. Key components of the BIW finite element model.](image1)

**Table 1. Design variables and probability distribution characteristics.**

<table>
<thead>
<tr>
<th>Design Variables</th>
<th>Symbol</th>
<th>Mean</th>
<th>Coefficient of Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-pillar outer plate</td>
<td>$t_1$</td>
<td>1.32</td>
<td>0.03</td>
</tr>
<tr>
<td>Rear seat frame</td>
<td>$t_2$</td>
<td>1.10</td>
<td>0.03</td>
</tr>
<tr>
<td>Front floor</td>
<td>$t_3$</td>
<td>1.62</td>
<td>0.03</td>
</tr>
<tr>
<td>Rear floor</td>
<td>$t_4$</td>
<td>1.08</td>
<td>0.03</td>
</tr>
<tr>
<td>Firewall</td>
<td>$t_5$</td>
<td>0.76</td>
<td>0.03</td>
</tr>
<tr>
<td>Rear cross member</td>
<td>$t_6$</td>
<td>1.26</td>
<td>0.03</td>
</tr>
<tr>
<td>Wheel well-R$^1$</td>
<td>$t_7$</td>
<td>0.88</td>
<td>0.03</td>
</tr>
<tr>
<td>Wheel well-L$^2$</td>
<td>$t_8$</td>
<td>0.88</td>
<td>0.03</td>
</tr>
<tr>
<td>Rail-L</td>
<td>$t_9$</td>
<td>1.71</td>
<td>0.03</td>
</tr>
<tr>
<td>Rail-R</td>
<td>$t_{10}$</td>
<td>1.71</td>
<td>0.03</td>
</tr>
<tr>
<td>Rear rail-L</td>
<td>$t_{11}$</td>
<td>1.65</td>
<td>0.03</td>
</tr>
<tr>
<td>Rear rail-R</td>
<td>$t_{12}$</td>
<td>1.65</td>
<td>0.03</td>
</tr>
<tr>
<td>Unibody frame-L</td>
<td>$t_{13}$</td>
<td>0.99</td>
<td>0.03</td>
</tr>
<tr>
<td>Unibody frame-R</td>
<td>$t_{14}$</td>
<td>1.71</td>
<td>0.03</td>
</tr>
</tbody>
</table>

$^1$L means left. $^2$ R means right. (Unit: mm).

The loads and constraints for the bending and torsion cases are shown in Figure 2a and Figure 2b, respectively, defining the vertical (gravitational) direction of the model as the $Z$-direction. The forces $F_{b1}$, $F_{b2}$ applied to the sill beams and $F_{t1}$, $F_{t2}$ applied to the shock towers are $Z$-directed forces (N): $F_{b1} = F_{b2} = 3336\text{N}$ and $F_{t1} = F_{t2} = 1177\text{N}$. For bending conditions, the four restraint positions (SPC1-SPC4) are located on the shock towers; for torsion conditions, the front restraint position (SPC1) is located in the middle of the front bumper frame, and the rear restraint positions (SPC2, SPC3) are located on the antiroll bar brackets.

![Figure 2. (a) Load collectors of bending; (b) Load collectors of torsion.](image2)
The locations of the observation points are illustrated in Figure 3. Under the bending condition, the maximum deformation (mm) in the Z-direction of the nodes below the left and right sill beams (six observation points are found uniformly on each side) are set to be $d_{L\text{ max}}$ and $d_{R\text{ max}}$, respectively, and, under the torsion condition, the Z-displacements (mm) at the left and right loading points of the shock towers are $d_L$ and $d_R$, respectively. Setting $d$ as the distance (mm) between the loading points of the shock towers, the equations of bending stiffness $K_b$ and torsional stiffness $K_t$ can be obtained:

$$
\begin{align*}
K_b &= \frac{F_b}{(|d_{L\text{ max}}| + |d_{R\text{ max}}|)/2} \quad (\text{N/mm}) \\
K_t &= \frac{F_t d}{1000 \arctan(|d_L - d_R|/d)} \cdot \frac{\pi}{180} \quad (\text{N\cdot m/°})
\end{align*}
$$

(1)

where $F_b = F_{b1} + F_{b2}$ and $F_t = F_{t1} = F_{t2}$.

![Figure 3. Schematic of observation points for bending and torsion conditions.](image)

3. MATLAB-Based Multi-Software Interactive Process

3.1. Batch Methods for MATLAB

Since the responses of bending and torsional stiffness are implicitly related to the thickness variables of key components, one efficient method is to build parametric structural models and to construct an automated analysis platform for DOE. Isight provides visual interfaces and tools to create complex engineering designs that are highly automated through custom components. However, there are some shortcomings in practical application; for example, when building multiple custom functions, the conversion and transmission of data will become complicated. If it is run only in a certain programming software, the operation is more flexible and can ensure data synchronization and consistency. For this reason, this study further summarizes the implementation and features of Isight, as well as two commonly used programming software options—MATLAB (version 2023a) and Python (version 3.11)—for batch processing, as shown in Table 2.
Table 2. Batch processing methods and features.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Execution Function</th>
<th>Execution Method</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isight</td>
<td><strong>“simcode” component</strong></td>
<td>Blocking Execution; Non-blocking</td>
<td>Integrates models and tools from different types of software into a unified optimization platform; offers a wealth of parameter optimization algorithms and tools; provides visual interfaces and tools that are easy to operate.</td>
</tr>
<tr>
<td>MATLAB</td>
<td><code>system()</code></td>
<td>Blocking Execution</td>
<td>Used to execute simple external commands and return the exit status code of the command. Provides more flexibility and control options to capture output, handle errors, set timeouts, etc. Suitable for Python 3.5 and above.</td>
</tr>
<tr>
<td>Python</td>
<td><code>os.system()</code></td>
<td>Blocking Execution</td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td><code>subprocess.run()</code></td>
<td>Blocking Execution</td>
<td>Ideal for situations where more advanced process-management functions are required, such as interacting with sub-processes, controlling input and output streams and implementing more complex process communications.</td>
</tr>
<tr>
<td>Python</td>
<td><code>subprocess.Popen()</code></td>
<td>Non-blocking, Call <code>process.wait()</code> to block execution</td>
<td></td>
</tr>
</tbody>
</table>

In terms of the execution mode, blocking execution means that, when an operation is called, the program will wait for the completion of the operation before continuing to execute the subsequent code, while, in non-blocking execution mode, the program will not wait for the completion of the operation and will continue to execute the subsequent code. Generally, it is necessary to call the results of batch execution by external software, so blocking execution is used.

The functions in Table 2 have different characteristics. The `system()` and `os.system()` function are suitable for simple external command execution and return the exit status code of the command, but cannot output or handle errors. The `subprocess.run()` and `subprocess.Popen()` function, on the other hand, provide richer functionality. The purpose of this study is to implement batch processing in blocking execution mode, so all of the above functions meet the general requirements.

MATLAB is widely applied in academia due to its powerful arithmetic and programming features in handling matrices and numerical computations, providing rich engineering toolboxes and an interactive development environment. Many uncertainty studies and open-source codes have been implemented based on MATLAB. For example, DACE [24] and ooDACE [25] are commonly used open-source toolboxes for kriging modeling, and a generic MATLAB-based uncertainty quantification framework, UQLab [22], has been developed at ETH Zurich, which includes modules for MC method simulation, sensitivity analysis, reliability analysis, active-learning surrogate modeling and Bayesian inversion. These toolboxes and the framework offer significant inspiration and facilitate programming implementations for uncertainty studies. Therefore, MATLAB is chosen to implement batch processing in this study.

Additionally, using MATLAB for batch processing enhances convenience in function calls and data transfers. Data saving and extraction can be achieved by navigating through different folders. Storing models with varying parameters in distinct directories allows for dynamic changes in the working path, which effectively prevents file overwriting during iterations. This approach greatly improves the flexibility and efficiency of model management.

### 3.2. Automate Tasks with Scripts

The purpose of batch processing is to automate the execution of different software to complete a series of tasks in modeling, pre-processing, finite element analysis and post-processing. In this study, ANSA (version 23.1.1) software is used to implement model parameterization and post processing, which is a professional finite element pre-processing software that provides powerful parameterization functions. The features of ANSA are as follows:
(1) It is easy to modify the model by defining parameters and to quickly generate the models of different variants, avoiding the errors or inconsistencies that may be introduced by manual modification;

(2) It integrates a variety of engineering design and simulation software options, such as a finite element solver, optimization software, etc., which provides convenience in model parameter transfer;

(3) It supports the combination of script and batch commands, which achieves the automated model modification, improving work efficiency and reducing repetitive work;

(4) META module is an important part of ANSA software for model post-processing, results analysis and report generation, which also supports batch operation to automate the analysis and processing of multiple results.

This study uses the 'system()' function in MATLAB to execute the batch command file (.bat). For example, in the case of ANSA batch processing, the MATLAB code can be written as “status = system('ANSA.bat')”, where “status” returns the execution status, 0 denotes successful execution, while other values indicate encountering errors or issues. Throughout the analysis process, ANSA, OptiStruct and META are executed sequentially in batch order, corresponding, respectively, to preprocessing, solving and post-processing.

The ANSA pre-processing requires the execution of a script (.script.ansa), which can be created through the “Task Manager” of ANSA. Select “Optimisation Task” in “Task Manager” to set up the pre-processing procedure, as shown in Figure 4a, including input parameter file (DVFile.txt), design variables and output model (BIW.fem), and then save the current operation as execution script (.script.ansa), which automatically assigns the parameters of DVFile.txt to the design variables of the model and exports the modified model (BIW.fem).

The modified model BIW.fem is solved using Optistruct by simply running the batch file (.bat). In post-processing, the displacements of the observation points are extracted using META. In the bending condition, the maximum deformations of the left and right side sill beams are extracted, and, since the Z-direction displacements are negative, the smallest (most negative) values are taken as the maximum deformations at the observation points, as shown in Figure 4b. In the torsion condition, the Z-direction displacements of the two front shock towers are extracted, as shown in Figure 4c. After the post-processing setup is completed, the execution script file (.ses) and the response file (.ses.results) are exported. Finally, the response results are extracted from the response file (.ses.results) through MATLAB, and the bending and torsional stiffness are calculated through Equation (1).
3.3. DOE Analysis

The DOE process and file interactions are shown in Figure 5. The design variables and their respective value ranges are presented in Table 1. MATLAB is used to generate random samples and modify the DVFile.txt sequentially. Next, ANSA modifies the model parameters according to the DVFile.txt. The rest of the process is the same as the previous subsection and is not repeated.

![Figure 5. DOE flowchart for BIW.](image)

BIW involves 14 variables; thus, each sample dimension is 14. In order to make the sample distribution more uniform, this study adopts the Latin hypercube method for sampling, and the initial sample size is set to 20. Figure 6 displays matrix plots of the five variables from $t_1$ to $t_5$; the diagonal histograms show that the samples are evenly distributed, and the remaining scatter plots show that the samples are distributed in the entire design space without evidence of multi-sample clustering.
When the DOE is completed, the main effect values of the 14 design variables on the bending and torsional stiffness responses ($K_b$ and $K_t$) are calculated and normalized. The results are shown in Figure 7: each variable exhibits a different degree of influence on $K_b$ and $K_t$. Taken together, it is observed that only $t_2$ has a smaller effect on both responses, but, for $K_t$, $t_2$ has more influence than $t_6$. Therefore, all 14 variables are considered in the next step of reliability analysis.

![Figure 7. Initial plotmatrix using Latin hypercube sampling.](image)

**4. Building an Active-Learning-Based Reliability Analysis Platform**

**4.1. PCK-Based Active-Learning Method**

The small initial DOE sample size makes it challenging to establish accurate $K_b$ and $K_t$ surrogate models. The output responses of each sample need to be solved by calling the FEM. In order to reduce the number of new samples in the DOE, this study further constructs a reliability analysis platform based on active learning. The purpose of active learning is to establish locally accurate limit state function surrogate models through iteration, thereby reducing the sample size and improving efficiency.
The process of active-learning-based reliability analysis and the methods available for each step are shown in Figure 8. The first step is DOE analysis, which has been carried out in the previous section. Further, the metamodels (surrogate models) are built. The next reliability analysis based on the metamodels can be performed by MC, IS, SS, etc.; in this study, the MC method is used for reliability analysis, and the sample size is chosen to be $10^6$. Then, it is determined whether convergence occurs. If yes, modeling is completed; otherwise, the active learning function is used to find the next optimal point, which is added to the DOE sample set and the metamodel is updated until the convergence condition is met.

![Figure 8. Active-learning reliability analysis process and available methods.](image)

The sample generation method for the initial DOE can be chosen according to the actual requirements. If the simulation is time-consuming and too many samples are not suitable, LHS sampling can be used. If a large sample size is necessary for the analysis, the MC method can be used. Sobol sampling has the advantage of being able to provide a global sensitivity index, which is suitable for exploring the complex relationship between inputs and outputs. Metamodels used for active learning include kriging, PCK and SVR. In this study, PCK is used to build the metamodel, which combines the strengths of PCE and kriging. The kriging method is a stochastic interpolation algorithm in which the estimated function value is a linear combination of a regression model and a stochastic process. Its general form is [26]

$$g_K(x) = f^T(x)\beta + \sigma^2 Z(x, \omega)$$ (2)

where $f^T(x)\beta$ denotes the regression model, $f(x) = [f_1(x), f_2(x), ..., f_n(x)]^T$ denotes the vector of regression basis functions, $\beta = [\beta_1, \beta_2, ..., \beta_n]^T$ is the vector of regression coefficients and $Z(x, \omega)$ is a zero mean, unit variance, stationary Gaussian process, characterized by the correlation function $R$ and its hyperparameter $\theta$.

Consider a random vector with independent components $x \in \mathbb{R}^n$ described by the joint probability density function $f_X$. PCE approximates the output of the computational model with a sum of orthogonal polynomials [27]:

$$\hat{g}_{PCE}(x) = \sum_{a \in A} y_a \Psi_a(x)$$ (3)
where $\Psi_\alpha(x)$ are multivariate polynomials that are orthonormal with respect to the input distributions, $\alpha \in A \subset \mathbb{N}^n$ are multi-indices and $y_\alpha$ are the corresponding coefficients.

PCK uses regression-based PCE to capture the global behaviour of the computational model and interpolation-based kriging to capture local variations, and its metamodeling technique is more effective than the metamodeling technique produced by PCE and kriging separately, which is expressed as [28]

$$g_{PCK}(x) = \sum_{\alpha \in A} y_\alpha \Psi_\alpha(x) + \sigma^2 Z(x, \omega)$$  \hspace{1cm} (4)

The active-learning function utilizes the $U$-function proposed by AK–MC [18]:

$$U(x) = \left| \frac{\hat{g}(x)}{\sigma_{\hat{g}}(x)} \right|$$  \hspace{1cm} (5)

where $\hat{g}(x)$ and $\sigma_{\hat{g}}(x)$ are the mean and standard deviation obtained from kriging estimation. The criterion for convergence is the learning function (LF)-based criterion that needs to be satisfied:

$$\min U(x) > \epsilon_U$$  \hspace{1cm} (6)

This means that the probability of any point in the candidate pool being misclassified is less than $\Phi(-\epsilon_U)$. In general, $\epsilon_U = 2$ is acceptable. The convergence criteria also include variance-based criteria (bounds on $P_f$, bounds on $\beta$) and stability-based criteria (stability on $P_t$, stability on $\beta$), where $P_t$ and $\beta$ denote the failure probability and the reliability index. Taking bounds on $P_t$ as an example, the expression is as follows:

$$\frac{\hat{P}_t^-}{\hat{P}_t^+} \leq \epsilon_{bound}$$  \hspace{1cm} (7)

where the three failure probabilities are defined as follows:

$$\begin{cases}
\hat{P}_t^0 = P(\hat{g}(x) \leq 0) \\
\hat{P}_t^- = P(\hat{g}(x) \mp k\sigma_{\hat{g}}(x) \leq 0)
\end{cases}$$  \hspace{1cm} (8)

where $k = \Phi^{-1}(1 - \alpha/2)$; the confidence interval $1 - \alpha/2 = 97.5\%$ is usually set, yielding $k = 1.96$. In this study, active-learning applications are developed with these two convergence criteria. For more details on other convergence criteria and learning functions in Figure 8, please refer to ref. [22].

4.2. Active-Learning Platform Construction and Reliability Analysis

According to the previous subsection, active learning is an iterative process, where each iteration necessitates invoking the PCK metamodel and the learning function, and then solving the FEM. Therefore, the PCK metamodel function and the learning function are established in MATLAB, which are convenient to be called at any time. Iterations are facilitated by the “while” function. The active learning process with multi-software interaction is shown in Algorithm 1. The sample size in step 2 can be set to $10^6$; steps 6–8 are automated batch processing of multi-software, where step 6 is pre-processing using ANSA to modify the model parameters to $X_{opt}$, step 7 is solving using OptiStruct and step 8 is post-processing using META to export the response file. In step 13, the reliability analysis is performed based on the PCK metamodel. If the LF-based convergence criterion is used, the failure probability can be calculated after the metamodel update is complete.

Algorithm 1 can be used as an active-learning reliability analysis platform, which is used to analyze the reliability of the bending and torsional stiffness of the BIW. The iterative process of the LF-based (U-learning function) convergence condition is illustrated in Figure 9. The initial DOE sample size is 20. Initially, the metamodels for bending
and torsional stiffness are established. The error in the torsional stiffness metamodel is statistically found to be larger than in the bending stiffness metamodel, so the active-learning reliability analysis is firstly carried out for torsional stiffness, and the convergence condition is reached after iterating nine samples. Subsequently, 29 samples are used as initial samples to analyze the bending stiffness. After iterating through 2 additional samples, convergence is achieved, totaling 11 new samples used in the process.

**Algorithm 1** Active-learning reliability analysis algorithms for multi-software interaction.

**Input:** The initial DOE samples $X_{\text{init}}$, and the corresponding response $Y_{\text{init}}$

**Output:** PCK metamodel; failure probability $P$

1. Build the initial PCK metamodel using $X_{\text{init}}$ and $Y_{\text{init}}$
2. Generate samples $X$ using the MC method and compute $U(X)$
3. $X_{\text{all}} \leftarrow X_{\text{init}}$; $Y_{\text{all}} \leftarrow Y_{\text{init}}$
4. **while** The convergence criterion is not satisfied **do**
5. Select the next optimal sample $X_{\text{opt}}$
6. system('ANSA.bat') ▷ Pre-processing
7. system('OptiStruct.bat') ▷ Solving
8. system('META.bat') ▷ Post-processing
9. Use MATLAB to extract the value of $Y_{\text{opt}}$
10. $X_{\text{all}} \leftarrow [X_{\text{all}}; X_{\text{opt}}]$; $Y_{\text{all}} \leftarrow [Y_{\text{all}}; Y_{\text{opt}}]$
11. Updating PCK metamodel using $X_{\text{all}}$ and $Y_{\text{all}}$
12. Perform step 2 to generate new samples $X$ and compute $U(X)$
13. Calculate $P_{f}^0$, $P_{f}^+$ and $P_{f}^-$
14. **end while**
15. PCK metamodel is obtained, and failure probability $P = P_{f}^0$

![Figure 9](image.png)

**Figure 9.** Iterative process of LF-based convergence criterion: (a) Active-learning iterative process for torsional stiffness; (b) Active-learning iterative process for bending stiffness.

The iterative process based on bounds on $P_{f}$ is shown in Figure 10. The initial sample size is still 20; after adding 47 new samples, the metamodel of torsional stiffness reaches convergence, and, using these 67 samples as the initial samples to update the metamodel of bending stiffness, as can be seen from Figure 10b, the accurate metamodel is established with only 1 new sample.
The U-learning function is used as the convergence criterion and is not constrained by variance-based criteria (bounds on $P_f$). As a result, although the error interval of the failure probability does not converge effectively, it remains within approximately 10%, meeting the general accuracy requirement and reducing the number of iterations.

Without the application of active learning, HyperStudy suggests conducting a DOE with 132 samples. Following the DOE, response surfaces are constructed. The $R^2$ (coefficient of determination) for bending and torsional stiffnesses are 0.9998 and 0.9997, respectively, indicating high accuracy. Reliability analysis is performed using the MC method, and the average of the 20 simulations is used as the benchmark value for the failure probability of each response. The results are shown in Table 3. The failure probability of the bending stiffness $P_b$ is $1.53 \times 10^{-4}$, and the failure probability of the torsional stiffness $P_t$ is $3.33 \times 10^{-4}$. The results of the PCK metamodel are close to these, which verifies the accuracy and efficiency of the scheme in this paper. Minimizing the number of training samples through active learning avoids the waste caused by too many samples, and also avoids the possible accuracy degradation caused by an insufficient number of samples. However, the sample size required may vary relatively widely with different convergence criteria, and the appropriate convergence criterion needs to be selected by weighing accuracy and efficiency in practical applications.

<table>
<thead>
<tr>
<th>Method</th>
<th>Convergence Criterion</th>
<th>Sample Size</th>
<th>$P_b$</th>
<th>$P_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active learning</td>
<td>LF-based</td>
<td>31</td>
<td>$1.51 \times 10^{-4}$</td>
<td>$3.25 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>Bounds on $P_f$</td>
<td>68</td>
<td>$1.65 \times 10^{-4}$</td>
<td>$3.36 \times 10^{-4}$</td>
</tr>
<tr>
<td>Response surface</td>
<td>–</td>
<td>132</td>
<td>$1.53 \times 10^{-4}$</td>
<td>$3.33 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

To further examine the stability of active learning, taking torsional stiffness as an example, the initial sample is set to 30, and the iterative process under different random seeds is shown in Figure 11. It can be observed that different random seeds lead to different initial samples, resulting in variations in the number of iterations and reliability results. Figure 9a and Figure 11a share the same random seed, but, due to different initial sample sizes, they also yield different numbers of iterations and reliability results. Despite these differences, the results generally fall within controllable ranges.
5. Conclusions

(1) This study proposes a multi-software interaction approach for active-learning reliability analysis, using BIW as a case study, to address the multi-dimensional implicit response relationships and time-consuming computational costs. MATLAB programming and secondary development are used to achieve multi-software interaction and automate the execution of pre-processing, solving and post-processing tasks, thereby minimizing manual interventions and enhancing efficiency. Subsequently, the PCK active-learning method is introduced to establish locally accurate surrogate models and reduce the number of DOE training samples. Active learning required 31 samples using the LF-based criterion and 68 samples with the bounds on $P_f$ criterion. The resulting failure probabilities are consistent with those obtained using the response surface method with 132 samples. This significantly reduces the sample size, thereby minimizing computational demands and further improving efficiency, specifically by reducing the number of finite element simulations required. Therefore, this method not only improves the design efficiency, but also provides an interactive modeling framework that is easy to understand and apply, enabling the rapid application of the technology to real-world engineering problems. This is conducive to promoting the innovation and development of automotive structural design, and brings tangible value to the field of engineering design;

(2) The efficiency and accuracy of active learning may be affected by a number of factors, such as the initial sample size, random seed, convergence criterion and degree of nonlinearity. Among these factors, the initial sample size and convergence criterion are easier to control. Therefore, during application, continuous practice is needed to balance the efficiency and accuracy by adjusting the initial sample size and convergence criteria;

(3) In terms of uncertainty quantification, as engineering problems become increasingly complex, future research can further explore and develop in the following three aspects. Firstly, intelligent parameter design needs to be enhanced to improve the stability and validity of models as parameters change. This will simplify the system design process and improve adaptability for various engineering applications. Secondly, capabilities for secondary software development need to be strengthened to improve flexibility and scalability. This will help provide tailored solutions for engineering analysis and meet the different needs of users. Finally, reliability methods need to be continuously refined and improved to address potential challenges and shortcomings for engineering applications, including how to improve efficiency and robustness when addressing multi-dimensional and nonlinear implicit response issues.

Author Contributions: Conceptualization, J.W. and J.C.; methodology, J.W.; software, J.W. and Y.Z. (Yuqi Zhang); validation, J.C. and F.L.; formal analysis, F.L. and J.C.; investigation, J.W. and Y.Z. (Yuqi Zhang); resources, F.L.; data curation, Y.Z. (Yunjiao Zhou); writing—original draft preparation, J.W;
writing—review and editing, J.C. and F.L.; visualization, J.W.; supervision, J.C. and Y.Z. (Yunjiao Zhou); project administration, Y.Z. (Yunjiao Zhou) and F.L.; funding acquisition, Y.Z. (Yunjiao Zhou)

All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China (Grant No. 52175267), Science and Technology Planning Project of Guangzhou City of China (Grant No. 202007020007) and Natural Science Foundation of Guangdong Province of China (Grant No. 2021A15150912).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy

**Conflicts of Interest:** The authors declare no conflicts of interest.

**References**


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.