OPEN ACCESS

**Saudi Journal of Civil Engineering**

Abbreviated Key Title: Saudi J Civ Eng
ISSN 2523-2657 (Print) | ISSN 2523-2231 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: https://saudijournals.com/journal/sjce/home

**Original Research Article**

# A Hybrid Neural Network-Kriging Ensemble Framework for Efficient Structural Reliability Analysis

Reza Javanmardi[1]*, Behrouz Ahmadi-Nedushan[1]

[1]Department of Civil Engineering, Yazd University, Yazd, Iran

**DOI:** https://doi.org/10.36348/sjce.2025.v09i09.002 | **Received:** 26.08.2025 | **Accepted:** 15.10.2025 | **Published:** 24.10.2025

***Corresponding author:** Reza Javanmardi
Department of Civil Engineering, Yazd University, Yazd, Iran

## Abstract

In practical engineering systems, accounting for various uncertainties during the design process is paramount. However, reliability analysis in structural engineering often entails substantial computational costs, particularly when dealing with implicit performance functions and scenarios involving very low failure probabilities. This inherent complexity underscores the challenges faced in real-world applications, where efficient and accurate reliability assessments are crucial for ensuring structural integrity and safety. In recent years, the concept of utilizing surrogate models for reliability analysis has garnered significant attention. The approach outlined in this study employs an innovative surrogate framework that concurrently integrates Cascade-forward Neural Networks (CFNN) and Self-Organizing Map (SOM) networks, alongside an optimized kriging model. The final reliability assessment is then determined as a weighted average of the outputs from these integrated models. To comprehensively illustrate the effectiveness of the proposed algorithm, a diverse range of examples are included: five mathematical examples and five engineering examples. Furthermore, a detailed discussion highlights the benefits of this proposed method in comparison to alternative approaches. The results demonstrate the effective performance of the developed methodology. For instance, in the mathematical examples, the minimum improvement observed over other methods is an 81%, coupled with an approximate 0% error in reliability calculation. Similarly, for the engineering examples, a minimum improvement of 47% is achieved over existing methods, with the reliability calculation error remaining low at approximately 1%.

**Keywords:** Time-Consuming Function, SAP2000, Cascade Forward Neural Network, Self-Organizing Map, Kriging, Reliability.



## 1. INTRODUCTION

In practical engineering systems, numerous uncertainties exist regarding boundary conditions, material characteristics, dimensions, analytical models, and applied loads. These factors must be meticulously considered when designing structures within a probabilistic framework. Structural reliability analysis, which assesses the probability of failure using a probabilistic model, offers various methods for this purpose. Notable approaches include Monte Carlo simulation (MCS) (Papadopoulos, Giovanis *et al*., 2012), the first-order reliability method (FORM) (Hasofer and Lind 1974) and the second-order reliability method (SORM) (Lim, Lee *et al*., 2014). The primary challenge in reliability methods lies in accurately estimating the probability of failure through the limit state function. To enhance the efficacy of reliability analysis, FORM and SORM were developed. Their efficiency has been demonstrated through their application in complex optimization problems, such as reliability-based topology optimization (Johari, Ahmadi Nedushan *et al*., 2017). However, these methods can sometimes yield unstable outcomes in certain reliability issues (Keshtegar and Sadegh 2015). In contrast, MCS offers a comprehensive approach for precisely estimating the probability of failure, especially when compared to analytical techniques.

In practice, this method presents considerable computational challenges for structural engineering, especially with implicit performance functions and low failure probabilities. This is largely because it demands numerous simulation sampling points and extensive structural analysis computations (Zhao, Yue *et al*., 2015). Consequently, the computational cost can become prohibitive for implicit limit state probabilistic models that rely on the finite element method for performance

function outputs, potentially leading to unacceptable reliability indices for low failure probability problems. Therefore, standard MCS is unsuitable for evaluating the probability of failure in such implicit limit state functions. In response, surrogate models have garnered significant interest. These models estimate outputs using far less computationally intensive principles than the original model, while preserving adequate accuracy. Initially, a surrogate model is constructed to approximate the performance function. Reliability can then be assessed by analyzing this metamodel. As evaluating the analytical metamodel is substantially faster than the actual performance function, computational efficiency is significantly improved. Furthermore, these models can leverage both CPU and GPU hardware, dramatically accelerating computations. Their effectiveness in complex engineering optimization problems has been recently demonstrated, showing suitable performance for these issues (Jahangiri, Ahmadi-Nedushan et al. 2015, Akbarzaghi, Ahmadi-Nedushan et al., 2024, Ahmadi-Nedushan, Akbarzaghi et al., 2025).The landscape of surrogate models is diverse, featuring approaches such as Support Vector Machines (SVM) (Rocco, Moreno *et al*., 2002, Hurtado 2004, Basudhar, Missoum *et al*., 2008, Bourinet, Deheeger *et al*., 2011), Artificial Neural Networks (ANN) (Hurtado 2004, Deng 2006, Cheng, Zhang *et al*., 2007, Kingston, Rajabalinejad *et al*., 2011, Papadopoulos, Giovanis *et al*., 2012), Kriging (Kaymaz 2005, Bichon, Eldred *et al*., 2008, Echard, Gayton *et al*., 2013, Zhang, Xiao *et al*., 2019), and Response Surface methods (Bucher and Bourgund 1990, Kleiber, Knabel *et al*., 2004, Kaymaz and McMahon 2005, Allaix and Carbone 2011). Within this spectrum, neural networks are particularly prominent, owing to their synergy with deep learning principles that have advanced so rapidly in recent years. Jin Cheng et al. (2008), for example, proposed a novel response surface method employing ANNs and the uniform design method for predicting structural failure probability. Their technique involves selecting training datasets to build an ANN model using uniform design, approximating the limit state function with this trained ANN, and then estimating failure probability via FORM. Jian Deng *et al*., (2005) also combined MCS with FORM and SORM methods alongside ANN models. Complementing these, Elhewy *et al*., (2006) introduced a technique where ANNs establish the crucial link between random variables (inputs) and structural responses, subsequently connecting to a reliability method (FORM, SORM, or MCS) for failure probability estimation. Beyond neural networks, Kriging has also been a fertile ground for reliability assessment. Kaymaz (2005) developed a Kriging-based reliability method, grounded in specific experimental designs, to estimate the Kriging meta-model and subsequently derive the failure probability. Building on this, Bichon *et al*., (2008) devised a strategy for adaptively identifying additional samples to refine experimental designs and enhance Kriging accuracy. Zhang *et al*., (2019) further advanced this with an active learning reliability method (AKEE-SS) that merges kriging with the exploration and exploitation of failure regions, coupled with subset simulation. In their approach, exploration utilizes samples from the initial levels of subset simulation, while exploitation uses samples from the final levels. To effectively manage the impact of meta-model error on failure probability estimates, they developed two error measurement functions that quantify this influence and guide the termination conditions for meta-model updates.

This study introduces a surrogate framework designed to calculate the probability of failure by synergistically integrating the strengths of neural networks and the kriging method. This approach concurrently employs complementary CFNN and SOM alongside an optimized kriging model. The final probability of failure is then determined as a weighted average of the results obtained from these integrated models. Once a suitable surrogate model is established, structural reliability is computed using this framework. Specifically, we utilize Kernel Density Estimation (KDE) (Bowman and Azzalini 1997) and the Weighted Average Simulation Method (WASM) (Rashki, Miri *et al*., 2012) to assess this structural reliability. To demonstrate the efficacy of our proposed algorithm, we present a comprehensive set of examples, including five mathematical and five engineering cases. These examples are accompanied by a detailed discussion highlighting the advantages of our method when compared to existing alternative approaches. This document is structured to guide you through our research systematically.

**Section 2** delves into the meta-models utilized in this study, providing a detailed description of the Complementary CFNN, SOM, and the optimized kriging model.

**Section 3** then examines the techniques employed for assessing the probability of failure, specifically focusing on KDE and the WASM.

**Section 4** introduces our proposed algorithm for determining the failure probability, building upon the meta-models and assessment techniques described earlier.

**Section 5** brings our approach to life with numerical examples, featuring a diverse range of both mathematical and engineering case studies.

Finally, **Section 6** offers a thorough discussion of the research findings, summarizing the outcomes and their implications.

## 2. META MODELS

This section outlines the meta-models utilized in this research, which encompass models CFNN, SOM, and kriging.

In this section, we provide a comprehensive overview of the meta-models central to our research. Specifically, Section 0 covers the complementary CFNN, the SOM, and an optimized kriging model.

### 2.1 Cascade-forward Neural Networks

Neural networks represent a class of computational techniques inspired by the architecture of biological neurons. A neural network is capable of identifying patterns within data through a training process. Among various network architectures, the multilayer feed-forward neural network (FFNN) is a prevalent type, consisting of an input layer, one or more hidden layers, and an output layer. Within an FFNN, neurons are organized into these distinct layers. The outputs from preceding neurons are processed and transformed, adjusting with relevant weights, to become inputs for the subsequent neuron. All aggregated inputs are then passed through an activation function. Neurons in the hidden layers assign weights to the sum of input variables (as depicted in equation (1)), while neurons in the output layer also apply weighting to their aggregated inputs.

$$O_i = f\left(\sum_{j=1}^{m} w_{ij}x_j + \theta_i\right) \quad (1)$$

$$O_k = \varphi\left[\sum_{i=1}^{q} w_{ki}\, f\left(\sum_{j=1}^{M} w_{ij}x_j + \theta_i\right) + a_k\right] \quad (2)$$

Where f and φ are the activation functions of the hidden layer and the output layer, respectively. m and q are the dimensions of the input vector for the input layer and the output layer, respectively. $w_{ij}$ and $w_{ki}$ are the weights of the input layer to the hidden layer and the weights of the hidden layer to the output layer, respectively. $\theta_i$ and $a_k$ are the biases of the hidden layer and the output layer, respectively.

CFNNs bear a resemblance to feed-forward networks. However, they uniquely incorporate a connection from the input and each preceding layer to the subsequent layers. Much like their feed-forward counterparts, a cascade network, comprising two or more layers, is capable of learning any finite input-output relationship to a high degree of accuracy, assuming an adequate number of hidden neurons are present.

### 2.2 Self-Organizing Map Neural Networks

SOM neural networks are employed for the automatic clustering of input data, a process achieved through unsupervised learning. The outcome generated by this network is a distilled representation of the information, typically of a lower dimensionality than the original data, which is aptly termed a 'map.' Beyond the number of neurons utilized, SOMs also feature parameters that define the dimensions of this map. In these models, inputs are connected to all neurons, meaning each neuron possesses a weight vector whose size matches that of the input vector. Crucially, the values of these weights are adjusted during the training process. The fundamental goal of this network architecture is to arrange inputs that are similar to each other in close proximity within the resulting map.

### 2.3 Kriging

The Kriging technique was initially introduced in 1950 by Daniel Krige, primarily for forecasting mineral distribution based on sample points (Krige and Metallurgy 1951). In contrast to linear regression, where the functional form (e.g., polynomial) is typically predetermined and data is used to find coefficients that minimize the root mean square error at training points, Kriging adopts a distinct approach. It operates under the assumption that detailed information about the function is scarce, beyond the correlation between function values at adjacent points (Viana, Haftka *et al*., 2009). This correlation is contingent upon the distance separating the points; as the distance increases, the correlation diminishes. Kriging is characterized as a 'gentle' predictor, ensuring that if two points are in close proximity, their Kriging estimates will also be close (Viana, Haftka *et al*., 2013) Broadly speaking, a Kriging metamodel comprises two key elements: linear regression and a stochastic process (Lophaven, Nielsen *et al*., 2002). Assume that **x** denotes input variables and y(**x**) denotes response, kriging is given as:

$$y(\mathbf{x}) = F(\beta,\mathbf{x}) + z(\mathbf{x}) \quad (3)$$

Where F(β,**x**) is the deterministic part and represents an averaged approximation of the response. It can be expressed as an ordinary polynomial regression of **x**. β is the vector of regression coefficients. The second part z(**x**) is the realization of stochastic process, and it provides the approximation of local fluctuation. Further details are available in reference (Lophaven, Nielsen *et al*., 2002).

## 3. Assessing the Probability of Failure

In this research, KDE and WASM techniques were employed to determine the probability of failure. These methods are detailed below.

### 3.1 Kernel Probability Density Function

KDE is a non-parametric technique used to estimate the probability density function (PDF) of a random variable from sample data. Unlike parametric methods, KDE does not require prior assumptions about the data's distribution; instead, it directly uses the data to reveal the underlying distribution's configuration (Bowman and Azzalini 1997). The method works by applying a kernel function to each data point, determining a local density at every point within the data domain. These local densities are then summed to produce a continuous and smooth approximation of the data's PDF. KDE finds applications across numerous scientific and industrial fields, including economics, biology, and big data analysis. A critical aspect of KDE is the selection of an appropriate bandwidth, which controls the influence of each data point on the final

estimate. An overly small bandwidth can result in oscillatory and noisy estimates, potentially highlighting minor fluctuations. Conversely, a bandwidth that is too large may obscure significant details within the data. Consequently, determining the optimal bandwidth is a primary challenge in KDE implementation. Despite this challenge, KDE offers significant advantages. It can accurately estimate data distributions without making specific distributional assumptions, making it particularly adept at detecting complex and unusual patterns, especially when the data distribution is variable or intricate. The kernel probability density function is defined as follows:

$$\hat{f}_h(x)=\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) \quad (4)$$

Where, $x_1,x_2,\ldots.x_n$ is a random sample from an unknown distribution, n is the number of sample members, K is the kernel function, and h is the bandwidth. In this research, the normal kernel function is used.

### 3.2 Weighted Average Simulation Method

Rashki *et al*., (2012) introduced the WASM for both estimating the probability of failure and identifying the most probable failure point. This method provides a novel definition for the probability of failure. WASM employs a weighted index, from which the probability of failure is calculated as the ratio of the sum of weighted indices within the failure domain to the total sum of weighted indices across the entire domain. Samples are generated at specific intervals using a uniform distribution function applied to all random variables. This process ensures comprehensive representation of the design space by the samples, while the weighted indices differentiate individual samples. Crucially, the sampling interval for each variable must include a segment of the failure region. In this method, the probability of failure is calculated as follows:

$$P_f=\frac{\sum_{i=1}^{N} I(i).W(i)}{\sum_{i=1}^{N} W(i)} \quad (5)$$

Where N is the number of instances created, W(i) is the weight of the ith sample. The index I(i) is calculated from the follows:

$$I(i)=\begin{cases}1, & \text{If the sample fails}\\ 0, & \text{otherwise}\end{cases} \quad (6)$$

## 4. Proposed Algorithm

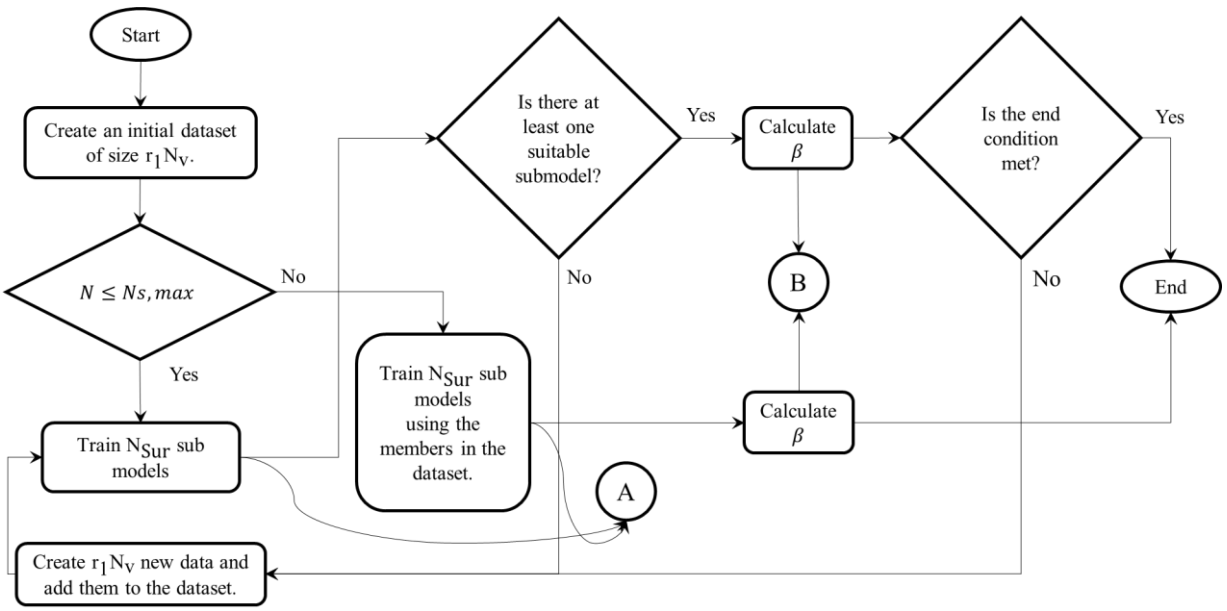
This section outlines the different stages of the proposed algorithm, which is designed based on the following principle:

> *The iterative process of sampling the limit state function and training sub-models continues until the error in the reliability index, calculated from two successive surrogate models, falls below or equals a permissible limit.*

### 4.1 Algorithm Overview

As illustrated in **Fig 1**, the main steps of this algorithm are as follows:

1. **Initialization:** Create an initial dataset of size $r_1N_v$ is created. In this research, $r_1$=10 , and $N_v$ represents the number of random variables.
2. **Dataset Size Check:** If the current number of samples in the dataset ( N ) exceeds the maximum permitted value ($N_{s,max}$), proceed to Step 3 and conclude the calculation.
3. **Sub-model Training:** Train $N_{sur}$ sub-models.



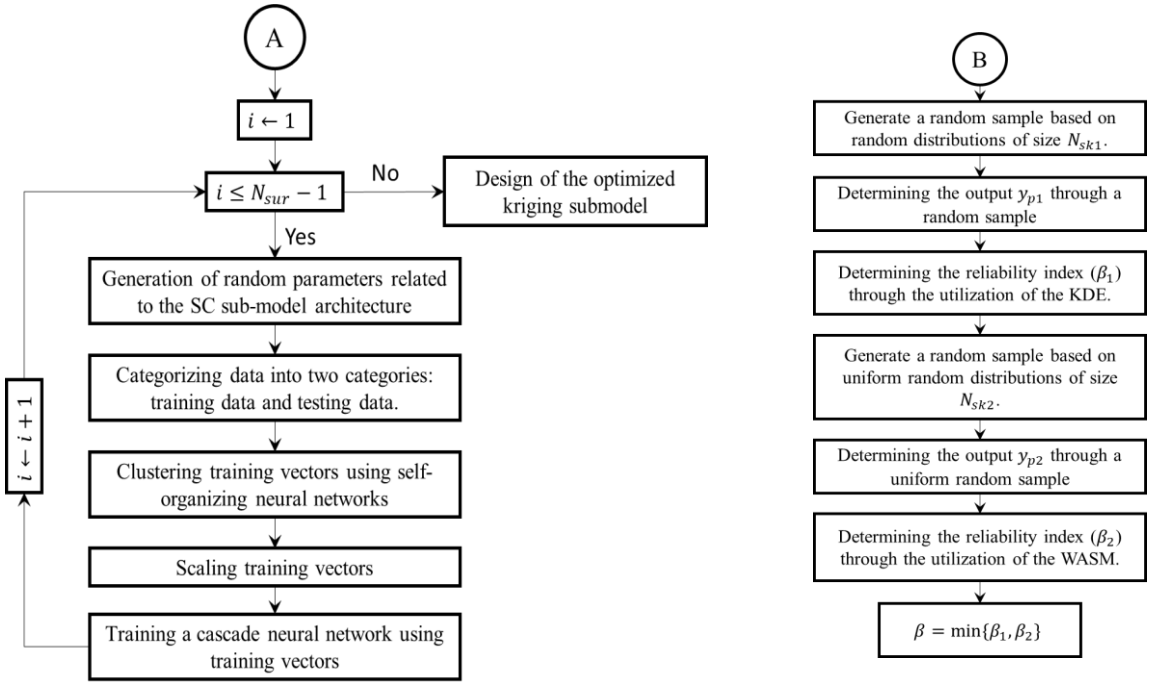



**Fig. 1: Flowchart of the proposed reliability index calculation algorithm.**

4. **Reliability Index Calculation and Iteration:**
   - If at least one appropriate sub-model is found, compute the reliability index.
   - If no appropriate model exists, generate new samples and reiterate the sub-model training process (return to Step 3).

5. **Convergence Check:**

If the error in the reliability index, obtained from two consecutive steps, is less than or equal to the allowable limit, the process terminates ( $e_\beta \leq e_a$ ). Otherwise, repeat the sampling and sub-model training operations. The error $e_\beta$ is defined as:

$$e_\beta = \frac{|\beta_c - \beta_p|}{\beta_p} \leq e_a \quad (7)$$

Where, $e_\beta$ is the error derived from the reliability index calculation between two consecutive steps, $e_a$ is the allowable limit error, $\beta_c$ is the current reliability index, and $\beta_p$ is the previous reliability index.

The surrogate model proposed in this research consists of $N_{sur} - 1$ hybrid SOM network integrated with CFNN models, termed SC models (as depicted in **Fig 2**). This is complemented by a single optimized kriging model. Within the SC hybrid model, input variables are initially clustered using the SOM network. The resulting class number for each sample (c) is then transmitted as an additional input variable to the CFNN.

The SC hybrid sub-models possess a flexible, variable architecture. By tuning specific parameters, users can modify the network's structure, thereby influencing and optimizing its performance.

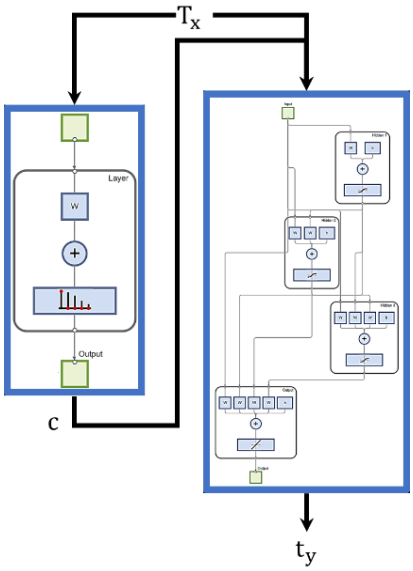



**Fig. 2: Structure of SC hybrid models**

This research optimizes the SC hybrid sub-models by considering the following architectural parameters:

- **SOM Network Dimensions:** $d_1$ and $d_2$.
- **CFNN Parameters:** Number of neurons per layer (W) and cascade depth (D).

In addition to these neural network components, an optimized kriging sub-model is also utilized. The overarching strategy leverages the predictive capabilities of neural networks as the primary foundation for alternative models. However, the inclusion of an optimized kriging model capitalizes on its distinct strengths.

As depicted in **Fig 2**, the initial parameter values for the SC model's architecture are determined randomly:

- **$d_1$ and $d_2$**: Random integers, constrained by $d_1, d_2 \leq d_m$, where $d_m$ equals the maximum map dimension in the SOM neural network.
- **W**: Random integer, satisfying $N_v/2 \leq W \leq W_m$. where, $W_m$ equals the maximum number of neurons in the hidden layers of the CFNN neural network.
- **D**: Random integer, constrained by by $D \leq D_m$, where, $D_m$ is the maximum number of hidden layers in the CFNN network.

### 4.2 Algorithm for Surrogate Sub-Model Creation (Label A in Fig 1)

This section details the systematic process for generating multiple surrogate sub-models. It begins with preparing the data through splitting, feature augmentation using SOM, and normalization. Subsequently, neural networks are trained on this processed data, a step that is iterated to create a pool of sub-models. Finally, an optimized Kriging model is developed, and an ensemble approach is used to combine the predictions from these sub-models via a weighted average.

1. **Data Splitting:**The dataset is divided into training (70%) and testing (30%) subsets.
2. **Feature Engineering and Normalization:** This section includes two basic steps:

- **SOM Clustering:** Training vectors are classified using a SOM network. The resulting group number for each vector is incorporated as an additional feature into the training vector.
- **Data Scaling:** Input vectors and the problem's output value are normalized to the range [−1/2,1/2] using the following formula:

$$t_i = \frac{x_i - l_{bi}}{u_{bi} - l_{bi}} - \frac{1}{2},\ i=1,2,\ldots N_v\ ,\ -1/2 \leq t_i \leq 1/2 \qquad (8)$$

Where, $x_i$ is the i-th original design variable., $t_i$ is the i-th scaled design variable, $l_{bi}$ is the lower bound of the i-th design variable, $u_{bi}$ is the upper bound of the i-th design variable. The lower and upper bounds are calculated as:

$$l_{bi} = F_i^{-1}(\varepsilon),\ i=1,2,\ldots N_v \qquad (9)$$

$$u_{bi} = F_i^{-1}(1-\varepsilon),\ i=1,2,\ldots N_v \qquad (10)$$

Here, ε is a small number (set to 0.01 in this research), and $F_i^{-1}$ is the inverse cumulative distribution function assuming a normal distribution. The output variable y is scaled using the same methodology.

3. **Neural Network Training:** The neural network is trained using the scaled training vector $T_x=[t_1,t_2,\ldots,t_{N_v}]$ and the scaled output $t_y$ . this step is repeated $N_{sur} - 1$ times to create multiple neural network sub-models.
4. **Optimized Kriging Sub-Model:** An optimized kriging model is developed as the final sub-model. Parameters influencing its effectiveness, including bias function type, kernel function

type, kernel scaling type, sigma parameter, and standardization type, are optimized using the Quasi-Newton algorithm. MATLAB (2023) was used for this optimization.

5. **Ensemble Prediction:** The central idea is to use a weighted average of the outputs from all $N_{sur}$ surrogate sub-models. The final predicted output $y_p$ is calculated as:

$$y_p = \frac{\sum_{i=1}^{N_{sur}} w_i y_{pi}}{\sum_{i=1}^{N_{sur}} w_i} \quad (11)$$

Where, $y_p$ is the ensemble predicted value, $y_{pi}$ is the predicted value from the i-th sub-model, $w_i$ is the weight of the i-th sub-model, calculated as:

$$w_i = \frac{1}{prf_i} \quad (12)$$

Where, $prf_i$ is the performance of the i-th sub-model, defined as the root mean square of the errors on the test dataset.

### 4.3 Algorithm for Reliability Index Calculation (Label B in Fig 1)

This section outlines the procedure for calculating a reliability index. The algorithm involves:

1. **Random Sample Generation (Distribution-Based):** Generate a random sample of size $N_{sk1}$ based on the random distributions of the design variables.
2. **Surrogate Model Prediction (Sample 1):** Determine the output $y_{p1}$ for this random sample using the ensemble surrogate model (Equation (11)).
3. **Reliability Index Estimation (KDE):** Calculate the reliability index $\beta_1$ using KDE applied to $y_{p1}$.
4. **Uniform Random Sample Generation:** Generate a random sample of size $N_{sk}$ based on uniform random distributions.
5. **Surrogate Model Prediction (Sample 2):** Determine the output $y_{p2}$ for this uniform random sample using the ensemble surrogate model (Equation (11)).
6. **Reliability Index Estimation (WASM):** Calculate the reliability index $\beta_2$ using the WASM, as defined by Equation (5).
7. **Final Reliability Index:** The final reliability index $\beta$ is determined by the minimum value obtained from Step 3 ($\beta_1$) and Step 6 ($\beta_2$).

## 5. Numerical Examples

This section provides numerical examples, including the assessment of reliability indices for nonlinear limit state functions and their evaluation across various engineering scenarios. **Table 1.** Displays the key parameters associated with the suggested algorithm for addressing these examples.

**Table 1: Primary parameters of the proposed method for the numerical examples**

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $w_d$ | 30 | $d_m$ | 4 |
| $e_a$ | 0.03 | $N_{sk}$ | 1000 $N_{var}$ |
| $N_{s,max}$ | 100 $N_{var}$ | $N_{pop}$ | 10 $N_{var}$ |
| $D_{max}$ | 2 | $N_{sur}$ | 3 |
| $W_{max}$ | $N_{var}$ | $R_{train}$ | 0.7 |

Where, $w_d$ is the minimum weight required for a suitable sub-model, $e_a$ is the permissible error of calculating the reliability index in two consecutive steps, $N_{s,max}$ is the maximum number of time-consuming function evaluations, $D_{max}$ is the maximum number of hidden layers of CFNN, $W_{max}$ is the maximum number of neurons in a CFNN hidden layer, $N_{sk}$ is the random sample size for calculating probability, $N_{pop}$ is the random sample size at each stage for constructing the surrogate model, $N_{sur}$ is the number of surrogate sub-models, and $R_{train}$ is the ratio of the number of training members of the dataset to the total members.

The $w_d$ parameter was selected to ensure the surrogate sub-model achieved an RMSE of no less than $(\frac{1}{30})^{-1}=0.0333$. The value of 0.03 for the $e_a$ parameter was selected to prevent an increased computational burden from additional function calls, which do not yield a commensurate improvement in the model's accuracy. The $N_{s,max}$ parameter was set to 100 times the number of random variables in the problem for two key reasons. First, it allows the algorithm to scale its sampling effort as the number of variables increases. Second, it is expected that the algorithm will maintain its ability to solve the problem effectively up to this specific threshold ($\leq 100\ N_{var}$). The $D_{max}$ parameter is configured with a maximum of two hidden layers, as surrogate sub-models are anticipated to deliver sufficient performance within this architectural constraint. The $W_{max}$ parameter, similarly capped at the number of problem variables, defines an appropriate limit for the required number of weights and biases in the hidden layers. The $d_{max}$ parameter is likewise set to 4, corresponding to a maximum of 16 clusters in the SOM data partitioning stage. This value provides an optimal balance, delivering adequate clustering resolution without introducing prohibitive computational overhead. The $N_{sk}$ parameter - employed in probability calculations via both the KDE and WASM methodologies - is set to 1000 times the number of variables. This value provides adequate resolution for both KDE fitting and subsequent WASM

processing. The $N_{pop}$ parameter is set to 10 times the number of problem variables. This value ensures gradual sampling escalation while avoiding prohibitive computational costs. The $N_{sur}$ parameter is configured for three surrogate models: two neural network-based models and one optimized Kriging model. This ensemble is expected to adequately capture the problem's complexity. The $R_{train}$ parameter is set to 0.7, which corresponds to a 70/30 split, allocating 70% of the data for training the surrogate models and 30% for validation/testing.

Moreover, the rates of error and enhancement are determined using the subsequent equations:

$$E_\beta = \frac{100|\beta_{ref}-\beta_p|}{\beta_{ref}} \quad (13)$$

$$E_r = \frac{100|r_{ref}-r_p|}{r_{ref}} \quad (14)$$

$$Im = \frac{100(NFE_{ref} - NFE)}{NFE_{ref}} \quad (15)$$

Where, $E_\beta$ is the computational error in estimating the reliability index, $E_r$ is the computational error in estimating reliability, Im is the percentage of improvement, $\beta_{ref}$ is the reliability index calculated in the reference, $r_{ref}$ is the estimated reliability in the reference, $NFE_{ref}$ is the number of function evaluations declared in the reference, $\beta_p$ is the reliability index calculated by the proposed method, NFE is the number of function evaluations by the proposed method, $r_p$ is the reliability estimated by the proposed method.

### 5. 1 Mathematical Examples

In this section, five nonlinear limit state functions are analyzed (**Table 2**). Functions 1 and 2 are nonlinear functions with 2 variables, while functions 3, 4, and 5 consist of 4, 6, and 6 variables, respectively.

**Table 2: Limit state functions and probability distributions of the random variables**

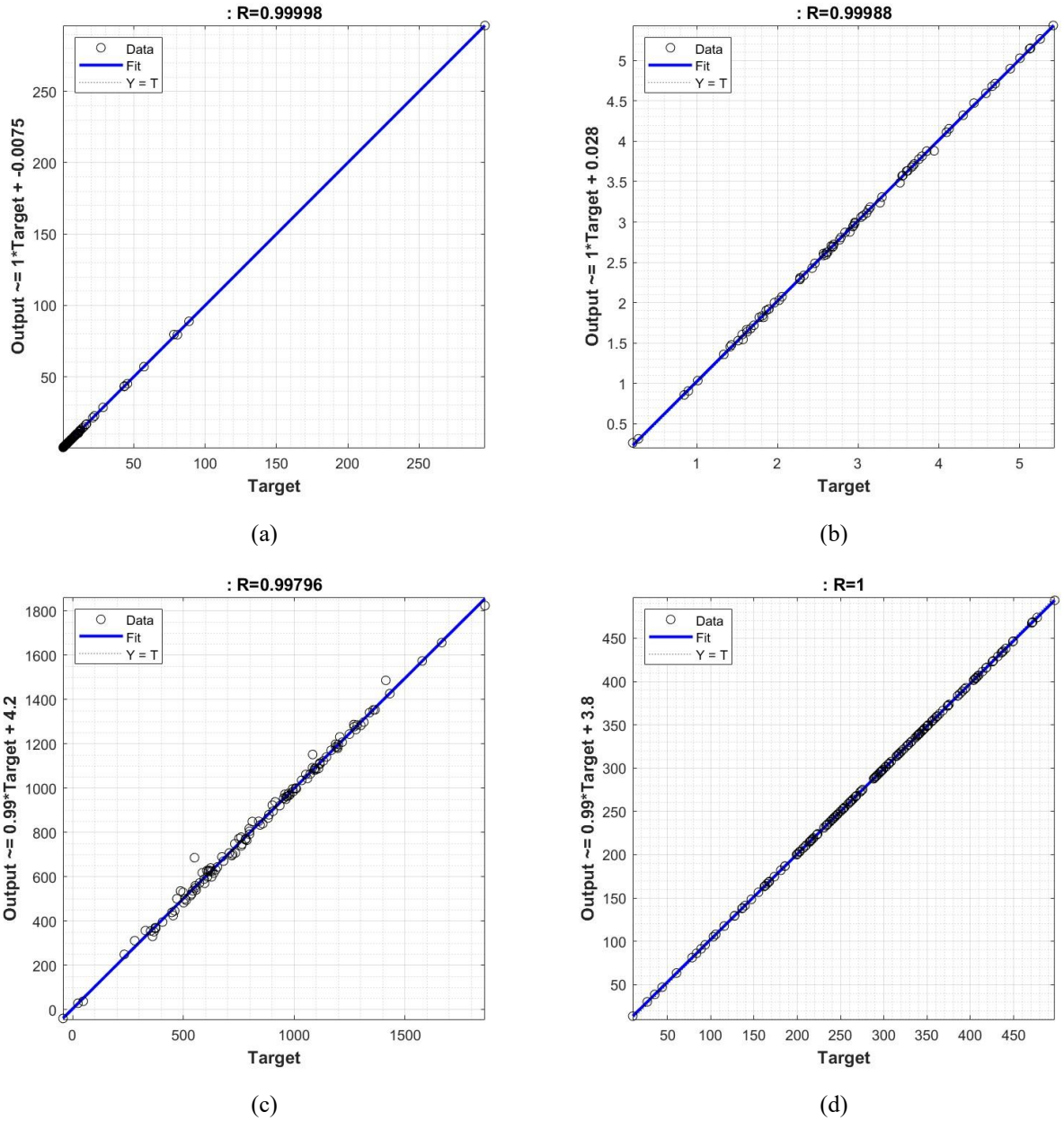
| No | limit State Function | PDFs of Random Variables | References |
|---|---|---|---|
| 1 | $g(X)=2.5-0.2357(x_1-x_2)+0.00463(x_1+x_2-20)^4$ | $x_1$& $x_2$ are N(10,3) | Shayanfar, Barkhordari *et al.*, (2017) |
| 2 | $g(X)=0.1(x_1-x_2)^2-\frac{x_1+x_2}{\sqrt{2}}+2.5$ | $x_1$& $x_2$ are N(0,1) | Shayanfar, Barkhordari *et al.*, (2017) |
| 3 | $g(X)=x_1.x_2.d-0.59\frac{(x_1.x_2)^2}{x_3.b}-x_4$ | $x_1$ is N(4.08,0.0816)<br>$x_2$ is N(44,4.62)<br>$x_3$ is N(3.12,0.4368)<br>$x_4$ is N(2052, 246.24)<br>b=12<br>d=9 | Nowak and Collins (2012) |
| 4 | $g(X) = x_1 + 2x_2 + 2x_3 + x_4 - 5x_5 - 5x_6 + 0.001\sum_{i=1}^{6} sin(100x_i)$ | $x_1$~ $x_4$ are LN(120,12)<br>$x_5$ is LN(50,15)<br>$x_6$ is LN(40,12) | Liping and Grandhi (1994), (Shayanfar, Barkhordari *et al.*, 2017) |
| 5 | $g=3r-\lvert\frac{2F}{mW^2}\sin(\frac{Wt_1}{2})\rvert$<br>$W^2=\frac{c_1+c_2}{m}$ | m is N(1,0.05)<br>$c_1$ is N(1,0.1)<br>$c_2$ is N(0.1,0.01)<br>r is N(0.5,0.05)<br>F is N(1,0.2)<br>$t_1$ is N(1,0.2) | Keshtegar (2017) |

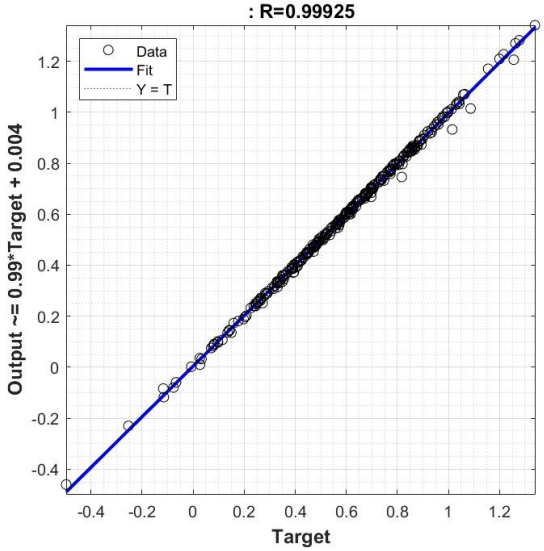
In **Table 3**, the reliability index is computed for the limit state functions presented in **Table 2**. The reliability index derived from the surrogate model is denoted as $\beta_p$, while the reliability index in the reference is represented by $\beta_{ref}$. NFE refers to the number of function evaluations in the proposed method, whereas $NFE_{ref}$ indicates the number of function evaluations in the reference. Additionally, R signifies the linear regression coefficient that relates the outputs of the surrogate model to those of the original model.

**Table 3: Reliability indices obtained from the surrogate model and the original model**

| No | NFE | $NFE_{ref}$ | R | $\beta_p$ | $\beta_{ref}$ | $r_p$ % | $r_{ref}$ % | Im % | $E_\beta$ % | $E_r$ % |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 1633 | ≈ 1 | 2.7149 | 2.7703 | ≈ 99.7 | ≈ 99.7 | ≈ 88 | ≈ 2 | ≈ 0 |
| 2 | 80 | 423 | ≈ 1 | 2.6957 | 2.6437 | ≈ 99.6 | ≈ 99.6 | ≈ 81 | ≈ 2 | ≈ 0 |
| 3 | 120 | Not reported | ≈ 1 | 2.2358 | 2.29 | ≈ 98.7 | ≈ 98.9 | - | ≈ 2 | ≈ 0 |
| 4 | 180 | 3402 | ≈ 1 | 2.2584 | 2.2353 | ≈ 98.8 | ≈ 98.7 | ≈ 95 | ≈ 1 | ≈ 0 |
| 5 | 300 | Not reported | ≈ 1 | 1.8659 | 1.8652 | ≈ 96.9 | ≈ 96.9 | - | ≈ 0 | ≈ 0 |

**Fig. 3** shows a linear regression comparing the outputs of the surrogate model against the original model. All regression coefficients are close to 1.






(e)

**Fig. 3: Linear regression between the outputs of the surrogate model and the original model for the nonlinear limit state function**

A more favorable outcome is observed for limit state function number 4, with an Im index of approximately 95%, an $E_\beta$ index of about 1%, and an $E_r$ index of nearly 0%. A less favorable outcome is noted for limit state function number 2, with an Im index of around 81%, an $E_\beta$ index of about 1%, and an $E_r$ index of roughly 0%.

### 5.2 Engineering Examples

Several engineering examples are examined in this section. These examples include *Roof Truss* (Keshtegar and Kisi 2017), *I Beam* (Keshtegar 2017) , *Three Span Continues Beam* (Zhao, Yue et al. 2015), *Buckling of Columns* (Huang and Zhang 2013) problems and a braced frame with Concrete-Filled Tube (CFT) members. A summary of the results is shown in **Table 9**.

#### 5.2.1 Roof Truss

In this example, the compression members of this truss are made of reinforced concrete and the tension members are made of steel (**Fig 4**) The limit state corresponds to the maximum deflection at point C.

$$Y= 0.03-\frac{ql^2}{2}\left(\frac{3.81}{A_cE_c}+\frac{1.13}{A_sE_s}\right) \quad (16)$$

where $A_c$, $A_s$ are the cross sectional areas of reinforced concrete and steel bars, respectively, $E_c$, $E_s$ are their corresponding elastic modulus, l is the length of the truss.

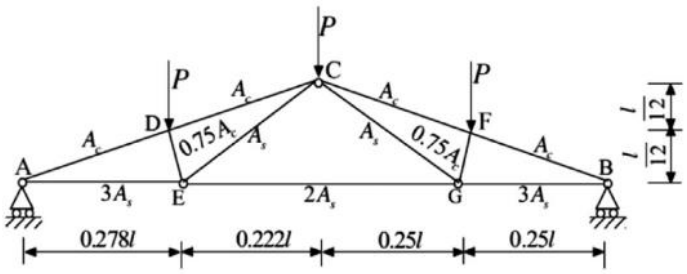



**Fig. 4: Schematic view of roof truss example**

This example includes six standard random variables, the statistical characteristics of which are detailed in **Table 4**.

**Table 4: Basic random variables for the roof truss**

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| q | Normal | 20000 | 1400 | N/m |
| l | Normal | 12 | 0.12 | m |
| $A_s$ | Normal | $9.82\times10^{-4}$ | $5.982\times10^{-5}$ | $m^2$ |
| $A_c$ | Normal | 0.04 | 0.0048 | $m^2$ |

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| $E_s$ | Normal | $1\times10^{11}$ | $6\times10^{9}$ | Pa |
| $E_c$ | Normal | $2\times10^{10}$ | $1.2\times10^{9}$ | Pa |

**Fig5** illustrates the linear regression relationship between the outputs of the surrogate model and those of the original model. The regression coefficient approaches 1 with a high degree of accuracy, signifying the effective performance of the proposed surrogate model in addressing this issue.

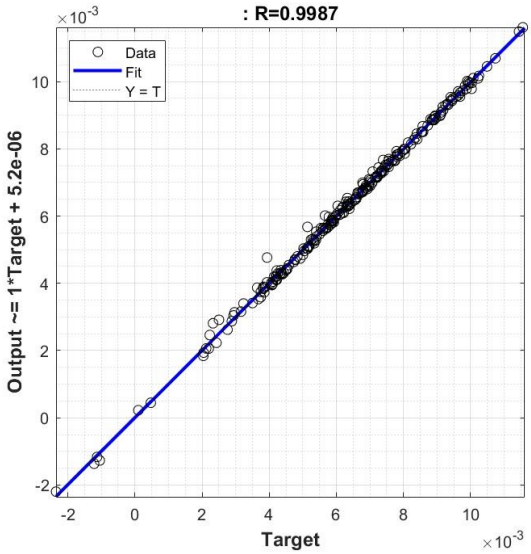



**Fig. 5: Linear regression between the results of the surrogate model and the original model for the roof truss problem**

The reference method investigated in this example (Keshtegar and Kisi 2017) proposes a hybrid reliability analysis approach that combines the M5 model tree (M5Tree) as a surrogate model with Monte Carlo Simulation (MCS). The M5Tree is first trained using a limited number of samples to approximate the complex implicit limit state function. Subsequently, a large-scale MCS is performed using this fast surrogate model instead of the original time-consuming function, enabling efficient and accurate estimation of failure probability with significantly reduced computational cost.

The NFE derived from the suggested method for addressing this example is 240. In comparison, the reference value is 500, signifying a 52% enhancement. The reliability index computed using the proposed method is 2.1667, whereas the reference index is 2.3459, indicating an error rate of 7.6%. The reliability calculated through the proposed method stands at 98.5, compared to 99.1% in the reference, which reflects an error margin of 0.6%. A summary of the results achieved in this problem is presented in **Table 9**.

### 5.2.2 I-Beam

In this example, an I beam is subject to a concentrate force P with a distance A away from the fixed end as depicted (**Fig 6**). The performance function is as follows:

$$Y= S-\frac{12\ PA\ (L-A)\ H}{2L\ (WH^3-(W-T_W)(H-2T_H)^3} \quad (17)$$

In which, all the parameters used are shown in **Fig 6Error! Reference source not found.**. This example involves eight random variables, whose statistical properties are presented in **Table 5**.

**Table 5: Basic random variables for the I-beam problem**

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| P | Normal | 6500 | 300 | N |
| L | Normal | 120 | 5 | mm |
| A | Log-Normal | 72 | 5 | mm |
| S | Gumbel | $1.7\times10^{5}$ | 4760 | Mpa |
| H | Normal | 2.3 | 0.04 | mm |
| W | Weibull | 2.97 | 0.075 | mm |

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| $T_W$ | Normal | 0.16 | 0.02 | mm |
| $T_H$ | Normal | 0.26 | 0.02 | mm |

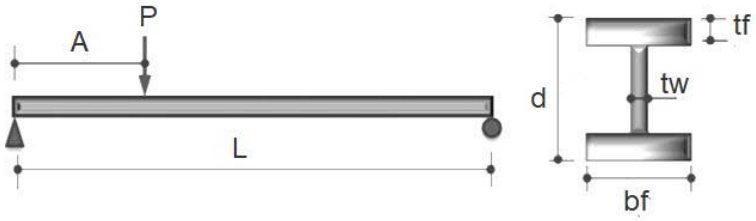



**Fig. 6: Schematic of the I-beam problem**

**Fig7** illustrates the linear regression relationship between the outputs of the surrogate model and those of the original model. The regression coefficient approaches 1 with a high degree of accuracy, indicating the applicability of the proposed surrogate model for this issue.

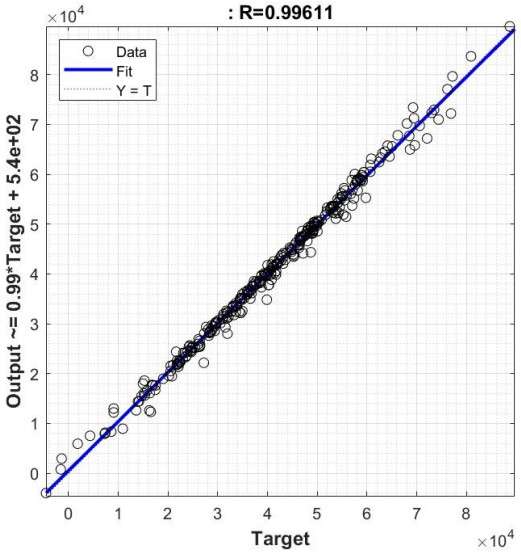



**Fig. 7: Linear regression between the outputs of the surrogate model and the original model for the I-beam problem**

The reference used in this example (Keshtegar 2017) proposes a Hybrid Conjugate Finite-Step Length (CFSL-H) method to enhance the robustness and efficiency of the FORM. This approach combines two conjugate gradient methods, Conjugate Descent (CD) and RMIL, using a dynamic participation factor to adaptively compute a hybrid conjugate search direction. This hybrid direction is integrated with a self-adaptive finite-step length determined via Armijo's rule and a sufficient descent condition, eliminating the need for complex line searches. The method efficiently locates the Most Probable Point (MPP) and ensures stable convergence, even for highly nonlinear and implicit limit state functions, outperforming traditional FORM variants in both accuracy and computational effort.

The NFE derived from the suggested method for addressing this example is 320. In comparison, the reference value is 400, signifying a 47% enhancement. The reliability index computed using the proposed method is 2.5187, whereas the reference index is 2.8196, indicating an error rate of 11%. The reliability calculated through the proposed method stands at 99.4%, compared to 99.8% in the reference, which reflects an error margin of 0.4%. A summary of the results obtained from this problem is presented in **Table 9**.

### 5.2.3 Three Span Continues Beam

A three-span beam structure, as illustrated in **Fig 8**, has been chosen for consideration as an additional engineering problem. The performance function can be defined:

$$Y=\frac{L}{400}-0.0069\frac{wL^4}{EI} \quad (18)$$

Where w represents the uniformly distributed load, E is the elastic modulus of the beam, and I is the moment of inertia. All three random variables are independent and follow a normal distribution, with their corresponding statistical properties provided in **Table 6**.

**Table 6: Basic random variables for the three-span continuous beam**

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| W | Normal | 10 | 0.4 | KN/m |
| E | Normal | $2\times10^7$ | $0.5\times10^7$ | $KN/m^2$ |
| I | Normal | $8\times10^{-4}$ | $1.5\times10^{-4}$ | $m^4$ |

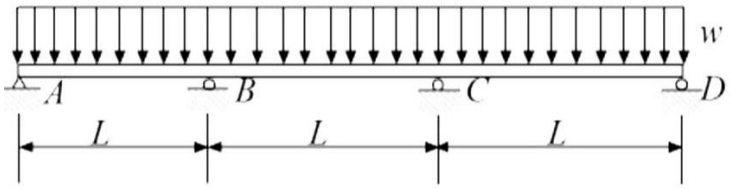



**Fig. 8: Schematic of the three-span continuous beam problem**

**Fig 9** shows the linear regression relationship between the surrogate model outputs and the original model outputs. The regression coefficient is close to 1 with high accuracy, demonstrating the surrogate model's effectiveness in addressing this problem.

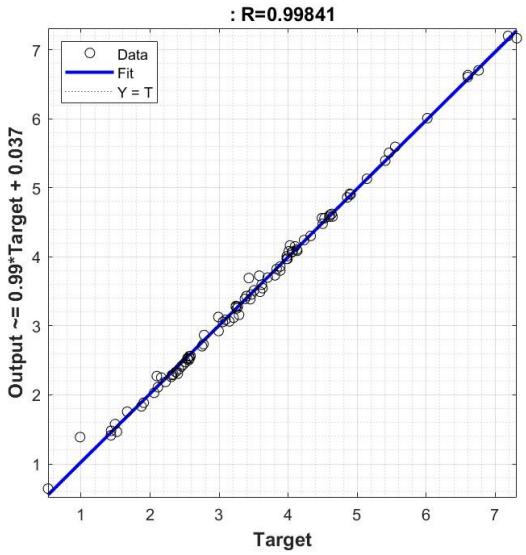



**Fig. 9: Linear regression between the outputs of the surrogate model and the original model for the three-span continuous beam problem**

The reference used in this example (Zhao, Yue *et al*., 2015), presents an efficient hybrid method for structural reliability assessment that combines Adaptive Importance Sampling and a Kriging Metamodel using an active learning mechanism. In this method, the Markov Chain Metropolis algorithm (Beichl, Sullivan *et al*., 2002) is employed to generate samples in the failure region, which are then used as centers for importance sampling. An initial Kriging model is constructed using these samples and is iteratively updated by identifying critical points near the limit state surface, via the U learning function, to enhance prediction accuracy in important regions. Finally, the failure probability and its coefficient of variation are computed by evaluating only a limited number of samples with the actual performance function, thereby avoiding costly and unnecessary evaluations. This method demonstrates high efficiency and accuracy, particularly for engineering problems with complex implicit performance functions.

The proposed approach required only 90 function evaluations (NFE) for this example, compared

to a reference value of 2100, representing a 96% reduction. The calculated reliability index was 2.7893 (reference: 3.0282), resulting in an 8% error. The corresponding reliability was 99.7% (reference: 99.9%), with an error margin of 0.2%. These results are summarized in **Table 9**.

### 5.2.4 Buckling of Columns

As illustrated in **Fig 10**, this example considers a column subjected to axial compressive loads. The distributions of the column's random variables are detailed in **Table 7**. The column possesses sufficient slenderness to experience failure by buckling under compressive load. The performance function for the column can be expressed as:

$$Y=\frac{\pi^2 EI}{(cL)^2}-F\ ,\ c=1,\ I=\frac{\pi}{64}((D+T)^4-D^4) \quad (19)$$

Where I is the moment of inertia of the column's cross-section about its centroidal axis, E is the modulus of elasticity of the material, L is the length of the column, T and D are the wall thickness and diameter of the hollow cylinder, respectively, and c=1 is the length coefficient.

**Table 7: Basic random variables for the column buckling problem**

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| E | Log-Normal | 203000 | 30.45 | Mpa |
| D | Normal | 23.5 | 0.2 | mm |
| T | Normal | 4 | 0.1 | mm |
| L | Normal | 2500 | 50 | mm |
| F | Gumbel | 3000 | 300 | N |

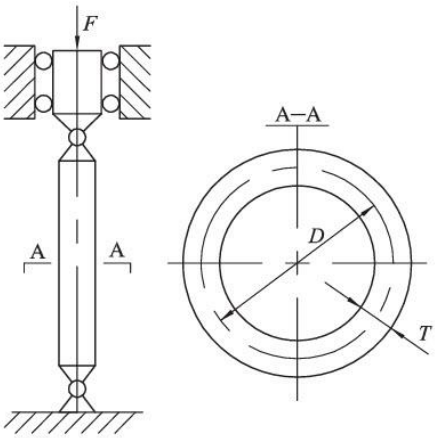



**Fig. 10: Schematic of the column buckling problem**

**Fig 11** shows the linear regression relationship between the outputs of the surrogate model and the outputs of the main model. The regression coefficient is close to 1 with high accuracy, indicating the effectiveness of the surrogate model in solving this problem.

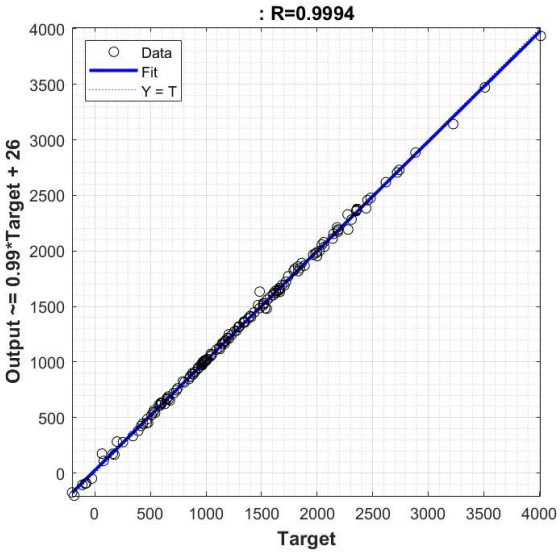



**Fig. 11: Linear regression between the outputs of the surrogate model and the original model for the column buckling problem**

The reference used in this example (Huang and Zhang 2013) presents a method for reliability-sensitivity analysis of mechanical and structural systems under uncertainty, which combines the Dimension Reduction Method (DRM) and Saddlepoint Approximation (SPA). In this method, the statistical moments of the performance function (such as mean, variance, skewness, and kurtosis) are first efficiently computed using the bivariate dimension reduction method and numerical integration. Then, using these moments, a truncated cumulant generating function (CGF) is approximated, and the saddlepoint approximation is employed to accurately estimate the failure probability, probability density functions (PDF), and cumulative distribution functions (CDF) of the structural response. Finally, through analytical differentiation and the use of kernel functions, the sensitivity of the failure probability with respect to the distribution parameters of the basic input variables (such as mean and standard deviation) is calculated. By eliminating the need for derivative calculations of the performance function and the identification of the most probable failure point, this method proves to be highly efficient and accurate for problems with complex and implicit response functions (such as finite element models).

The reliability index calculated by the proposed method is 1.8724, compared to 1.7396 in the reference, indicating an 8% error. The reliability calculated by the proposed method is 96.9%, versus 95.9% in the reference, corresponding to a 1% difference. A summary of the results for this problem is presented in **Table 9**.

### 5.2.5 Braced Frame with CFT Members

As demonstrated in **Fig 12** ,a braced frame with CFT members is investigated under loads $P_1$ to $P_2$. The structure was modeled using the SM toolbox (defined in MATLAB) within SAP 2000 software (Javanmardi and Ahmadi-Nedushan 2021, Javanmardi, Ahmadi-Nedushan *et al*., 2023). The distributions of the random variables are detailed in **Table 8**. The constant parameters are as follows:

**Table 8: Basic random variables for the CFT frame**

| Variable | Distribution | Mean | Standard Deviation | Dimension |
|---|---|---|---|---|
| $P_1$ | Gumbel | 30000 | 3000 | N |
| $P_2$ | Gumbel | 35000 | 3500 | N |
| $P_3$ | Gumbel | 40000 | 4000 | N |
| $P_4$ | Gumbel | 45000 | 4500 | N |
| $E_s$ | Log-Normal | $2\times10^5$ | $2\times10^4$ | Mpa |
| $E_c$ | Log-Normal | $2.5\times10^4$ | $2.5\times10^3$ | Mpa |

L=5 m , H=4 m, are the width and height of the frame, respectively. $d_{br}$=120 mm , $t_{br}$=10 mm are the diameter and thickness of the Br sections, respectively. $d_{be}$=100 mm , $t_{be}$=10 mm are the diameter and thickness of the Be sections,

respectively. $d_{col}$=150 mm , $t_{col}$=10 mm are the diameter and thickness of the Col sections, respectively.

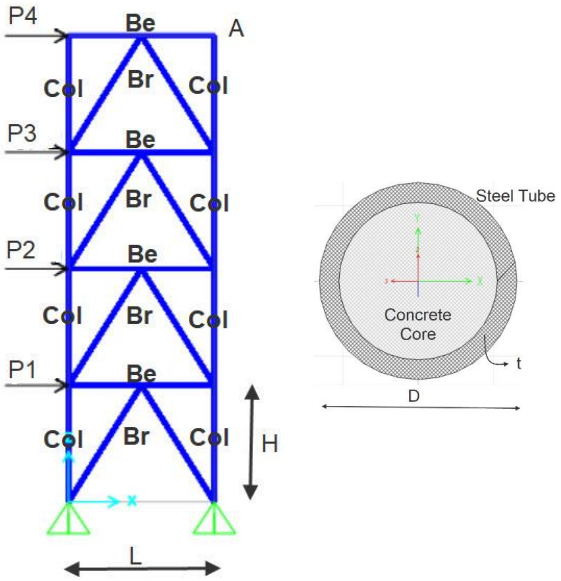



**Fig. 12: Schematic of the CFT frame problem**

The limit state function for this problem can be expressed as:

$$Y=\frac{\Delta_p}{\Delta_A}-1, \quad \Delta_p=10 \text{ mm} \qquad (20)$$

where $\Delta_A$ is the displacement of point A.

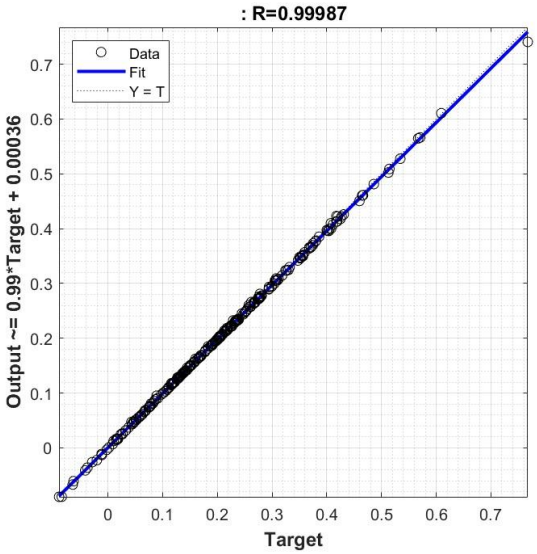



**Fig. 13: Linear regression between the outputs of the surrogate model and the original model for the CFT frame problem**

The reliability index calculated by the proposed method is 1.5546, compared to 1.5613 using the KDE function, indicating a 0.4% error. The reliability calculated by the proposed method is 94.0%, versus 94.08% using the KDE function, showing a 0.1% difference. A summary of the results for this problem is presented in **Table 9**. A random sample of size 5000 from SAP 2000 was used to construct the kernel density function. **Fig 13** shows the linear regression relationship between the outputs of the surrogate model and the outputs of the main model.

### 5.2.6 Summary of Results of Engineering Problems

This section presents a summary of the results obtained from the engineering problems in **Table 9**.

**Table 9: Summary of reliability index estimation results for the engineering problems**

| Problem | NFE | $NFE_{Ref}$ | R | $\beta_p$ | $\beta_{ref}$ | $r_p$ % | $r_{ref}$ % | Im % | $E_\beta$ % | $E_r$ % |
|---|---|---|---|---|---|---|---|---|---|---|
| (Roof Truss) | 240 | 500 | ≈ 1 | 2.1667 | 2.3459 | ≈ 98.5 | ≈ 99.1 | ≈ 52 | ≈ 7.6 | ≈ 0.6 |
| (I-Beam) | 320 | 400 | ≈ 1 | 2.5187 | 2.8196 | ≈ 99.4 | ≈ 99.8 | ≈ 47 | ≈ 11 | ≈ 0.4 |
| (Three Span Continues Beam) | 90 | 2100 | ≈ 1 | 2.7893 | 3.0282 | ≈ 99.7 | ≈ 99.9 | ≈ 96 | ≈ 8 | ≈ 0.2 |
| (Buckling of Columns) | 150 | Not Reported | ≈ 1 | 1.8724 | 1.7396 | ≈ 96.9 | ≈ 95.9 | - | ≈ 8 | ≈ 1 |
| (Braced with CFT members) | 300 | 5000 | ≈ 1 | 1.5546 | 1.5613 | ≈ 94 | ≈ 94.08 | ≈ 94 | ≈ 0 | ≈ 0 |

A more favorable outcome is observed for Problem 5.2.3 (Three-Span Continuous Beam), where the Im index is approximately 96%, the $E_\beta$ index is about 8%, and the $E_r$ index is nearly 0.2%. less favorable outcome is noted for Problem 5.2.2 (I-Beam), where the Im index is around 47%, the $E_\beta$ index is approximately 11%, and the $E_r$ index is roughly 0.4%.

## 6. CONCLUSION

This research introduces a technique for determining the probability of failure that leverages the capabilities of neural networks combined with the advantages of the kriging method. The core concept is as follows: the procedure for sampling the limit state function and training the sub-models continues until the error in the reliability index, computed from two successive surrogate models, falls below or equals a permissible threshold.

To evaluate the effectiveness of the proposed algorithm, several examples are presented, including five mathematical and five engineering problems. The results are summarized below:

- **Mathematical Examples:**
  - The linear regression coefficient in all examples is close to 1 with high accuracy, indicating the satisfactory performance of the proposed surrogate model.
  - The lowest Im is approximately 81%.
  - $E_\beta$ remains below 3% for all limit state functions.
  - $E_r$ is nearly 0% for all limit state functions.

- **Engineering Examples:**
  - The linear regression coefficient in all examples is close to 1 with high accuracy, confirming the model's effectiveness.
  - The minimum Im is roughly 47% (Problem 5.2.2).
  - The maximum $E_\beta$ across all problems is nearly 11% (Problem 5.2.2).
  - The highest $E_r$ observed is approximately 1% (Problem 5.2.4).

The summarized results reveal that for certain case studies, the calculated reliability index exhibited a higher-than-expected error. A comprehensive analysis of this observation must consider two key factors: first, the inherent error present in all model-based methodologies, including the one proposed; and second, the crucial distinction that the ultimate objective is the accuracy of the reliability probability itself, not the intermediate index. Given that the maximum error in the final reliability probability is shown to be only 1%, the proposed algorithm can be deemed highly effective, provided its predefined error margins are acceptable. For instance, in problem 5.2.2, the error in calculating the reliability index is approximately 11%. However, the final reliability probability predicted by the proposed algorithm is $\Phi(\beta_p) = 99.4\%$, compared to a reference value of $\Phi(\beta_{ref}) = 99.8\%$. This translates to an error of merely 0.4% in the probability, the metric of primary importance.

### Appendix

In this appendix, the pseudocodes of the method presented in the paper are explained. First, the main flow of the algorithm is described, and then the pseudocodes will be explained.

### Main Algorithm Flow

This algorithm consists of fundamental steps that are explained below:

1. **Initialization**: Set parameters and initialize variables
2. **Sampling Loop**:
   - Generate random samples from input distributions
   - Evaluate true function at sample points
3. **Surrogate Training**:
   - Train multiple neural networks with random architectures
   - Train one Kriging model
4. **Model Selection**: Filter models based on performance weights.

5. **Reliability Estimation**:
   - Predict using surrogate models
   - Calculate reliability index using two methods
   - Check convergence

6. **Output**: Return final reliability index and trained models.

The algorithm combines neural networks with self-organizing maps for preprocessing and Kriging for global approximation, using an iterative approach to refine the reliability estimate.

**Pseudocode for SCK**

In this section, the pseudocode of the SCK function, which is the main function for calculating reliability, is provided.

***FUNCTION*** *SCK(hf, Pdfs, Options)*
// INPUTS:
hf: function handle for the limit state function
Pdfs: cell array of probability distribution objects for each variable
Options: structure containing various options (wd, IterMax, Dm, Wm, dm, nPop, NSM, rmsetrn, trnRatio, elu, trainTime, useParallel, useGPU, KrgOptimizer, KrgMaxFunEvals, Nsk, erra)
// INITIALIZE:
*B = []* (to store beta values)
*EndCondition = false*
*WHILE EndCondition is false DO*
*Xt = []* (training inputs)
*yt = []* (training outputs)
*FOR ii from 1 to IterMax DO*
// Generate samples
*FOR jj from 1 to Nv (number of variables) DO*
*S(:, jj) = random samples from Pdfs{jj} of size nPop*
*END FOR*
*Xt = [Xt; S]*
// Evaluate the limit state function
*FOR jj from 1 to nPop DO*
*ys = hf(S(jj, :))*
*yt = [yt; ys]*
*END FOR*
// Train surrogate models (neural networks and Kriging)
*[Nets, Krg] = TrainSubSurs(Xt, yt, Dm, Wm, dm, ...* (other options))
// Store the training data
*DS.X = Xt*
*DS.y = yt*
// Check the weights of the models and remove those below threshold wd
*w = []*
*FOR jj from 1 to number of Nets DO*
*wn = Nets{jj}.w*
*IF wn < wd THEN*
*Nets{jj} = []  (remove the model)*
*ELSE*
*w = [w, wn]*
*END IF*
*END FOR*
// Check Kriging model weight
*wn = Krg.w*
*IF wn < wd THEN*
*Krg = []* (remove the model)
*ELSE*
*w = [w, wn]*
*END IF*
// If there are any models left (with weight >= wd)
*IF there are models (w is not empty) THEN*
// Generate a large set of samples for prediction
*FOR jj from 1 to Nv DO*
*X(:, jj) = random samples from Pdfs{jj} of size Nsk*
*END FOR*
// Predict the output for the large set using the trained models
*y = Predict(X, Nets, Krg)*
// Store the prediction data
*DSK.X = X*
*DSK.y = y*
// Compute the reliability index using the predicted outputs
*[pf, pdz] = cdfks(0, y)  (compute the CDF at 0)*
*ps = 1 - pf*
*bk = inverse CDF of normal distribution for ps (with mean 0, std 1)*
// Also compute the reliability index using WASM method
*[bw, ~] = WASM(Nets, Krg, Pdfs, 5, Nsk)*
// Store the minimum of the two beta estimates
*B = [B, min(bk, bw)]*
// Check for convergence
*IF length(B) >= 2 THEN*
*b2 = B(end)*
*b1 = B(end-1)*
*err = |b2 - b1| / b2*
*IF err <= erra THEN*
*BREAK  (break out of the inner loop)*
*END IF*
*END IF*
*ELSE*
*CONTINUE  (if no models, continue to next iteration)*
*END IF*
*END FOR* (end of inner loop)
// Set the final beta value and end the while loop
*b = B(end)*
*EndCondition = true*
*END WHILE*
*RETURN: b, Nets, Krg, DS, DSK*
***END FUNCTION***

**Pseudocode for Helper Functions**

In this section, the pseudocode for the helper functions used in the SCK function is explained.

**TrainSubSurs**

This function is responsible for training the sub-surrogate models.

***FUNCTION*** *TrainSubSurs(X, y, Dm, Wm, dm, Options)*
// INPUTS: X, y, Dm, Wm, dm, and options (NSM, rmsetrn, trnRatio, elu, trainTime, useParallel, useGPU, KrgOptimizer, KrgMaxFunEvals)

*PRINT: "Nonlinear regression using the SCK method."*
*Nets = cell(NSM-1, 1)*
// Generate random parameters for the neural networks
*d = random integers in [1, Dm] of size (NSM-1, 1)*
*w = random integers in [round(Nv/2), Wm] of size (NSM-1, 1)*
*c1 = random integers in [1, dm] of size (NSM-1, 1)*
*c2 = random integers in [1, dm] of size (NSM-1, 1)*
*FOR ii from 1 to NSM-1 DO*
*D = d(ii)*
*W = w(ii)*
*C1 = c1(ii)*
*C2 = c2(ii)*
*IF C1==1 and C2==1 THEN*
*C2 = 2*
*END IF*
*PRINT: "W, D, d1, d2"*
*Nets{ii} = Train(X, y, D, W, C1, C2, ... (other options))*
*END FOR*
// Train Kriging model
*IF useParallel is "no" THEN*
*KrgPr = false*
*ELSE*
*KrgPr = true*
*END IF*
*Krg = Kriging(X, y, ... (options))*
*RETURN: Nets, Krg*
***END FUNCTION***

**Train (for neural network)**

This function is responsible for training neural networks as sub-surrogate models.

***FUNCTION*** *Train(X, y, D, W, d1, d2, Options)*
// INPUTS: X, y, D, W, d1, d2, and options (rmsetrn, trnRatio, elu, trainTime, useParallel, useGPU)
*Split data into training and testing sets.*
// Self-organizing map (SOM)
*SOM = selforgmap([d1, d2])*
*Train SOM with training input (transposed)*
*Use SOM to assign classes to training data and add as a new feature.*
*Scale the training inputs and outputs.*
// Cascaded forward network
*Net = cascadeforwardnet(HL) where HL is a vector of D layers, each with W neurons.*
*Set network parameters (performance function, training function, goal, time, etc.)*
*Initialize and train the network.*
*RETURN: structure containing Net, SOM, number of parameters, scaling parameters, training record, and weight (inverse of RMSE on test set)*
***END FUNCTION***

**Kriging**

This function handles the training of Kriging models as sub-surrogate models.

***FUNCTION*** *Kriging(X, y, Options)*
INPUTS: X, y, and options (trnRatio, elu, trainTime, useParallel, Optimizer, MaxFunEvals)
*Split data into training and testing sets.*
*Scale the training inputs and outputs.*
*Set options for fitrgp.*
*Train a Gaussian process regression (Kriging) model.*
*Scale the testing inputs and outputs.*
*Predict on the testing set and compute the weight (inverse of RMSE).*
*RETURN: structure containing the model, scaling parameters, and weight.*
***END FUNCTION***

**Predict**

This function is responsible for making predictions based on the trained surrogate model.

***FUNCTION*** *Predict(Xp, Nets, Krg)*
INPUTS: Xp (points to predict), Nets (cell array of neural network models), Krg (Kriging model)
*FOR each neural network model in Nets (if not empty) DO*
*Use SOM to assign a class to each point in Xp and add as a new feature.*
*Scale the input (including the class) using the model's scaling parameters.*
*Predict using the neural network.*
*Unscale the output and multiply by the model's weight.*
*Store the weighted prediction.*
*END FOR*
*FOR the Kriging model (if not empty) DO*
*Scale the input (without adding class) using Kriging's scaling parameters.*
*Predict using the Kriging model.*
*Unscale the output and multiply by the model's weight.*
*Store the weighted prediction.*
*END FOR*
*Combine the weighted predictions from all models by summing and then dividing by the sum of weights.*
*RETURN: combined prediction (yp)*
***END FUNCTION***

**cdfks**

The purpose of this function is to compute probabilities using the KDE method.

***FUNCTION*** *cdfks(x, S)*
INPUTS: x (point at which to evaluate CDF), S (sample)
*Compute the mean and standard deviation of S.*
*Standardize S to get z.*
*Fit a kernel distribution to z.*
*Compute the CDF at (x - mean)/std.*
*RETURN: p (CDF value), pdz (fitted distribution)*
***END FUNCTION***

**WASM**

The purpose of this function is to compute probabilities using the WASM approach.

***FUNCTION*** *WASM(Nets, Krg, Pdfs, b0, Ns)*
INPUTS: Nets, Krg, Pdfs, b0 (initial beta), Ns (number of samples)
*IF b0 < 2.5 THEN*
*b = b0*
*pf = CDF of standard normal at -b*
*RETURN b, pf*
*ELSE*
*Set p0 based on b0 (a very small probability)*
*FOR each variable, define a uniform distribution between the p0 and 1-p0 quantiles of the original distribution.*

*Generate Ns samples from the uniform distributions.*
*Predict the output for these samples using the surrogate models.*
*FOR each sample DO*
*IF predicted output <= 0 THEN*
*I = 1*
*ELSE*
*I = 0*
*END IF*
*Compute the weight W as the product of the original PDFs evaluated at the sample.*
*END FOR*
*Compute pf = sum(I * W) / sum(W)*
*b = inverse CDF of standard normal for (1 - pf)*
*RETURN b, pf*
***END FUNCTION***

## REFERENCES


- Ahmadi-Nedushan, B., Akbarzaghi, M., & Tajammolian, H. (2025). Computational Efficiency in Triple Friction Pendulum Bearing Design Optimization Using Surrogate Modeling. *Journal of Earthquake Engineering*, *29*(7), 1444-1468. https://doi.org/10.1080/13632469.2025.2467322.
- Akbarzaghi, M., Ahmadi-Nedushan, B., & Tajammolian, H. (2024). Optimization of triple-friction pendulum isolator using Kriging as a surrogate model under El Centro earthquake. *Journal of Structural and Construction Engineering*, *11*(9), 143-162. https://doi.org/10.22065/jsce.2024.421572.3246.
- Allaix, D. L., & Carbone, V. I. (2011). An improvement of the response surface method. *Structural safety*, *33*(2), 165-172. https://doi.org/10.1016/j.strusafe.2011.02.001
- Basudhar, A., & Missoum, S. (2008). Adaptive explicit decision functions for probabilistic design and optimization using support vector machines. *Computers & Structures*, *86*(19-20), 1904-1917. https://doi.org/10.1016/j.compstruc.2008.02.008
- Beichl, I., & Sullivan, F. (2002). The metropolis algorithm. *Computing in Science & Engineering*, *2*(1), 65-69. https://doi.org/10.1109/5992.814660
- Bichon, B. J., Eldred, M. S., Swiler, L. P., Mahadevan, S., & McFarland, J. M. (2008). Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA journal*, *46*(10), 2459-2468. https://doi.org/10.2514/1.34321
- Bourinet, J. M., Deheeger, F., & Lemaire, M. (2011). Assessing small failure probabilities by combined subset simulation and support vector machines. *Structural Safety*, *33*(6), 343-353. https://doi.org/10.1016/j.strusafe.2011.06.001
- Bowman, A. W., & Azzalini, A. (1997). *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations* (Vol. 18). OUP Oxford.
- Bucher, C. G., & Bourgund, U. (1990). A fast and efficient response surface approach for structural reliability problems. *Structural safety*, *7*(1), 57-66. https://doi.org/10.1016/0167-4730(90)90012-E
- Cheng, J., Li, Q. S., & Xiao, R. C. (2008). A new artificial neural network-based response surface method for structural reliability analysis. *Probabilistic Engineering Mechanics*, *23*(1), 51-63. https://doi.org/10.1016/j.probengmech.2007.10.003
- Cheng, J., Zhang, J., Cai, C. S., & Xiao, R. C. (2007). A new approach for solving inverse reliability problems with implicit response functions. *Engineering structures*, *29*(1), 71-79. https://doi.org/10.1016/j.engstruct.2006.04.005
- Deng, J. (2006). Structural reliability analysis for implicit performance function using radial basis function network. *International journal of solids and structures*, *43*(11-12), 3255-3291. https://doi.org/10.1016/j.ijsolstr.2005.05.055
- Deng, J., Gu, D., Li, X., & Yue, Z. Q. (2005). Structural reliability analysis for implicit performance functions using artificial neural network. *Structural safety*, *27*(1), 25-48. https://doi.org/10.1016/j.strusafe.2004.03.004
- Echard, B., Gayton, N., Lemaire, M., & Relun, N. (2013). A combined importance sampling and kriging reliability method for small failure probabilities with time-demanding numerical models. *Reliability Engineering & System Safety*, *111*, 232-240. https://doi.org/10.1016/j.ress.2012.10.008
- Elhewy, A. H., Mesbahi, E., & Pu, Y. (2006). Reliability analysis of structures using neural network method. *Probabilistic Engineering Mechanics*, *21*(1), 44-53. https://doi.org/10.1016/j.probengmech.2005.07.002
- Hasofer, A. M., & Lind, N. C. (1974). Exact and invariant second-moment code format. *Journal of the Engineering Mechanics division*, *100*(1), 111-121. https://doi.org/10.1061/JMCEA3.0001848
- Huang, B., & Du, X. (2006). Uncertainty analysis by dimension reduction integration and saddlepoint approximations. *Journal of Mechanical Design*, *128*(1), 26-33. https://doi.org/10.1115/1.2118667
- Huang, X., & Zhang, Y. (2013). Reliability–sensitivity analysis using dimension reduction methods and saddlepoint approximations. *International Journal for Numerical Methods in Engineering*, *93*(8), 857-886. https://doi.org/10.1002/nme.4412
- Hurtado, J. E. (2004). An examination of methods for approximating implicit limit state functions from the viewpoint of statistical learning theory. *Structural Safety*, *26*(3), 271-293. https://doi.org/10.1016/j.strusafe.2003.05.002

- Jahangiri, M., Ahmadi-Nedushan, B., & Rahimi Bondarabadi, H. (2015). Structural damage localization and quantification based on multi-objective optimization method. *In 2nd International & 6th National Conference on Earthquake & Structures, At ACECR of Kerman, Kerman, Iran*.
- Javanmardi, R., & Ahmadi-Nedushan, B. (2021). Cost optimization of steel-concrete composite I-girder bridges with skew angle and longitudinal slope, using the Sm toolbox and the parallel pattern search algorithm. *Int. J. Optim. Civil Eng*, *11*(3), 357-382.
- Javanmardi, R., & Ahmadi-Nedushan, B. (2023). Optimal design of double-layer barrel vaults using genetic and pattern search algorithms and optimized neural network as surrogate model. *Frontiers of Structural and Civil Engineering*, *17*(3), 378-395. https://doi.org/10.1007/s11709-022-0899-9
- Johari, M., & Ahmadi Nedushan, B. (2017). Reliability-based topology optimization of bridge structures using first and second order reliability methods. *Journal of Structural and Construction Engineering*, *4*(1), 5-18. https://doi.org/10.22065/jsce.2016.40432.
- Kaymaz, I. (2005). Application of kriging method to structural reliability problems. *Structural safety*, *27*(2), 133-151. https://doi.org/10.1016/j.strusafe.2004.09.001
- Kaymaz, I., & McMahon, C. A. (2005). A response surface method based on weighted regression for structural reliability analysis. *Probabilistic Engineering Mechanics*, *20*(1), 11-17. https://doi.org/10.1016/j.probengmech.2004.05.005
- Keshtegar, B. (2017). A hybrid conjugate finite-step length method for robust and efficient reliability analysis. *Applied Mathematical Modelling*, *45*, 226-237. https://doi.org/10.1016/j.apm.2016.12.027
- Keshtegar, B. (2017). Limited conjugate gradient method for structural reliability analysis. *Engineering with Computers*, *33*(3), 621-629. https://doi.org/10.1007/s00366-016-0493-7
- Keshtegar, B., & Kisi, O. (2017). M5 model tree and Monte Carlo simulation for efficient structural reliability analysis. *Applied Mathematical Modelling*, *48*, 899-910. https://doi.org/10.1016/j.apm.2017.02.047
- Keshtegar, B., & Sadegh, M. O. (2015, September). Instabilities control of reliability analysis using hybrid HL-RF and conjugate optimization algorithms. In *2015 4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)* (pp. 1-4). IEEE. https://doi.org/10.1109/CFIS.2015.7391656
- Kingston, G. B., Rajabalinejad, M., Gouldby, B. P., & Van Gelder, P. H. (2011). Computational intelligence methods for the efficient reliability analysis of complex flood defence structures. *Structural Safety*, *33*(1), 64-73. https://doi.org/10.1016/j.strusafe.2010.08.002
- Kleiber, M., Knabel, J., & Rojek, J. (2004). Response surface method for probabilistic assessment of metal forming failures. *International journal for numerical methods in engineering*, *60*(1), 51-67. https://doi.org/10.1002/nme.954
- Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, *52*(6), 119-139.
- Lim, J., Lee, B., & Lee, I. (2014). Second-order reliability method-based inverse reliability analysis using Hessian update for accurate and efficient reliability-based design optimization. *International Journal for Numerical Methods in Engineering*, *100*(10), 773-792. https://doi.org/10.1002/nme.4775
- Liping, W., & Grandhi, R. V. (1994). Efficient safety index calculation for structural reliability analysis. *Computers & structures*, *52*(1), 103-111. https://doi.org/10.1016/0045-7949(94)90260-7
- Lophaven, S. N., Nielsen, H. B., Sondergaard, J., & Dace, A. (2002). A matlab kriging toolbox. *Technical University of Denmark, Kongens Lyngby, Technical Report No. IMM-TR-2002-12*.
- MathWorks. MATLAB Documantation. Natick: MathWorks, 2019
- Nowak, A. S., & Collins, K. R. (2012). *Reliability of structures*. CRC press.
- Papadopoulos, V., Giovanis, D. G., Lagaros, N. D., & Papadrakakis, M. (2012). Accelerated subset simulation with neural networks for reliability analysis. *Computer Methods in Applied Mechanics and Engineering*, *223*, 70-80. https://doi.org/10.1016/j.cma.2012.02.013
- Rashki, M., Miri, M., & Moghaddam, M. A. (2012). A new efficient simulation method to approximate the probability of failure and most probable point. *Structural Safety*, *39*, 22-29. https://doi.org/10.1016/j.strusafe.2012.06.003
- Rocco, C. M., & Moreno, J. A. (2002). Fast Monte Carlo reliability evaluation using support vector machine. *Reliability Engineering & System Safety*, *76*(3), 237-243. https://doi.org/10.1016/S0951-8320(02)00015-7
- Shayanfar, M. A., Barkhordari, M. A., & Roudak, M. A. (2017). An adaptive importance sampling-based algorithm using the first order method for structural reliability. *Int. J. Optim. Civil Eng*, *7*(1), 93-107.
- Viana, F. A., Haftka, R. T., & Steffen Jr, V. (2009). Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, *39*(4), 439-457. https://doi.org/10.1007/s00158-008-0338-0
- Viana, F. A., Haftka, R. T., & Watson, L. T. (2013). Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global*

*Optimization*, *56*(2), 669-689. https://doi.org/10.1007/s10898-012-9892-5

- Zhang, J., Xiao, M., & Gao, L. (2019). An active learning reliability method combining Kriging constructed with exploration and exploitation of failure region and subset simulation. *Reliability Engineering & System Safety*, *188*, 90-102. https://doi.org/10.1016/j.ress.2019.03.002
- Zhao, H., Yue, Z., Liu, Y., Gao, Z., & Zhang, Y. (2015). An efficient reliability method combining adaptive importance sampling and Kriging metamodel. *Applied Mathematical Modelling*, *39*(7), 1853-1866. https://doi.org/10.1016/j.apm.2014.10.015