Structural Reliability Analysis Using Artificial Neural Networks and Bootstrap Techniques

Wellison J. S. Gomes

1Center for Optimization and Reliability in Engineering (CORE), Department of Civil Engineering, Federal University of Santa Catarina. Rua João Pio Duarte, 205, Córrego Grande, 88037-000, Florianópolis, SC, Brazil. wellison.gomes@ufsc.br

Keywords. Structural reliability, reliability analysis, surrogate models, artificial neural networks, bootstrap techniques

Abstract. Structural reliability theory has been applied to many engineering problems in the last decades, to deal with the uncertainties which affect engineering structures and quantify the safety of such structures. In some cases, approximated methods such as FORM and SORM (First and Second Order Reliability Methods, respectively) may be applied, allowing estimating failure probabilities at acceptable computational costs. However, for more complex problems, e.g. those involving highly nonlinear limit state functions or combinations of several limit states, many times the only alternatives available are those involving more demanding approaches, such as the Monte Carlo Simulation Methods. In these cases, the computational burden may easily become prohibitive and surrogate models have been widely employed as an attempt to keep the computational effort acceptable. In the present paper, an adaptive approach for reliability analysis using surrogate models, previously proposed in the literature in the context of Kriging and Polynomial Chaos Expansions, is adapted for the case of multi-layer perceptron artificial neural networks (ANNs). The approach combines the bootstrap resampling method and adaptive experimental designs to improve the efficiency and quality of the surrogate model. It focuses on those points in the random variables space which are more significant to the construction of the surrogate for reliability analysis purposes and tries to use as few evaluations as possible of the original model to be approximated. The methodology is employed in the solution of three benchmark reliability problems and compared to simple Monte Carlo simulation. As expected, in all cases the ANNs led to results very close to those obtained by MCS, and required much less evaluations of the limit state function. These results confirmed the applicability of the methodology to the reliability analysis of problems involving small failure probabilities and also allowed to identify aspects of the methodology which could be further improved in future studies.

1 INTRODUCTION

Structural reliability theory has been applied to many engineering problems in the last decades, to deal with the uncertainties which affect engineering structures and quantify the safety of such structures (Madsen, Skjong, and Tallin, 1987; Biondini et al., 2004; Estes, Frangopol and Foltz, 2004; Hart et al., 2014). Approximated methods such as FORM and SORM (First and Second Order Reliability Methods, respectively) have been among the most used ones, allowing estimating failure probabilities at acceptable computational costs. For more complex problems, e.g. those involving highly nonlinear limit state functions or combinations of several limit states, many times the only alternatives available are those involving more demanding approaches, such as the Monte Carlo Methods. In these cases, the computational burden may easily become prohibitive and surrogate models, also known as metamodels, have been widely employed as an attempt to keep the computational effort acceptable.

The basic idea of surrogate modelling for reliability analysis purposes is usually to replace the true time-consuming limit state function by an approximation. Among the surrogate models available in the literature are the Response Surface Method (Faravelli, 1989; Soares et al., 2002), Kriging (Stein, 1999; Dubourg, Sudret, and Deheeger, 2013), Polynomial Chaos Expansions (Ghanem and Spanos, 1991; Blatman and Sudret, 2010) and Artificial Neural Networks (Gomes and Beck, 2013; Bucher and Most, 2008). Although Artificial Neural Networks (ANNs) are one of the many methods available, they have been attracting research interest for many years, in the field of structural reliability and many other fields. A review of applications of ANNs in structural reliability analysis may be found in Chojaczyk et al. (2015).

The applications found in the literature include, but are not limited to those which: a) focus on the combination of ANNs with structural reliability methods such as Monte Carlo Simulation (Papadrakakis and Lagaros, 2002), Subset Simulation (Papadopoulos et al., 2012), FORM and SORM (Deng et al., 2005); b) compare ANNs with other surrogate models, for example Response Surface Method (Gomes and Awruch, 2004; Bucher and Most, 2008) and Support Vector Machine (Tan et al., 2011); and c) applies ANNs to specific problems with significant computational cost, for example, problems dealing with uncertainties and optimization at the same time (Gomes, Awruch and Lopes, 2011; Gomes and Beck, 2013).
In some cases, the metamodel is constructed considering a single experimental design, that is, a fixed number of points on which the true limit state function is evaluated. Then, the reliability analysis is performed using the previously constructed metamodel. However, a general tendency in surrogate modelling has been the use of adaptive experimental designs, which allows refining the experimental design during the reliability process, increasing the efficiency of the metamodeling approach by choosing carefully a design enrichment strategy. Some procedures for adapting and refining ANNs during the reliability analysis have been already proposed in the literature (Shao and Murotsu, 1997; Schueremans and Vangemert, 2005).

In the present paper, an adaptive approach proposed by Marelli and Sudret (2016) for polynomial chaos expansions (PCEs), based on a previous paper by Echard, Gayton and Lemaire (2011), is adapted for the case of multi-layer perceptron (MLP) artificial neural networks. The approach combines the bootstrap resampling method and adaptive experimental designs to improve the efficiency and quality of the surrogate model. Details about the application of this approach to the case of ANNs are detailed and discussed. Then, the methodology is employed in the solution of three benchmark reliability problems and compared to simple Monte Carlo simulation (MCS).

The remainder of this paper is organized as follows. Section 2 gives a brief description of some basic concepts of structural reliability. ANNs and the adaptive approach are presented in Section 3. In section 4, three benchmark problems in structural reliability are presented and solved by using adaptive ANNs and by MCS. Finally, some concluding remarks are drawn in Section 5.

2 STRUCTURAL RELIABILITY

Let $\mathbf{X}$ be a vector of random variables which represents all random or uncertain parameters of a structural system. Existence of uncertainty implies possibilities of undesirable structural responses. The boundary between desirable and undesirable structural responses is defined by limit state functions, $g(\mathbf{X})$, in such a way that failure and survival domains, $\Omega_f$ and $\Omega_s$, respectively, are given by:

$$
\Omega_f = \{ \mathbf{x} | g(\mathbf{x}) \leq 0 \}
$$

$$
\Omega_s = \{ \mathbf{x} | g(\mathbf{x}) > 0 \}
$$

Each limit state describes one possible failure mode of the structure. For each failure mode, the probability of undesirable structural responses, usually known as probability of failure, is defined as:

$$
P_f = P[\mathbf{X} \in \Omega_f] = \int_{\Omega_f} f_\mathbf{X}(\mathbf{x}) d\mathbf{x}
$$

where $f_\mathbf{X}(\mathbf{x})$ is the joint probability density function of vector $\mathbf{X}$. Equation (2) may also be employed to compute failure probabilities of structural systems; in this case, $\Omega_f$ must be defined as a combination of all limit state functions involved.

The multidimensional integral in Eq. (2) may be solved by means of structural reliability methods such as First and Second Order Reliability Methods (FORM and SORM) and Monte Carlo simulation (Madsen, Krenk and Lind, 1986; Melchers, 1999).

When simple Monte Carlo simulation is employed, failure probabilities can be estimated via Eq. (3), by randomly generating $n_{MC}$ samples of $\mathbf{X}$ according to its joint distribution $f_\mathbf{X}(\mathbf{x})$ and by considering a so-called indicator function, $I[\mathbf{x}]$, which is equal to one if $\mathbf{x}$ belongs to the failure domain and zero otherwise. It is noted that application of Eq. (3) requires one limit state function evaluation per sample. Besides, the smaller the failure probability to be estimated, the larger the number of samples required to achieve an acceptable accuracy. As engineering structures usually present very small probabilities of failure, the computational burden easily becomes prohibitive.

$$
P_f = E[I[\mathbf{X}]] \approx \frac{1}{n_{MC}} \sum_{i=1}^{n_{MC}} I[\mathbf{x}_i]
$$

3 ARTIFICIAL NEURAL NETWORKS AND ADAPTIVE DESIGNS

3.1 Artificial Neural Networks

Artificial Neural Networks were introduced as universal function approximators by McCulloch and Pitts in 1943 (McCulloch and Pitts, 1943) and have been extensively used in many areas ever since. They are computational models based on a simplified analogy to the behavior of the human brain, where information is processed by small processing units, corresponding to the neurons, which are mathematically represented by relatively simple yet flexible functions,
such as linear or sigmoid functions (Bucher and Most, 2008). These processing units communicate with each other by means of weighted connections, corresponding to the synapses of the brain (Gomes and Beck, 2013).

Different networks can be constructed by choosing different numbers of neuron layers, the type and number of neurons in each layer, and the type of connection between neurons. For a specific configuration of the network and for a given set of input-output data, the so-called training of the network consists of adjusting its parameters in order for the network to reproduce the input-output data as accurately as possible. Each iteration of the training process is called an epoch.

It is common procedure to separate input–output data into three groups: training, validation and test data. The first group is the only one used for adjustment of the parameters; validation is used between epochs, in order to halt the training process if the network error starts to increase due to overfitting; and the test data is used to verify the networks predicting capacity at the end of training (Gomes and Beck, 2013). When the network is updated in an iterative manner, which is the case for adaptive experimental designs, it is common not to use the test set; in some cases, not even the validation set is used, assuming that the number of epochs is sufficiently small or employing another strategy to avoid overfitting.

The most widely used network type for approximation problems, which is adopted herein, is the multi-layer perceptron (MLP), also called feed-forward backpropagation network (Bucher and Most, 2008). MLP networks are built with: one input layer, with one neuron for each input parameter; one output layer, with one neuron for each output parameter; and an arbitrary number of hidden layers. The neurons of one layer are connected with each neuron of the previous layer, but information only flows in the forward direction, from the input towards the output layer.

According to Bucher and Most (2008), for approximation of functions with minor discontinuities generally a combination of layers with sigmoid activation functions and a linear output layer is used. Furthermore, it has been proved (Hornik, Stinchcombe and White, 1990) that multilayer feed-forward networks with as few as a single hidden layer and an appropriately smooth hidden layer activation function are capable of approximating any arbitrary function and its derivatives. Hence, in the present implementation only one hidden layer is employed. The number of neurons \((n_2)\) in this layer may follow the Kolmogorov–Nielsen (Hecht-Nielsen, 1989) theorem:

\[
n_2 = \alpha \cdot n + 1
\]

where \(n\) is the number of input parameters of the problem and \(\alpha = 2\).

The type of neuron in each layer is defined by the chosen activation functions. In this paper, linear activation functions are used for the input and output layers, and tangent-sigmoid neurons are chosen for the hidden layer. Input data is first processed by the neurons in the input layer. Each neuron receives one element of the input vector and transmits an impulse to all neurons of the next layer. If the activation function of the input layer is linear, input values are simply passed onto the neurons in the next layer. In the passage, each value is multiplied by the respective weight, which characterizes the connection between neurons of the layers; hence a weighted sum is passed to the hidden layer.

The approximation of the Hessian matrix is an other method for second-order training speed using an approximation of the Hessian matrix (Beale, Hagan and Demuth, 2011), without calculation of the Hessian matrix itself. When the performance function of the ANN has the form of a sum of squares, as is typical in training feedforward networks, the Hessian matrix can be approximated as:

\[
y_{ANN} = \theta^{(3)}_1 + \sum_{i=1}^{n_2} (w^{(3)}_{i1} \cdot \text{tansig} (\theta^{(2)}_i + \sum_{j=1}^{n_1} w^{(2)}_{ji} x_j))
\]

where \(y_{ANN}\) is the network output and \(\text{tansig}(.)\) is the tangent sigmoid function.

In this paper, the MATLAB neural network toolbox (Beale, Hagan and Demuth, 2011) is employed for generating, training and using the ANNs. The MLP network is trained by using the Levenberg–Marquardt method, a modification of the Gauss–Newton method (Hagan and Menhaj, 1994) based on the least squares method for nonlinear models. Like other quasi-Newton methods, this method was designed to approach second-order training speed using an approximation of the Hessian matrix (Beale, Hagan and Demuth, 2011), without calculation of the Hessian matrix itself. When the performance function of the ANN has the form of a sum of squares, as is typical in training feedforward networks, the Hessian matrix can be approximated as:
\[ H = J^T J \]  

where \( J \) is the Jacobian matrix, which contains first derivatives of the network errors with respect to the parameters of the ANN.

The Levenberg-Marquardt algorithm uses this approximation, resulting in the following updating function:

\[ W_{k+1} = W_k - (J^T J + \mu I)^{-1} J^T e \]  

where \( I \) is the identity matrix, \( e \) is a vector of network errors, \( W \) contains both the weights and biases of the ANN, \( \mu \) is a scalar, a parameter of the algorithm, and \( k \) represents the current training epoch. Also, if a better approximation is required for some regions of the output space, for example, the vector of network errors may be weighted by using a vector \( EW \) which contains the error weights. In this case, \( e \) is multiplied component-wise by \( EW \).

One notes that the parameters of the ANN are adjusted in the opposite direction to the gradient of the error function, in an attempt to decrease the error at each epoch. When the scalar \( \mu \) is zero, the conventional Newton’s method is recovered. When \( \mu \) is large, the equation results in a gradient descent scheme, with a small step size. Thus, \( \mu \) is usually decreased after a successful step, which is a step where the error is reduced, and increased otherwise. The default value for \( \mu \) is 0.001, with decreasing and increasing factors equal to 0.1 and 10, respectively.

In some cases, improvements of the network capabilities may be achieved by scaling the input and/or the output data considering given ranges. By default, the MATLAB toolbox processes the input data by using the mapminmax function, which normalizes the data so that all inputs fall in the range \([-1, 1]\). The same function can also be applied to normalize the output data, and the ranges may be specified by the user.

Further details about ANNs can be obtained, for example, in Haykin (1999). Details about the MATLAB neural network toolbox can be found in Beale, Hagan and Demuth (2011).

### 3.2 Adaptive Artificial Neural Network for reliability analysis

When using surrogate models for limit state function approximation in reliability analysis, an experimental design (ED), consisting of \( n_{ED} \) points \( \{x^{(1)}_{ED}, x^{(2)}_{ED}, \ldots, x^{(n_{ED})}_{ED}\} \), with \( x^{(i)}_{ED} \in \mathbb{R}^n \), and the respective function evaluation values, \( y^{(i)}_{ED} = g(x^{(i)}_{ED}) \in \mathbb{R} \), is used to construct the approximation. After that, Eq. (3) is solved by using the surrogate to evaluate the indicator function for each one of the \( n_{MC} \) samples. In this case, the true limit state function needs to be evaluated only \( n_{ED} \) times, where \( n_{ED} \) is the size of the ED.

Noting that the limit state function serves the sole purpose of defining the bounds of integration in Eq. (2), these models aim at approximating the limit state function to obtain a sufficiently accurate prediction of its sign, particularly in the region which contributes most to the failure probability (Bucher and Most, 2008; Echard, Gayton and Lemaire, 2011). This led the path to the use of active learning and adaptive designs of experiments, which allows for adapting and refining the surrogate during the reliability analysis, until a required accuracy is reached.

The methodology used herein is based on Echard, Gayton and Lemaire (2011), where an active learning reliability method combining Kriging and Monte Carlo Simulation was proposed, and on Marelli and Sudret (2016), where the method by Echard, Gayton and Lemaire (2011) was adapted for the case of Polynomial Chaos Expansions. While in the former Kriging provides not only predicted values but also estimations of the local variance of the predictions, which are directly used for the selection of new points to be added to the design of experiments, in the latter no such estimators are directly available, in such a way that Marelli and Sudret (2016) proposed an alternative approach, which uses the bootstrap resampling method (Efron, 1992; Efron and Tibshirani, 1993) to derive similar local error estimates. In other words, instead of using only one surrogate during the reliability analysis, the approach proposed by Marelli and Sudret (2016) employs a number of surrogates, and these surrogates are used to determine which points should be added to the ED in order to improve the failure probability estimates.

Thus, the following procedure, very similar to the one proposed by Echard, Gayton and Lemaire (2011) and considering some of the adaptations presented in Marelli and Sudret (2016), is adopted. The procedure, which consists of 7 stages and was implemented in MATLAB, is presented first in a general way. Then, in the next sections, details about the implementation involving ANNs are discussed.

1. **Generation of a Monte Carlo population in the random variables space.** A population of \( n_{MC} \) samples of \( X \) is randomly generated, according to the joint distribution \( f_X(x) \). This population remains the same during the whole learning process and the points which constitute the design of experiments are always selected from it.
2. **Definition of the initial experimental design.** The initial ED comprises \( n_{ED} \) points selected from the population and includes the respective limit state function evaluations.
3. **Initialization of the ANNs.** A total of \( B \) ANNs are generated and trained considering the initial ED.
4. **Prediction by ANN and estimation of the probability of failure.** ANN predictions are obtained for the entire population by Eq. (5). Then, failure probability estimates are obtained for each surrogate, by dividing the
number of points with a negative of null ANN prediction by \( n_{MC} \). For a given surrogate, \( b \), where \( b = 1, 2, ..., B \), the failure probability is given by:

\[
P_f^{(b)} = \frac{n_{ANN \leq 0}}{n_{MC}} \tag{8}
\]

5. **Evaluation of the convergence criterion.** A convergence criterion based on \( P_f^{(b)} \) is evaluated. If the criterion is met, the algorithm ends and the \( P_f \) to be returned is the average of all failure probabilities given by Eq. (8). Otherwise, the algorithm continues.

6. **Identification of the points to be included on the ED.** A suitable learning function is evaluated on the population. One or more points belonging to the population are chosen according to their learning function values. The limit state function is evaluated on these points and the new information is added to the ED;

7. **Updating of the ANNs.** The ANNs are trained again, using the enriched ED. The algorithm returns to step 4.

3.2.1 Initialization of the ANNs

Considering that a higher number of neurons increases the network’s approximation capabilities but may lead to overfitting, the B ANNs are divided into three groups, each one with a different number of neurons in the hidden layer, \( n_z \). The number of neurons in each group is defined by Eq. (4), taking \( n \) as the number of random variables of the problem and considering three different values for the parameter \( \alpha \), that is, \( \alpha = [0.5, 1.0, 2.0] \), respectively. This, together with the updating process of the ANNs, allows for a simple adaptation of the number of neurons according to the complexity of the problem at hand.

Initialization of backpropagation networks is usually performed by the Nguyen-Widrow method (Nguyen and Widrow, 1990). For each layer, initial values of weights and biases are chosen in such a way that the active regions of the neurons result approximately evenly distributed across the input space of the layer. These values represent the initial point of a multidimensional optimization problem (the training of the network), and they contain a degree of randomness. Thus, it is common to initialize the network many times to try to avoid local minima. For this reason, for each group of ANNs, a total of \( 5 \times B \) networks are initialized and trained and the one presenting the best performance on the test dataset is chosen. After that, all ANNs of each group are initially defined as copies of the respective chosen network. At this stage, 70% of the initial ED is used for training, 15% for validation and the remaining 15% for test; a maximum of 100 epochs is allowed.

Note that the proposed initialization scheme leads to only three different ANNs, but all of them should be different for active learning purposes. Considering this, the bootstrap resampling method is applied to the initial ED to generate \( B \) distinct initial designs. This leads to \( B \) different bootstrap-surrogates after further training of the initial ANNs is performed. A minimum number of iterations of the entire adaptive process, say 3, is adopted to ensure that the required diversity is achieved.

3.2.2 Initial experimental design

In order to improve the space-filling properties of the initial experimental design, selection is performed by means of a clustering procedure applied to the entire population. The \texttt{kmeans} MATLAB function, an implementation of the \texttt{k}-means clustering method (Haykin, 1996), is employed, considering a total of \( n_{ED} \) clusters. The squared Euclidean distance is taken as the distance measure to be minimized and five iterations are allowed. The selected points are those nearest to the center of each cluster. According to Echard, Gayton and Lemaire (2011), a dozen of points are enough. In all cases studied in Section 4, \( n_{ED} \) is adopted equal to 50.

3.2.3 Convergence criteria

The convergence criterion chosen was developed by Schöbi, Sudret and Marelli (2017) and used also by Marelli and Sudret (2016). It is based on the stability of the estimated failure probability, \( \hat{P}_f \), at the current iteration, and is given by:

\[
\frac{\hat{P}_f^+ + \hat{P}_f^-}{\hat{P}_f} \leq \epsilon_{P_f} \tag{9}
\]

where \( \hat{P}_f^+ \) and \( \hat{P}_f^- \) are the highest and lowest estimates computed by using the \( B \) ANNs at the current iteration, defined by:

\[
\hat{P}_f^- = \min(P_f^{(b)}) \quad \hat{P}_f^+ = \max(P_f^{(b)}) \tag{10}
\]

and the tolerance, \( \epsilon_{P_f} \), is taken as 0.5% in the present paper.
Convergence is achieved when the criterion is satisfied in the current iteration.

3.2.4 Learning function and enrichment of the ED

The learning function adopted is the one proposed by Marelli and Sudret (2016), related to the misclassification probability of the population samples. According to them, it is straightforward to define the fraction of failed bootstrap replicates (FBR), \( U_{FBR}(x^{(i)}) \), as follows:

\[
U_{FBR}(x^{(i)}) = \left| \frac{B_S(x^{(i)}) - B_f(x^{(i)})}{B} \right|
\]  

(11)

where \( B_S(x^{(i)}) \) and \( B_f(x^{(i)}) \) are the number of surrogates which identify the sample \( x^{(i)} \) as being in the safe region and in the failure region, respectively. If \( U_{FBR}(x^{(i)}) = 1 \), the classification of the sample is resulting the same for all surrogates. On the other hand, if \( U_{FBR}(x^{(i)}) \) is close to zero, there is a great amount of epistemic uncertainty in the classification of point \( x^{(i)} \), and information about this point should be added to the ED.

In the present paper, in order to add \( n_{ADD} \) points to the ED at each iteration, the population is clustered in \( n_{ADD} \) different regions by using the k-means clustering method. Note that, because the population does not change during the learning process, the definition of the clusters needs to be done just once, during the first iteration. Then, \( U_{FBR} \) is evaluated on the entire population and one point of each cluster is selected, among those presenting the smallest value of \( U_{FBR} \). These points are directly added to all initial EDs. Since they are located in important regions of the random variables space, bootstrap resampling is not applied to them.

3.2.5 Error function for comparison of ANNs

Although the ANN training via Levenberg-Marquardt method requires the use of a performance function in the form of a sum of squares, another error function, specifically defined for reliability analysis purposes, may be stabilised to be used during the updating phase. For this purpose, let us first compute the squared error of the ANN considering the ED:

\[
y_{ERROR}^{(i)} = (y^{(i)}_{ANN} - y^{(i)}_{ED})^2
\]  

(12)

where \( i = 1, 2, \ldots, n_{ED} \).

This error should have a significant impact on the error function for those points of the ED for which the surrogate leads to a misclassification. However, if the surrogate is successful in determining the sign of the limit state function but predicts a different value, this error should have a minor impact. To take this into account, a vector of misclassified points is defined as:

\[
MP^{(i)} = \begin{cases} 
0, & \text{if } \text{sign}(y^{(i)}_{ANN}) = \text{sign}(y^{(i)}_{ED}) \\
1, & \text{otherwise}.
\end{cases}
\]  

(13)

Finally, the error function used to compare ANNs during the updating phase is given by:

\[
ANN_{error} = \sum_{i=1}^{n_{ED}} (MP^{(i)} \cdot y_{ERROR}^{(i)}) + \frac{1}{10^8} \sum_{i=1}^{n_{ED}} y_{ERROR}^{(i)}
\]

(14)

in such a way that the ANNs with less misclassified points are considered the better ones, and among those with the same number of misclassified points, the one with the lesser sum of squared errors is considered the best.

3.2.6 Updating of the ANNs

The updating of the ANNs is basically a new training of the existing neural networks using the enriched ED. However, it must take into account the necessity of adaptation of the number of neurons in the hidden layer as well as the fact that the ANN training is an optimization process which can easily lead to local minima. For this reason, an updating scheme consisting of two steps is proposed: a training step and a replacement step.

The training step is further divided into two substeps. In the first substep, training is performed just for one epoch, using the default values of \( \mu \) and its decreasing and increasing factors, see Eq. (7). In the second substep, a refined training is carried out, with a total of two epochs, and parameters \( \mu = 1.0 \), and decreasing and increasing factors of 0.8 and 1.5, respectively. The second substep results in a gradient descent scheme, with a step size significantly smaller than the one of the first substep. In all cases, if training does not result in increasing the performance of the ANN, given by Eq. (14), then the previous values of weights and biases are restored. This is necessary because the bootstrapped ED may not contain enough points to ensure that a better approximation for the entire limit state is obtained.
In the second step, the so-called replacement step, the ANNs are first ordered from the best to the worst, according to their $ANN_{error}$, given by Eq. (14). After that, the last $n_{REP}$ ANNs are chosen for improvement or possible replacement and the first $n_{REP}$ ANNs are chosen as candidates to replace the worst ones. The $k^{th}$ worst ANN and the $k^{th}$ best ANN are both temporarily trained for 200 epochs, considering the ED related to the $k^{th}$ worst; this generates two auxiliary ANNs. The ANN which is chosen to succeed the $k^{th}$ worst is the one which presents the best performance, among the two auxiliary and the $k^{th}$ worst itself.

During these two steps, all the information contained in the ED is used for training, that is, no validation or test sets are considered. This is the main reason why a small number of epochs is used during the first step. On the other hand, in the second step many more epochs are allowed because there is a need to improve the performance of the worst surrogates in order to increase the overall accuracy of the estimates.

3.2.7 Scaling of the data and error weights

In all cases, the data presented to the ANNs is scaled before any training or evaluation takes place, that is, right before the initialization of the networks. Scaling of the input data is performed considering maximum and minimum values of each random variable, computed from the population; for the output data, an interval defined by the maximum value of the limit state function for the initial ED, $[-\gamma_{max}, \gamma_{max}]$, is considered. The maximum value is given by:

$$\gamma_{max} = \max(y_{ED}^{(1)}, y_{ED}^{(2)}, \ldots, y_{ED}^{(n_{ED})})$$

(15)

Another important aspect related to the application of ANNs to reliability analysis concerns the fact that the most important points of the ED are those for which the limit state function value is closer to zero. For this reason, error weights, $EW^{(i)}$, given by Eq. (16), are applied to the Levenberg-Marquardt training algorithm.

$$EW^{(i)} = \min \left( \frac{1}{y_{ED}^{(i)}}, 10^5 \right)$$

(16)

4 NUMERICAL EXAMPLES

In this section, validation of the ANNs and the adaptive algorithm is made by applying them to solve three benchmark reliability problems. The results obtained herein are compared with those obtained by MCS. Both ANNs and MCS solutions use the same population.

For all examples, the size of the initial ED and the number of samples to be added to the ED at each enrichment step are kept constant, with $n_{ED} = 50$ and $n_{ADD} = 3$. Also, a fixed number of surrogates, $B=50$, is adopted. The number of ANNs to be possibly replaced in each iteration is $n_{REP} = 5$, for examples 1 and 2, and $n_{REP} = 2$, for the third example. Comparison of the computational effort is performed by using the number of calls to the limit state function, $n_{CLS}$, required by each method. Although the computational cost for training the ANNs may be significant for these examples, the use of surrogates aims at complex problems, where each evaluation of the limit state function is extremely time-consuming in comparison to the construction of the surrogate.

4.1 Example 1: Series system with four branches

This example consists of a series system with four branches, originally proposed in Waarts (2000), but also studied by Echard, Gayton and Lemaire (2011), Marelli and Sudret (2016), Schueremans and Vangemert (2005) and other authors. Two standard normal distributed random variables are considered and the limit state function reads:

$$g(x_1, x_2) = \min \left\{ \frac{3 + 0.1(x_1 - x_2)^2 - (x_1 + x_2)}{\sqrt{2}}, \frac{3 + 0.1(x_1 - x_2)^2 + (x_1 + x_2)}{\sqrt{2}}, \frac{(x_1 - x_2) + k}{\sqrt{2}}, \frac{(x_2 - x_1) + k}{\sqrt{2}} \right\}$$

(17)

where $k$ is taken as equal to 6, as in Marelli and Sudret (2016).

Table 1 presents the reference failure probability, computed by Monte Carlo simulation, as well as the results obtained by the adaptive ANNs. These results are in accordance with those presented in the literature (Schöbi, Sudret and Marelli, 2017), where the failure probability was obtained as $4.46 \times 10^{-2}$, for MCS, using $10^5$ samples.
Table 1. Reliability results for example 1.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\hat{P}_f$</th>
<th>$n_{CLS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$4.445\times10^{-3}$</td>
<td>$5\times10^6$</td>
</tr>
<tr>
<td>ANN</td>
<td>$4.4493\times10^{-3}$</td>
<td>$128$</td>
</tr>
</tbody>
</table>

The initial experimental design and the points added to it during the adaptive procedure are presented in Fig. 1a. The convergence of the estimated failure probability is illustrated in Fig. 1b. In this case, at the end of the iterative process, after 27 iterations, all ANNs presented a number of neurons in the hidden layer equal to 5, which is obtained by Eq. (4) for $\alpha = 2$ and two random variables ($n=2$). In other words, the replacement procedure leads all ANNs to the third group of surrogates.

Figure 1a indicates that the ANNs correctly identifies the samples of the population which are closest to the limit-state equation, $g(x_1, x_2) = 0$. With a few exceptions, these are the points added to the ED during the enrichment phase. Figure 1b shows that the probability of failure estimated by ANNs approaches the reference value even for a lesser number of calls to the limit state function, say $n_{CLS} = 100$; although convergence is not achieved at this point, according to the convergence criterion adopted.

4.2 Example 2: dynamic response of a nonlinear oscillator

This example consists of a nonlinear undamped single degree of freedom system (Fig. 2), studied, for example, in Schueremans and Vangemert (2005) and Echard, Gayton and Lemaire (2011). The limit state function is defined by:

$$
g(c_1, c_2, m, r, t_1, F_1) = 3r - \frac{2F_1}{m\omega_0^2}\sin\left(\frac{\omega_0 t_1}{2}\right)
$$

(18)

where $\omega_0 = \sqrt{\frac{c_1 + c_2}{m}}$. The parameters of the six random variables are shown in Tab. 2, where P.D.F. stands for probability density function.

Figure 1. a) Limit state equations (solid lines), initial experimental design (black dots) and added points (red diamonds). b) Convergence of the failure probability over the number of limit state evaluations: $\hat{P}_f$ (solid line), $\hat{P}_f^{-1/k}$ (dotted line) and reference failure probability (horizontal dashed line).

Figure 2. Nonlinear oscillator: system definition and applied load.
Table 2. Example 2: random variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>P.D.F.</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>Normal</td>
<td>1.0</td>
<td>0.05</td>
</tr>
<tr>
<td>$c_i$</td>
<td>Normal</td>
<td>1.0</td>
<td>0.10</td>
</tr>
<tr>
<td>$c_2$</td>
<td>Normal</td>
<td>0.1</td>
<td>0.01</td>
</tr>
<tr>
<td>$r$</td>
<td>Normal</td>
<td>0.5</td>
<td>0.05</td>
</tr>
<tr>
<td>$F_1$</td>
<td>Normal</td>
<td>1.0</td>
<td>0.20</td>
</tr>
<tr>
<td>$l_2$</td>
<td>Normal</td>
<td>1.0</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Results for this case are presented in Tab. 3 and can be checked against the result obtained by Echard, Gayton and Lemaire (2011), $P_f = 2.834 \times 10^{-2}$, using MCS with the same population size.

Table 3. Reliability results for example 2.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\hat{P}_f$</th>
<th>$n_{CLS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$2.8343 \times 10^{-2}$</td>
<td>$7 \times 10^4$</td>
</tr>
<tr>
<td>ANN</td>
<td>$2.8328 \times 10^{-2}$</td>
<td>125</td>
</tr>
</tbody>
</table>

In this case, convergence was achieved after 26 iterations, with 24 ANNs at the second group ($\alpha = 1$ and $n_2 = 7$) and 26 at the third group ($\alpha = 2$ and $n_2 = 13$) of surrogates. Figure 3 illustrates the convergence of the estimated failure probability.

Figure 3. Convergence of the failure probability for the second example: $\hat{P}_f$ (solid line), $P_{f^{-/+}}$ (dotted line) and reference failure probability (horizontal dashed line).

As can be seen in Fig. 3 and Tab. 3, the result obtained via ANNs was again very close to the reference one.

4.3 Example 3: High dimensional example

The last example was proposed in Rackwitz (2001) and also studied in Echard, Gayton and Lemaire (2011). According to Echard, Gayton and Lemaire (2011), it consists of an analytical limit state function, where the number of variables can be changed without modifying significantly the level of failure probability. The limit state function is given by:

$$g(x_1, x_2, ..., x_n) = (n + 3\sigma\sqrt{n}) - \sum_{i=1}^{n} x_i$$

(19)

The random variables are taken as lognormally distributed, with unit means and standard deviation $\sigma = 0.2$. Results are obtained herein for a fixed number of variables, $n = 40$, and shown in Table 4. The failure probability found in the literature (Echard, Gayton and Lemaire, 2011) by MCS with $3 \times 10^5$ samples is $1.813 \times 10^{-3}$.
Table 4. Reliability results for example 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\hat{P}_f$</th>
<th>$n_{CLS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>1.8267×10^{-3}</td>
<td>3×10^3</td>
</tr>
<tr>
<td>ANN</td>
<td>1.8297×10^{-3}</td>
<td>314</td>
</tr>
</tbody>
</table>

Convergence of the failure probability is illustrated in Fig. 4 and occurred after 89 iterations. At the end of the process, all ANNs presented $n_2 = 21$, belonging to the first group of surrogates.

![Figure 4. Convergence of the failure probability for the third example: $\hat{P}_f$ (solid line), $\hat{P}_f^{-/+}$ (dotted line) and reference failure probability (horizontal dashed line).](image)

In this example, a higher number of iterations is required to achieve convergence. However, Fig. 4 indicates that a larger tolerance, $\varepsilon_{\hat{P}_f}$, could be applied on the stopping criterion. The estimated failure probability is close to the reference value for example when $n_{CLS} = 248$, which corresponds to the 67th iteration; in this case, $\hat{P}_f = 1.8017 \times 10^{-3}$.

It is also noted that the fewest number of neurons allowed in the hidden layer was imposed to all ANNs. This is related to the small number of points available for training the networks, but also reflects the fact that the approximation of the limit state equation at hand, $g(x_1, x_2, ..., x_n) = 0$, does not require a very complex ANN, for example, an ANN with $n_2 = 2 \cdot n + 1 = 81$ neurons in the hidden layer. Maybe, even a network with $n_2 < 21$ would be enough to solve this example.

5 CONCLUSIONS

In this paper, an adaptive procedure for reliability analysis using surrogate models, previously proposed in the literature in the context of Kriging and Polynomial Chaos Expansions, was adapted to the case of Artificial Neural Networks. Details about number of layers, number of neurons in each layer, initialization of the neural networks, initialization and enrichment of the experimental design, as well as other aspects, were presented and discussed.

The proposed methodology was employed in the solution of three benchmark reliability problems, with different characteristics and number of random variables, and compared to the simple Monte Carlo simulation method. As expected, in all cases the ANNs led to results very close to those obtained by MCS, and required much less evaluations of the limit state function, about 100 or 300 for the cases studied. These results confirmed the applicability of the ANNs, as well as of the proposed methodology, to the reliability analysis of problems involving small failure probabilities.

The results also allowed to identify aspects of the methodology which could be further improved in future studies. For example, a better strategy to increase or decrease the number of neurons in the hidden layer should be proposed, according to the complexity of the function to be approximated. This strategy should also consider the number of points available for training of the ANN. Another aspect which should be improved is the replacement strategy adopted during the learning process, which could, for example, consider a minimum acceptable performance to decide if a given ANN must be replaced or not.

Further investigations on the number of points of the initial experimental design, number of points to be added in each iteration, and many other parameters employed are also required.
ACKNOWLEDGEMENTS

The author acknowledges sponsorship of this research project by the National Counsel of Technological and Scientific Development (CNPq) via grant #442183/2014-3.

REFERENCES


McCulloch, W., Pitts, W., 1943, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biology, 5:115–33.


RESPONSIBILITY NOTICE

The author is the only responsible for the printed material included in this paper.