ELSEVIER

Comput. Methods Appl. Mech. Engrg. 191 (2001) 113–132

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Neural-network-based reliability analysis: a comparative study

Jorge E. Hurtado [*], Diego A. Alvarez

Universidad Nacional de Colombia, Apartado 127, Manizales, Colombia

Received 17 November 2000; received in revised form 16 February 2001

Abstract

A study on the applicability of different kinds of neural networks for the probabilistic analysis of structures, when the sources of randomness can be modeled as random variables, is summarized. The networks are employed as numerical devices for substituting the finite element code needed by Monte Carlo simulation. The comparison comprehends two network types (multi-layer perceptrons and radial basis functions classifiers), cost functions (sum of square errors and cross-entropy), optimization algorithms (back-propagation, Gauss–Newton, Newton–Raphson), sampling methods for generating the training population (using uniform and actual distributions of the variables) and purposes of neural network use (as functional approximators and data classifiers). The comparative study is performed over four examples, corresponding to different types of the limit state function and structural behaviors. The analysis indicates some recommended ways of employing neural networks in this field. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Reliability analysis; Monte Carlo simulation; Neural networks

1. Introduction

The increasing demand for the assessment of reliability calculations of mechanical components and entire structures under the influence of several random parameters has fostered the research of numerical procedures for facilitating the estimation of their usually low failure probabilities. Symbolically, the reliability problem can be stated as

$$P_{\mathrm{f}} = \int_{\mathscr{F}} f_{\boldsymbol{X}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \tag{1}$$

where P_{f} is the probability of exceeding a critical threshold of an important variable, $f_{\boldsymbol{X}}(\boldsymbol{x})$ is the multivariate density function of the vector $\boldsymbol{X}$ of random parameters and $\mathscr{F}$ denotes the mapped failure domain in the $\boldsymbol{x}$-space. This latter can be expressed as

$$\mathscr{F} = \{\boldsymbol{x} | g(\boldsymbol{x}) \leqslant 0\}, \tag{2}$$

where $g(\boldsymbol{x})$ is the so-called limit state function. In practice, complex failure domains are usually common due to the several constraints to which a structural design is submitted. Such complex domains are constituted by unions and intersections of simple domains defined by different limit state functions $g_j(\boldsymbol{x})$, $j = 1, 2, \ldots, J$.

The available methods for estimating the probability of failure P_{f} can be roughly classified into two groups, which can be labeled as *gradient-based* and *simulation-based* methods. In the first case there is a need of estimating the gradient of the limit state function in a relevant point about which the largest concentration of the probability mass in the failure region can be found. Several variants of the so-called FORM and

[*] Corresponding author. Tel.: +57-6-886-3182; fax: +57-6-886-3220.
E-mail address: jhurtado@emtelsa.multi.net.co (J.E. Hurtado).

SORM approaches have been proposed in the literature to this purpose. A crucial point in their application is the need of knowing the limit state function explicitly. Since in conventional finite element calculation it is only known implicitly (e.g., as $g(\boldsymbol{x}) = \bar{r} - r(\boldsymbol{x})$, where $\bar{r}$ is a the threshold value for a response $r(\boldsymbol{x})$), the use of the statistical technique known as response surface method (RSM) has been proposed to approximate the function about the design point by a first- or second-order polynomial in the basic variables, which allows the estimation of the gradient. On the other hand, simulation-based methods hinge upon the creation of a synthetic set of S response samples on which the probability of failure can be estimated as

$$\hat{P}_{\mathrm{f}} = \frac{S_{\mathscr{F}}}{S}, \tag{3}$$

where $S_{\mathscr{F}}$ is the number of samples lying in the failure domain. The simplest technique, which is the so-called basic Monte Carlo method (MCM), consists in generating random numbers from the probability distribution of the basic variables and solving the structural problems resulting from their random combination. For an ample description, the reader is referred to [17]. Since the cost of this method is very high, inasmuch as it requires a very large number of computations (roughly about $100/P_{\mathrm{f}}$), special methods, such as conditional simulation [2], importance sampling (IS) [1,5] and directional simulation (DS) [8], have been proposed to reduce such computational effort. Besides, simulation techniques have also been combined with the response surface approach in order to further reduce the number of simulations, in such a way that the response surface acts as a surrogate to the finite element solver for obtaining most of the samples [9,10,16]. In that case the only finite element solver calls are those employed in computing the response surface. Further discussions can be found in [14,23].

The artificial neural network (ANN) approach is in this respect similar to the just-quoted combination. In this case, the neural network is trained with some actual simulated values and then used in substitution of the model code. A clear advantage of this approach over the MCM–RSM combination is the possibility of allowing mappings of the basic variables which are more involved than the polynomials usually employed in the RSM approach, due to the intensive interconnection of the composing units that facilitates the treatment of complex failure domains, which are commonly composed by irregular limit state functions [13].

There are some reports on research using ANN. First to be mentioned must be the work by Chapman and Crossland [6], who employed ANN to risk plant assessment, and a paper by Saraiva and Ebecken [22], who used the classical multi-layer perceptron to obtain good approximations of the failure probability of some structural models. Worth mentioning is above all the paper by Papadrakakis et al. [18], who applied the MCM–ANN combination to the reliability analysis of elasto-plastic structures and compared the performance of several architectures trained with the classical back-propagation (BPX) algorithm. Since several alternatives of ANN design have been proposed in the literature, regarding learning philosophy, training algorithms and cost function, as well as several ways of employing neural networks in this field, further comparative studies are in order, especially for the common case of failure probabilities lower than those of the examples considered in [18], which are greater than 0.01. Such is the task undertaken in this paper in order to approach a standard procedure for employing the ANN technique for reliability analysis. Since the purpose of the paper is to judge the applicability of ANN in reliability analysis from the roots, the study was made with the basic Monte Carlo method in order to avoid bias in the interpretation of the results, which could eventually be introduced by a combination with advanced Monte Carlo techniques. However, some comparative remarks about the combinations with these latter are made before the conclusions.

Within this framework, comparisons are made in the paper with respect to the following issues:

1. *Network types and learning algorithms.* The types tested were multi-layer perceptrons (MLPs) and radial basis function (RBF) networks. In the first case, several alternatives of finding the optimal network parameters were tested, which are: standard back-propagation with variable learning rate and momentum (BPX), Gauss–Newton–Levenberg–Marquardt (GNLM) and Newton–Raphson–Levenberg–Marquardt (NRLM). In the second case, the exact algebraic (EA) and the least-squares (LS) approaches are examined.
2. *Cost functions.* Sum of square errors (SSE) and cross-entropy (CE).
3. *Purpose of ANN use.* For functional approximation (FA) and data classification (DC).
4. *Sampling procedures.* Random number generation from the actual distribution functions of the basic variables (CDF) and from uniform distributions of the basic variables with k standard deviations about the mean (UNk).

The paper is divided as follows: initially, a brief description of the above-listed algorithms, functions and criteria is presented. Then some suggested rules for employing ANN in reliability analysis are exposed. On the basis of these the numerical study performed on four selected examples is summarized, focusing on some relevant comparisons. Finally, some conclusions are drawn that are intended to serve as a guide for practical ANN use in reliability analysis in connection with the Monte Carlo method.

2. Comparison items

This section is devoted to a description of the most important aspects of the design features, learning procedures and ways of use of the ANN employed in the present study.

2.1. Network types and learning algorithms

Two kinds of supervised learning designs were tested in the present study, namely, the MLP, in which the errors existing between the actual and the estimated output are used for updating the network parameters, and the RBF networks, in which the input vectors are classified according to their clustering about a common output [7].

2.1.1. Multi-layer perceptrons

Fig. 1 illustrates the MLP architecture. It is composed of three or more layers of neurons. In the first one, no operations are performed onto the input data, which are represented by the L neurons depicted in the figure. These values are directly transmitted to the M neurons of the second (or hidden) layer affected linearly by weights w_{lm}. The total activation value of each neuron in this layer is

$$a_m = \sum_{l=1}^{L} w_{lm} x_l. \tag{4}$$

The output of each neuron is a linear or nonlinear function of this activation value:

$$s_m = f(a_m). \tag{5}$$

This transformation is represented in the figure by a unipolar sigmoid function inserted into each neuron, since this is one of the most usually employed in ANN applications. It is given by

$$f(t) = \frac{1}{1 + \exp(-\alpha t)}, \tag{6}$$

where α is a parameter defining the slope of the function. Finally, the information arrives to and is transformed by the neurons of the subsequent layers in the same way. In the last (or output) layer similar transformations are applied. A linear function is depicted in the figure, which is used in this study for FA employment of ANN, while a unipolar sigmoid function was used for DC application, due to its saturation properties. The value estimated by the network at neuron n is denoted as $\hat{y}_n$ to distinguish it from the actual one, y_n.

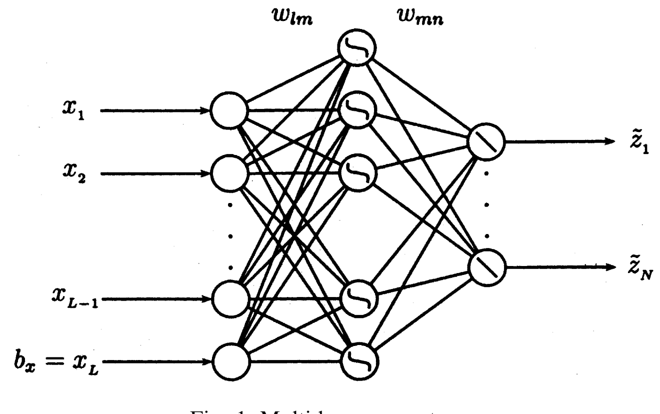

Fig. 1. Multi-layer perceptron.

Several methods can be applied to find the values of the weights leading to a minimum error with respect to the actual output values. A detailed description of these algorithms would occupy a large amount of space, so that only the basic equations of each are presented in the next description. The algorithms are the following:

1. *BPX.* This algorithm is based on the conventional gradient descent technique and is amply described in several books (e.g., [7]). At step $[k+1]$ the vector of the network weights is updated according to the following rule:
$$\boldsymbol{w}[k+1] = \boldsymbol{w}[k] + \Delta\boldsymbol{w}[k], \tag{7}$$
with
$$\Delta\boldsymbol{w}[k] = -\eta\nabla_{\boldsymbol{w}}E[k] + \omega\Delta\boldsymbol{w}[k-1]. \tag{8}$$
In these equations η and ω are factors known as *learning rate* and *momentum* in the soft computing literature, and $\nabla_{\boldsymbol{w}}E$ is the gradient vector of the error (or cost) function E, defined in Section 2.2, with respect to the weights. The learning rate can be set to be adaptive as the training progresses. In this study, two types of error function were tested, which are described in the sequel.
2. *GNLM.* In this method, the weight vector is updated according to the following equation:
$$\boldsymbol{w}[k+1] = \boldsymbol{w}[k] - (\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} + \lambda\boldsymbol{I})^{-1}\nabla_{\boldsymbol{w}}E, \tag{9}$$
where $\boldsymbol{J}$ is the Jacobian matrix of the network outputs with respect to the weights, $\boldsymbol{I}$ is the identity matrix and λ is a suitable nonnegative value.
3. *NRLM.* In this case, the weight vector changes with the rule
$$\boldsymbol{w}[k+1] = \boldsymbol{w}[k] - \eta(\boldsymbol{H} + \lambda\boldsymbol{I})^{-1}\nabla_{\boldsymbol{w}}E, \tag{10}$$
where $\boldsymbol{H}$ is the Hessian matrix of the ANN outputs with respect to the weights.

A theorem by Hornik et al. [12] is often quoted as the basis of the MLP methodology for functional approximation. The theorem states that MLP with as few as one hidden layer and arbitrary so-called squashing activation functions and linear or polynomial integration functions can virtually approximate any function provided many hidden units are included in the perceptron. Most activation functions used in MLP applications fall under the class of squashing functions.

2.1.2. Radial basis function networks

The other network type considered in the study was the RBF, which is widely used in the field of pattern recognition [20] for data classification and clustering. This goal resembles that of the reliability analysis in that the limit state functions define two well-distinguished regions, namely, the safe and failure ones. Accordingly, RBF networks are good candidates for estimating the probability of failure of mechanical systems.

Fig. 2 illustrates the RBF network. It differs from the MLP in that it is intended to learn the input–output mapping not by building a complex nonlinear relationship but by classifying the data into several groups according to their proximity to a common center and weighting the participation of each cluster into the output. There are several proposals for the architecture and training algorithms of this network type. In this study, use was made of the following design. The first layer corresponds to the input values x_l, $l = 1, 2, \ldots, L$, to which a bias neuron is added. The input values are directly transmitted to the hidden layer without being affected by weights. The activation functions of each neuron m of this layer are intended to be receptive fields of the input information, which classify the data according to their proximity to a center c_m in the form
$$s_m = f(\|\boldsymbol{x} - \boldsymbol{c}_m\|). \tag{11}$$
In the design used herein, a Gaussian function
$$f(t) = \exp\left(\frac{t^2}{2\omega^2}\right) \tag{12}$$
was employed. The constant ω defines the width of the receptive field. Since it is assigned beforehand it can be used as a label to identify the network. The output of the neuron n of the end layer is computed in a similar way as in the MLP network:

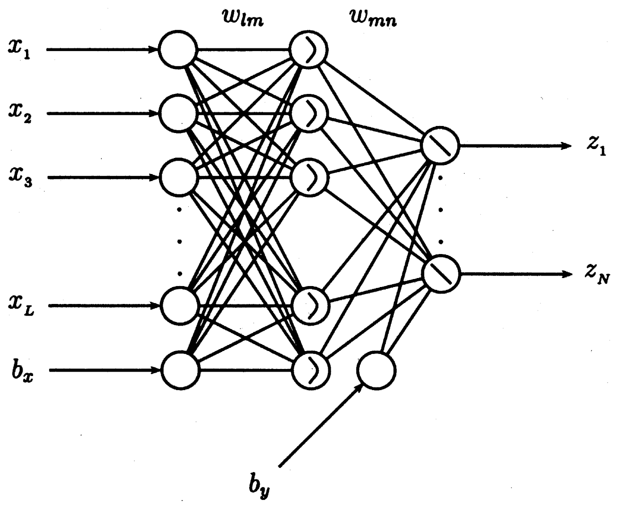

Fig. 2. RBF network.

$$\hat{y}_n = p\left(\sum_{m=1}^{L} w_{mn} s_m\right), \tag{13}$$

where $p(\cdot)$ is the activation function of the layer, which is usually the purely linear one, and w_{nm} is the corresponding weight. In such a case the following algebraic problem can be set to find out the unknown weights:

$$\boldsymbol{S}\boldsymbol{w} = \boldsymbol{y}, \tag{14}$$

where $\boldsymbol{S}$ is the matrix of the radial basis outputs given by Eq. (11), $\boldsymbol{w}$ the vector of unknown weights and $\boldsymbol{y}$ the actual outputs. This problem can be formulated basically in two ways, namely, as an EA problem or as an LS one. In the first case, the size of matrix $\boldsymbol{S}$ is $P \times P$, where P is the number of training patterns. This means that there must be also P receptive field centers, which are usually the same training points. Thus the RBFs will be activated according to the relative Euclidean distance of the data points to each other. This approach normally implies a large number of neurons in the hidden layer. In the second case, the size of the matrix is $P \times M$, where $M \ll P$ is the number of hidden neurons (see Eq. (13)). A strategy usually applied in this situation is to increase the size of the hidden layer by adding a neuron and solving the LS problem

$$\boldsymbol{w} = (\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S})^{-1}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{y} \tag{15}$$

until the square error reaches a minimum. For practical applications, it is important to balance the relative advantages of these two approaches: the first implies a solution of a large system of equations but has no compromises with respect to the accuracy in terms of the training set; the second leads to a minimal network that can be very fast in the production phase, but at the cost of some loss of accuracy. The training time of the EA approach is normally minimum, while that of the LS one depends on the problem in hand.

2.2. Cost functions

An important issue in ANN application in this field is the selection of the error function. In most applications use is made of the SSE, given by

$$E = \sum_{p=1}^{P} \sum_{n=1}^{N} (y_n(p) - \hat{y}_n(p))^2, \tag{16}$$

where $y_n(p)$ is the actual output associated with neuron n at the end layer, $\hat{y}_n(p)$ its ANN estimate, N the number of neurons in the output layer and P the number of training patterns.

In the reliability context, another possibility of the cost function can be rooted in entropy concepts. In fact, since the reliability problem can be associated to the assigning of data to two classes (namely, failure and safe ones) as stated before, the CE function, usually employed in pattern recognition, is worth testing [24]. It is given by

$$E = \sum_{p=1}^{P} \sum_{n=1}^{N} y_n(p) \ln \hat{y}_n(p) + (1 - y_n(p)) \ln(1 - \hat{y}_n(p)). \tag{17}$$

In this case $y_n(p)$ denotes the probability that the associated input belongs to the failure or safe region, taking values 1 and 0, respectively, and $\hat{y}_n(p)$ is the ANN estimate, obtained with a unipolar sigmoid function in the output layer neurons, which can be regarded as a posterior probability. Once the network has been trained, an estimation $\hat{y}_n(p) > 0.5$ can be interpreted as a point belonging to the failure zone and to the safe one otherwise.

2.3. Purpose of ANN use

Artificial neural networks can be used in mechanical analysis in general to approximate complex functions. For instance, in reliability analysis they can be trained to approximate the value of the limit state function $g_j(\boldsymbol{x})$. This approach is labeled as functional approximation in this study. However, taking into account the similarity to the classification problem, another possibility is just to use ANNs as classifying devices, so that the training values are simply ones and zeroes, depending on the location of the data point in the failure or safe regions, respectively. This approach is herein referred to as data classification.

2.4. Sampling procedures

A final issue of concern is the various ways that can be followed to generate the training population. Standard Monte Carlo procedures require the use of random number generation from the distributions of the variables (DF) entering into a given problem. However, for training purposes one can also generate them from uniform distributions with k standard deviations around the mean value of each basic variable (UNk) with the purpose of having training points in the far regions. Also, if the limit state function is known explicitly, one can even train the network by assigning the training points close to it in a deterministic fashion as shown next.

3. Criteria for ANN application

Before considering in detail the relative efficiency of the above-listed methods and algorithms, it is necessary to state some criteria that were adopted for the employment of neural networks for reliability analysis and judging their accuracy. It is necessary to take into account that, since the weights of the connections in MLP and the centers of RBF networks are initiated at random, and that the network architecture as such can be varied, the final result of the optimization problem can be somewhat different in each case, thus leading to different estimates of the limit state function or of the region membership. In passing, it must be said that this is a typical problem found in all Monte Carlo methods and it is by no means exclusive of the ANN approach.

As a consequence, a criterion must be established to select the estimate of the probability of failure amongst the manifold of results given by the networks and therefore of the error of the ANN–MCM method as such. (In this regard, it must be said that for most training algorithms the computation time is short, so that training several networks does not require a serious computational effort.)

A naive approach, usually applied when using standard methods such as importance sampling, is to take as the final estimate the average of several estimations and to take the standard deviation as a measure of the error of the method and of the confidence interval of the final estimate. However, in contrast to basic and advanced Monte Carlo methods, the ANN technique allows using a sophisticated approach, which is detailed next, to arrive at the final estimate without the costly resampling required for averaging different IS or DS estimates. Also, since the neural networks are initiated at random, this method involves some kind of randomness that cannot be reached by the RSM technique, which on the basis of a single training population gives a unique mapping.

The proposed approach for obtaining the estimate of the probability of failure and its confidence interval is based on the treatment of the limit state function as a classification rule. According to the pattern recognition view, the mode of several estimates makes more sense as an estimate of a probability than their mean, inasmuch as the mode represents a consensus of different networks about the frequency of repetition of a phenomenon. As is well known, the mode of a population is not an easy parameter to draw. The task, however, is alleviated in this case by the binary nature of the classification problem. Suppose we have an odd number Q of networks of the same type, perhaps with different architectures and activation function parameters, all of them displaying a little training error. The problem is then to arrive to an end estimate of the probability of failure $\hat{P}_{\mathrm{f}}$ by consensus of the network estimates. It would be possible to calculate this mode as that of the values

$$\hat{P}_{\mathrm{f},q} = \frac{S_{\mathscr{F},q}}{S}, \tag{18}$$

where $q = 1, 2, \ldots, Q$, $S_{\mathscr{F},q}$ is the number of samples falling in the failure region for the current ANN and S is the total number of Monte Carlo samples, which is assumed to be equal for all the Q networks. However, unless some lucky coincidences occur in the network population, it would be difficult or impossible to establish its mode. The procedure devised in the present study to this purpose is construed on a sample-by-sample basis. In other words, the estimate of the probability of failure can be obtained taking the mode of the indicator function values $I_q(\hat{g}(\boldsymbol{x}_s) \leqslant 0)$ for each sample $s = 1, 2, \ldots, S$ amongst the Q neural networks. These modes are equal to 0 or 1, depending on the number of coincidences of the Q ANNs on each sample. The estimate is, hence,

$$\hat{P}_{\mathrm{f}}^{*} = \frac{S_{\mathscr{F}}^{*}}{S}, \tag{19}$$

where $S_{\mathscr{F}}^{*}$ is the number of ones in this computation. This procedure is illustrated by Fig. 3. Notice that the selection of an odd number of ANNs is required by the need of having a mode for each sample.

The joint probability of failure $\hat{P}_{\mathrm{f}}^{*}$ can be considered more robust than any of the estimates $\hat{P}_{\mathrm{f},q}$ and the mode thereof, as it involves a consensus about each sample. Hence it can be defined as the central value of a normal distribution with variance

$$\sigma^2_{\hat{P}_{\mathrm{f}}^{*}} = \hat{P}_{\mathrm{f}}^{*}(1 - \hat{P}_{\mathrm{f}}^{*}) \tag{20}$$

because $\hat{P}_{\mathrm{f}}^{*}$ can be regarded as a Bernoulli trial. If most of the Q estimates $\hat{P}_{\mathrm{f},q}$ are contained in a suitable confidence interval for the estimate $\hat{P}_{\mathrm{f}}^{*}$ thus specified, then the entire set of ANNs can be accepted and $\hat{P}_{\mathrm{f}}^{*}$ is the end estimate. Otherwise, a decision should be taken on whether the ANN design is to be modified or the training population size should be augmented. This decision can be assisted by observing the fraction of the neural networks defining the mode of $\hat{P}_{\mathrm{f},q}$. This fraction is denoted as ρ in Fig. 3. If ρ is greater than, say, 0.5, and if the mode of $\hat{P}_{\mathrm{f},q}$ is close or equal to $\hat{P}_{\mathrm{f}}^{*}$, a new network design and training is scarcely justified. For instance, in the illustration of Fig. 3, $\rho = 3/5$ and the mode of $\hat{P}_{\mathrm{f},q}$ equals 3/6, which coincides with $\hat{P}_{\mathrm{f}}^{*}$. Hence this value can taken as the end estimate with the confidence interval given by Eq. (20).

$s =$	1	2	3	4	5	6	$\hat{P}_{\mathrm{f},q}$
$q = 1$	1	1	0	1	0	0	3/6
2	1	1	1	1	0	0	4/6
3	0	0	1	1	0	0	2/6
4	1	0	1	0	1	0	3/6
5	1	0	1	1	0	0	3/6
Mode	1	0	1	1	0	0	

$\therefore \hat{P}_{\mathrm{f}}^{*} = 3/6$

Fig. 3. Computation of the failure probability using several networks.

4. Numerical study

The issues listed in Section 2 imply that a wide numerical study should be performed if all comparisons were to be made simultaneously. In this study, some selected tests were made at a time in order to get insight into the different possibilities of ANN use in the reliability context and the best alternatives in each comparison were used in the next analyses. The comparisons were made over the following cases:

1. *A single and explicit nonlinear limit state function.* It is

$$g(\boldsymbol{X}) = -0.3X_1^2 - X_2 + 1.5, \tag{21}$$

where $\boldsymbol{X}$ obeys a bivariate normal distribution with zero mean and a covariance matrix equal to $0.25\boldsymbol{I}$. The reference probability of failure, evaluated with 100,000 Monte Carlo simulations, is 0.00325. The networks were trained with a short set of data points, shown in Fig. 4, which are very close to the limit state function on both sides.

2. *A multiple, linear and explicit limit state function.* This corresponds to limit situations of collapse of the frame depicted in Fig. 5. The moment capacity of the hinges as well as the external loads are random variables whose probability information are shown in Table 1. The limit state is defined as the collapse

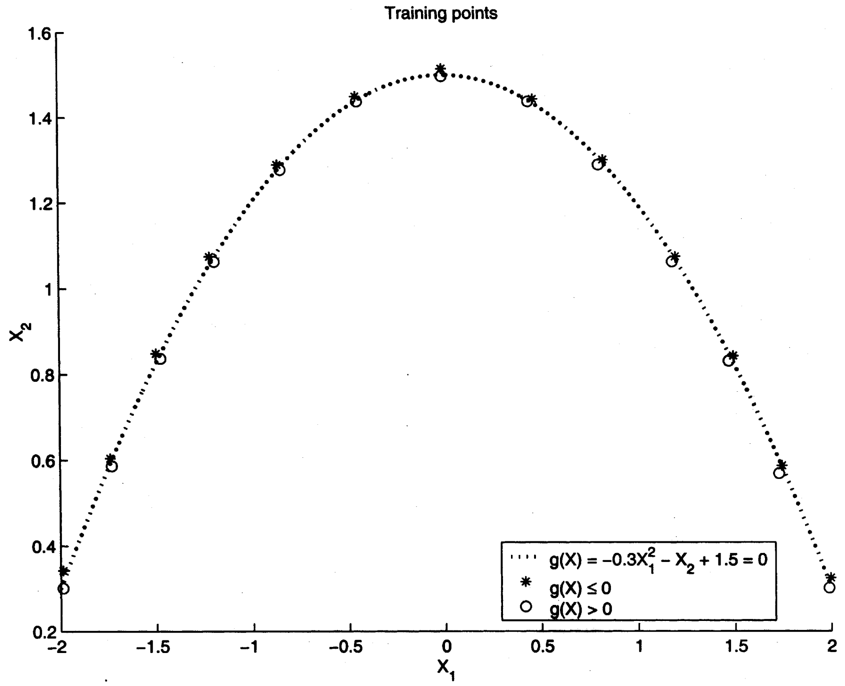

Fig. 4. Parabolic limit state function and training points.

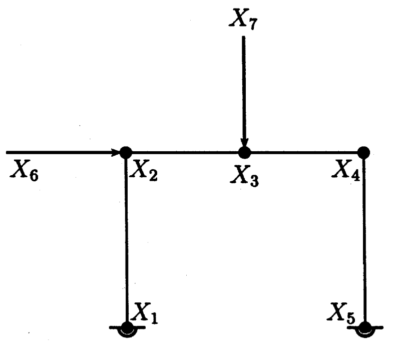

Fig. 5. Plastic frame.

Table 1
Case No. 2: Probability distributions of the basic variables

Variable	Distribution	Mean	Coefficient of variation
X_1, X_2, X_3, X_4, X_5	Lognormal	134.9 kN m	0.1
X_6	Normal	50 kN	0.1
X_7	Gumbel	40 kN	0.25

of the frame, i.e. the exceeding of the capacity of the plastic hinges by the load moments in three different failure modes. Accordingly, the multiple limit state function is the following:

$$\begin{aligned} g_1(\boldsymbol{X}) &= X_1 + X_2 + X_4 + X_5 - hX_6, \\ g_2(\boldsymbol{X}) &= X_1 + 2X_2 + 2X_4 + X_5 - hX_6 - hX_7, \\ g_3(\boldsymbol{X}) &= X_1 + 2X_3 + X_4 - hX_7, \end{aligned} \tag{22}$$

where h is the frame height, equal to 3 m. The probability of failure was estimated by direct Monte Carlo with 10^6 samples as 0.000184.

3. *An implicit, nonlinear limit state function for an elastic frame.* This corresponds to the exceeding of a limit value (11 cm) by the roof displacement of the linear frame depicted in Fig. 6. The basic random variables are as described in Table 2. The exact probability of failure is 0.00545, computed over 350,000 samples.
4. *An implicit, nonlinear limit state function for a plastic plane structure under dynamic load.* Fig. 7(a) shows a plastic frame discretized with 20 plane stress finite elements subject to a suddenly applied random horizontal load $P(t) = \varpi p(t)$, where ϖ is a normal variable with mean 50 N and standard deviation 15 N and $p(t)$ is the function shown in Fig. 7(b). The modulus of elasticity is lognormal, with mean 200,000 N/m^2 and standard deviation 20,000 N/m^2. The uniaxial yielding stress is 50 N/m^2. No hardening is assumed to be present in the material, whose plastification is supposed to follow the Von Mises criterion. The total number of degrees of freedom is 170. Failure is defined as the surpassing of a limit displacement of 0.00085 by the horizontal degree of freedom of the top right joint. Fig. 8 shows a typical history of this degree of freedom. The reference probability of failure, calculated with 21,000 Monte Carlo samples, is 0.00743. Notice that the plastic and dynamical characteristics of the problem makes more uncertain the end value of the displacement.

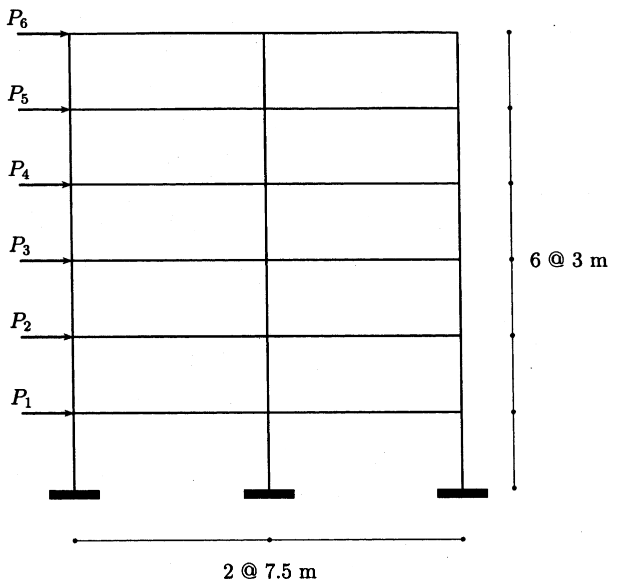

Fig. 6. Linear frame.

Table 2
Case No. 3: Probability distributions of the basic variables

Variable	Distribution	Mean	Standard deviation
E of columns	Lognormal	1,960 kN/cm^2	196 kN/cm^2
E of beams	Lognormal	1,960 kN/cm^2	196 kN/cm^2
I of columns	Lognormal	100,000 cm^4	10,000 cm^4
I of beams	Lognormal	150,000 cm^4	15,000 cm^4
P_1	Normal	24.50 kN	6.125 kN
P_2	Normal	27.44 kN	6.860 kN
P_3	Normal	28.42 kN	7.105 kN
P_4	Normal	29.40 kN	7.350 kN
P_5	Normal	30.38 kN	7.595 kN
P_6	Normal	31.36 kN	7.840 kN

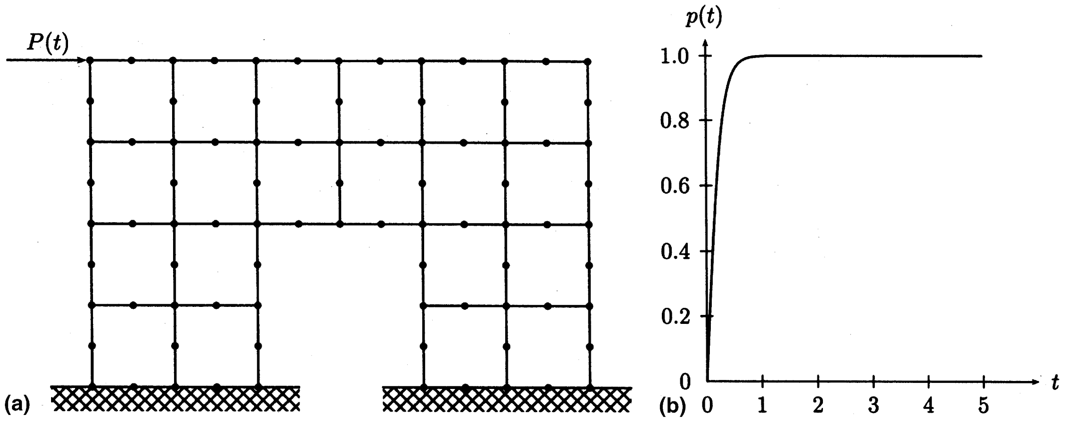

Fig. 7. Plastic plane structure under dynamic load: (a) finite element discretization; (b) load history.

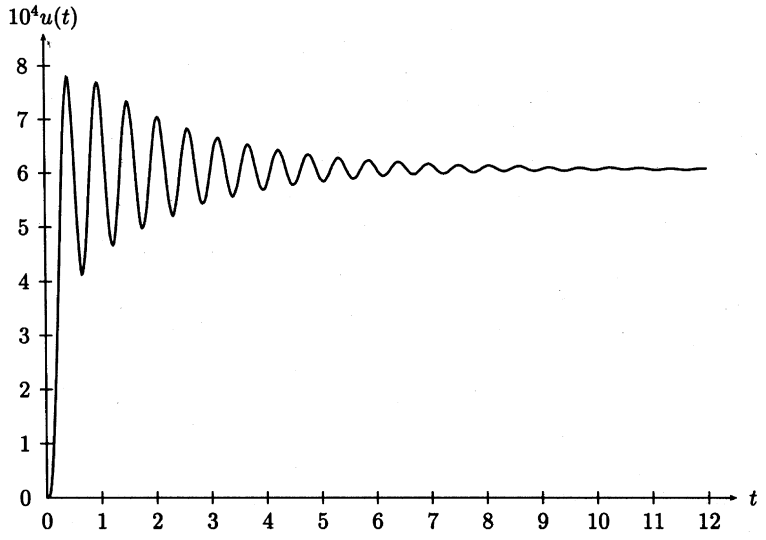

Fig. 8. Typical displacement response of the plane structure.

The following sections summarize the most relevant conclusions drawn from the ample numerical study performed over these examples, which implied more than 5000 ANN trainings. The comparisons parallel the main issues listed in Section 2 of this paper.

4.1. Purpose of ANN use

A first comparison concerns the use of neural networks for functional approximation and data classification. Table 3 displays the results of the failure probabilities calculated with an MLP trained using the algorithm GNLM with the SSE error function for Case No. 1. The relative error in the failure probability in this and the next tables is defined as

$$\epsilon_P = \frac{|\hat{P}_{\mathrm{f}}^* - P_{\mathrm{f}}|}{P_{\mathrm{f}}}. \tag{23}$$

Five networks were used for this calculation. The probability of failure was estimated by the procedure described in Section 3. It can be observed that the use of ANN for data classification is more prone to error than the FA option. This fact was observed also in other calculations not displayed herein.

4.2. Cost function

Table 4 compares the above SSE–GNLM–FA result with two alternatives of training the networks using the CE error function (Eq. (17)). It can be seen that the networks based on the latter are not as accurate as those using the SSE error function. Hence, after the results shown in Tables 3 and 4, it can be concluded that the use of ANN for functional approximation seems to be better in these two respects than the approach inspired by pattern recognition tasks. However, this is anyhow worth of more research especially for complex, implicit, nonlinear limit state functions. In fact, the DC approach can give a close picture of the limit state function that can be useful for many purposes. This is illustrated by Figs. 9 and 10, which show the response surfaces obtained by neural networks of the FA and DC approaches. While the first is similar to the surface that could be obtained with the classical RSM technique, the second is an interesting result that seems difficult to arrive at with other methods than ANN. Moreover, the training time using the DC approach is roughly one-half that required by conventional FA.

4.3. Network types and sampling procedures

In this section and the next, several comparisons concerning network types and designs and training algorithms are considered together.

Tables 5 and 6 show the characteristics of the ANN tested for analyzing Cases No. 2 and 3. Table 5 indicates that neural networks having from 2 to 10 hidden neurons were tested the number of times indicated by the third column. Since each neural network was trained with three sets (of 50, 150 and 600 for Cases No. 2 and 3) and use was made of the CDF and two UN sampling procedures, we have that the total number of neural networks trained for the SSE–BPX and SSE–GNLM types is that given by the last column (in fact, $9 \times 20 \times 3 \times 3 = 1620$ for Case No. 2). The columns of Table 6 are explained similarly.

Table 3
Case No. 1: Comparison of FA and DC alternatives

Algorithm	$\hat{P}_{\mathrm{f}}^*$	ϵ_P
SSE-GNLM-FA	0.00325	0.00%
SSE-GNLM-DC	0.00319	1.85%

Table 4
Case No. 1: Comparison of SSE and CE error functions

Algorithm	$\hat{P}_{\mathrm{f}}^*$	ϵ_P
SSE-GNLM-FA	0.00325	0.00%
CE-NRLM	0.00320	1.54%
CE-BPX	0.00319	1.85%

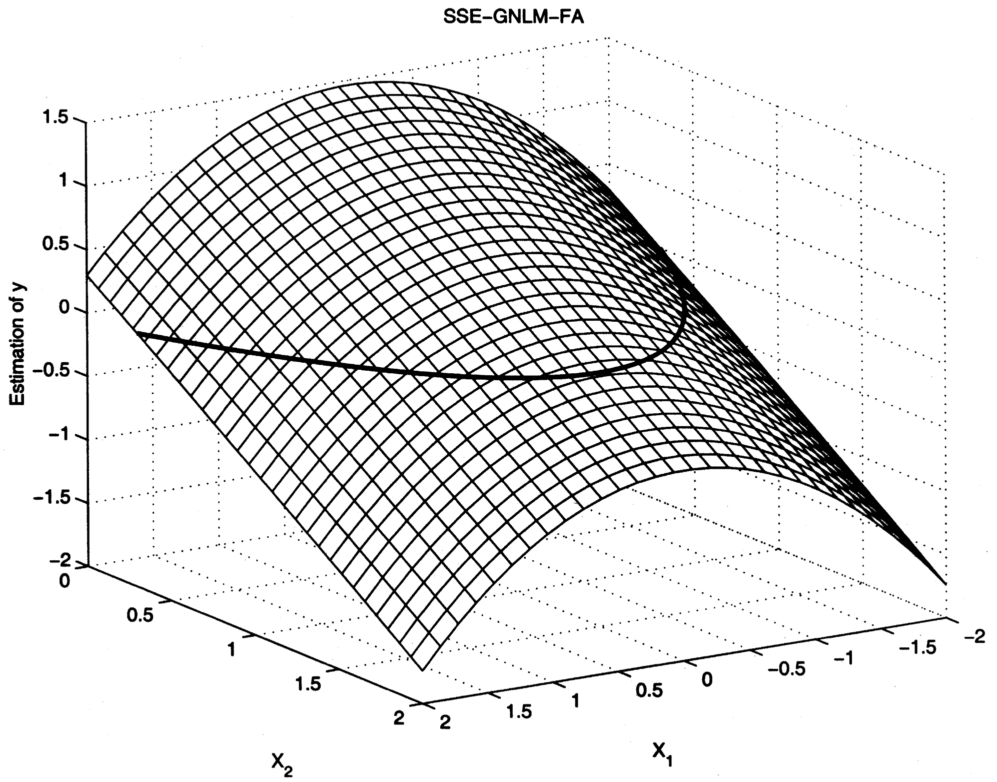

Fig. 9. Functional approximation surface generated by the MLP.

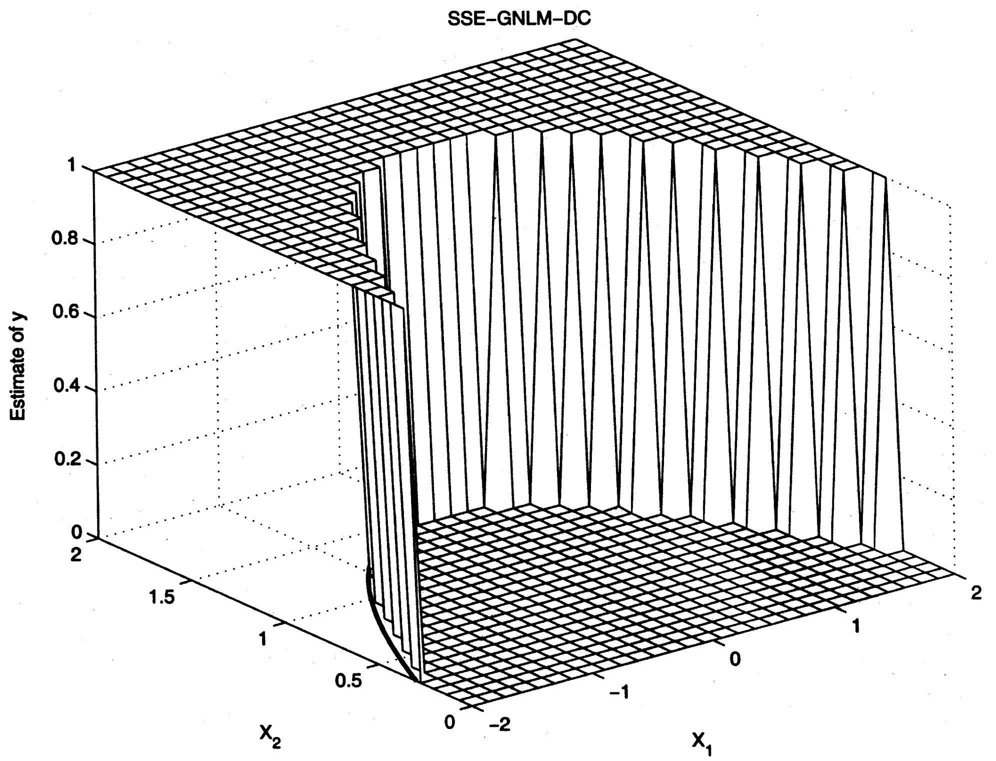

Fig. 10. Data classification surface generated by the MLP.

Table 5
Cases No. 2 and 3: MLP network characteristics

Case	No. of hidden neurons	No. of training trials	No. of SSE-BPX and SSE-GNLM
2	2 to 10	20	1620
3	2 to 8	3	189

Table 6
Cases No. 2 and 3: RBF network characteristics

Case	No. of hidden neurons	No. of training trials	Total No. of ANN
2	14	20	180
3	100	13	117

Table 7
Case No. 2: Performance of three network types

Algorithm	Sample size	Sample type	$\hat{P}_f^*$
SSE-BPX	50	CDF	0
		UN3	0
		UN5	0
	150	CDF	0
		UN3	0
		UN5	0
	600	CDF	0
		UN3	0
		UN5	0
SSE-GNLM	50	CDF	0
		UN3	0
		UN5	0
	150	CDF	0
		UN3	0
		UN5	0
	600	CDF	0.000184
		UN3	0.000177
		UN5	0
RBF-LS	50	CDF	0.000184
		UN3	0.000184
		UN5	0.000184
	150	CDF	0.000184
		UN3	0.000184
		UN5	0.000184
	600	CDF	0.000184
		UN3	0.000184
		UN5	0.000184

The ample number of networks used in the two cases, shown by the last columns of the tables, gives good support to the conclusions that are stated below.

The result of the selection and probability of failure computation for both cases is shown in Tables 7 and 8. The latter also shows the quantities τ_t and τ_p, amplified by a factor of 10^3, which are, respectively, the training times employed in the training and production phases, relative to the time required for the direct computation of the failure probability by Monte Carlo simulation. Notice that the LS algorithm of the RBF network employed in these computations needs much more training time than the exact algebraic model, because it augments the hidden layer by adding neurons until a minimum error is obtained. Several

Table 8
Case No. 3: Performance of three network types

Algorithm	Sample size	Sample type	$\hat{P}_f^*$	$10^3\tau_t$	$10^3\tau_p$
SSE-BPX	150	CDF	0.00114	1.5	1.7
		UN3	0.00498	1.0	1.3
		UN5	0.00874	1.1	1.3
	600	CDF	0.00454	2.8	3.1
		UN3	0.00563	2.8	3.1
		UN5	0.00793	2.9	3.1
	1000	CDF	0.0050	4.4	4.6
		UN3	0.00514	4.3	4.7
		UN5	0.00967	4.3	4.6
SSE-GNLM	150	CDF	0.00491	2.8	3.1
		UN3	0.005303	3.0	3.2
		UN5	0.00967	2.6	2.8
	600	CDF	0.00539	8.6	9.0
		UN3	0.00536	8.6	8.9
		UN5	0.00612	7.9	8.2
	1000	CDF	0.00543	14.7	15.0
		UN3	0.00539	15.9	16.2
		UN5	0.00592	12.8	13.1
RBF-LS	150	CDF	0.00501	1.0	15.4
		UN3	0.00524	1.0	15.3
		UN5	0.00548	1.0	15.4
	600	CDF	0.00513	6.0	20.4
		UN3	0.0053	6.1	20.5
		UN5	0.00573	6.0	20.6
	1000	CDF	0.00515	13.7	28.2
		UN3	0.00530	13.6	28.0
		UN5	0.00572	13.6	28.1

observations stem from these two and other tables that are introduced in the following. A discussion of the reasons behind these results is made in the next section.

1. There is no clear difference in training with CDF and with uniform distributions. The only trend that can be concluded is that a very wide uniform distribution (such as UN5 in Table 7 and UN3 in Table 8) produces a low-density training population which affects the learning process. However, for low failure probabilities, sampling with compact uniform distributions for generating the training population is recommended.
2. For all network models, the sum of training and production times shown in Table 8 is much less than the time required by standard Monte Carlo simulation to calculate the failure probability. Notice first that the training time indicated corresponds to the calculation of the mode, which involves several ANN trainings as indicated previously. This supports the abovesaid about the possibility of training several networks for applying the procedure sketched above without serious increase of the total computation time. If this is not desired and a single network were to be used, the relative advantage over standard Monte Carlo would be much higher in this respect.
3. According to the results of Cases No. 2 and 3, the RBF networks are clearly superior than MLPs trained with the previously selected algorithms. The superiority of the RBFs can be concluded not only on the basis of their accuracy but also in the sense of their stability, as measured by the frequency of times the mode of $\hat{P}_{f,q}$ is reached and the closeness of this mode to the overall estimate $\hat{P}_f^*$. It must be said that, for a complex problem like the third one, the RBF–LS requires much more neurons than in a simple case like No. 2, as indicated by Table 6. This increases the computation time. However, when use is made of the RBF–EA algorithm, the training time is dramatically reduced. This is illustrated by Table 9, which compares these two types of RBF algorithms in the framework of Case No. 3 (sample set size: 150; sampling algorithm: UN3). Notice that the factor multiplying the relative time is now 10^5.

Table 9
Case No. 3: Performance of seven RBF-EA

q	ω	$\hat{P}_{\mathrm{f},q}$	$10^5\tau_{\mathrm{t}}$	$10^5\tau_{\mathrm{p}}$
1	1083333	0.005437	0.15	369
2	1166666	0.005437	0.12	376
3	1250000	0.005437	0.11	365
4	1416666	0.005426	0.15	365
5	833333	0.005474	0.11	371
6	916666	0.005474	0.16	366
7	1000000	0.005437	0.11	359

4. The superiority of the SSE–GNLM method for training MLPs over SSE–BPX is confirmed in these two cases. This is due to the fact that the former often leads to less training errors than the latter. This is illustrated by Table 10, which displays the mean and standard deviation of the relative error

$$\epsilon_g = \frac{|\hat{g}(\boldsymbol{x}) - g(\boldsymbol{x})|}{g(\boldsymbol{x})} \tag{24}$$

computed over the training and validation sets for Cases No. 2 and 3. This difference can be attributed to the different paths the methods take in the minimization procedure, which lead to different solutions. In fact, very different final weights were observed for the same connection when applying the two techniques. However, the advantage of the GNLM over the BPX procedure is of little interest in comparison to the significant superiority of RBF networks over MLPs. In fact, as the table also shows, the errors of the RBF algorithm are several orders of magnitude lower than those of the MLPs in both examples.
5. However, the superiority of RBF networks is not confirmed by the results for Case No. 4 summarized in Table 11. The error in the estimate given by the MLP set is less than that of the RBF–EA networks used. This is due to a special condition found in this example that makes it noteworthy about the ANN mapping, as shown next. In passing, it must be said that the ratio of the training and evaluation times of the ANN to the time required by Monte Carlo simulation in this case is about 10^{-5}.

4.4. Discussion

The above observations deserve a further discussion that may be useful for illuminating possible ways of improving the neural-network-based reliability analysis. To this purpose it is important to examine in

Table 10
Case No. 3: Statistics of the training errors

Case	Algorithm	Training set		Validation set	
		μ_ϵ	σ_ϵ	μ_ϵ	σ_ϵ
No. 2	SSE-BPX	19.9	38.4	14.5	75.4
	SSE-GNLM	0.7	0.7	0.4	2.0
	RBF-LS	0.00003	0.0001	0.00002	0.00008
No. 3	SSE-BPX	2.4	2.6	2.3	1.3
	SSE-GNLM	1.0	1.1	1.2	0.7
	RBF-LS	0.2	0.2	0.2	0.2

Table 11
Case No. 4: Comparison of network types

Algorithm	$\hat{P}_{\mathrm{f}}^{*}$	ϵ_P
SSE-GNLM-FA	0.00690	7.05%
RBF-EA	0.00624	16.03%

greater detail the approximation performed by MLP and RBF networks. Similarly to the MLP case, the approximation of functions via radial basis atoms has been theoretically justified and grounded [11]. Moreover, the RBF approach has a link to modern approximation techniques, mostly used in signal processing. In fact, combining Eqs. (11) and (13), the output of the nth neuron of the last layer reads

$$\hat{y}_n = \sum_{m=1}^{L} w_{mn} f(\|\boldsymbol{x} - \boldsymbol{c}_m\|) \tag{25}$$

in the case of a linear activation function in that unit. Fig. 11 illustrates the case when Gaussian basis functions are used for this kind of approximation. Note that while classical regression procedures use globally active basis, such as x, x^2, e^x, etc., the RBF network composes the estimate with a weighted sum of displaced functions with an infinite but locally active support, similarly to wavelets (e.g., [19]) and fuzzy regression approaches [15]. This justifies the multiple combinations of these approaches that can be found aplenty in recent literature on statistical methods, artificial intelligence, system identification and control.

The difference between MLP and RBF mappings lies in this localization aspect. As a matter of fact, the MLP estimate is evidently global in the sense that it covers the entire space of input variables, as illustrated by Fig. 12(a). This is due to the fact that the transformations performed at the neurons are purported to establish a nonlinear global mapping in which all incoming impulses to each of them participate in each output according to their weights. As a consequence, all the training data are taken into account for the calculation of the optimal connection weights. Conversely, the RBF performs several local mappings in which the importance of each data point is given by its proximity to the center of a receptive field, in the manner illustrated by Fig. 12(b). In fact, since each center has an associated output, it is easy to conclude that a training point is relevant to the end value of some connections only if their corresponding centers are close to it.

The localization feature explains the overwhelmingly superior performance of the RBF method over MLPs in Cases No. 2 and 3. In fact, the compromise of MLP to produce a globally acceptable fitting imposes a heavy responsibility over the optimization procedure, especially in the case of multiple limit states as in example No.

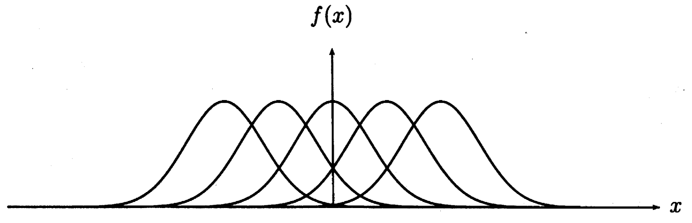

Fig. 11. Localization property of RBFs.

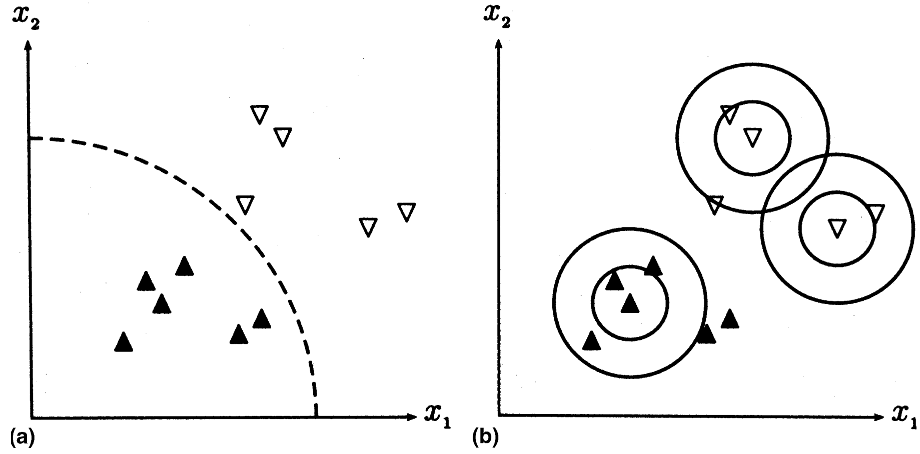

Fig. 12. Types of mappings: (a) global, with MLP networks and RSM; (b) local, with RBF networks.

2, which leads to a solution for the weights that is far from optimal, as Table 10 shows. The remarkable improvement of the performance of MLP in Case No. 3 over that in Case No. 2 is also attributable to the simplification of the mapping occurring in the former, which comprises only a single limit state function.

The localization feature of RBF can be exploited for combinations of neural networks with variance reduction techniques, as described in the next section. However, this very localization characteristic of RBF networks makes them vulnerable to bifurcation problems, which are commonly found in Monte Carlo analysis of some nonlinear systems. This expression is to be understood as the generation of very different output responses corresponding to very similar input data. A situation like this occurs in Case No. 4, as shown by Fig. 13, which shows the Monte Carlo results and a trace of the limit state function. The presence of two different clouds for similar input values disorients the RBF networks as to the correct receptive field for each input vector. On the contrary, the addressing of the dominant cloud towards a small region makes it easy for the MLP to learn the short trace of the limit state function, at variance Case No. 2, whose multiple states are not well captured by this network type. Besides, the small secondary cloud in this bifurcation problem does not affect the MLP learning, due to the global nature of its mapping. These reasons explain the superior performance of MLP over RBF in Case No. 4.

4.5. Combination with variance reduction techniques

Amongst variance reduction methods, IS is one of the most used, not only as an independent technique, but also in combination with other strategies, such as DS [8], and as a basis for building more sophisticated ones intended to cope with multiple limit states [5]. In order to compare this procedure with the ANN approach, it is important to recall that it is based on the use of an ancillary sampling density $h(\boldsymbol{x})$, which together with the given multivariate density $f_X(\boldsymbol{x})$ (see Fig. 14) leads to an estimate of the failure probability of the form

$$\hat{P}_{\mathrm{f}} = \frac{1}{n}\sum_{i=1}^{n} I[g(\boldsymbol{x}_i) \leqslant 0]\frac{f_X(\boldsymbol{x}_i)}{h(\boldsymbol{x}_i)}, \tag{26}$$

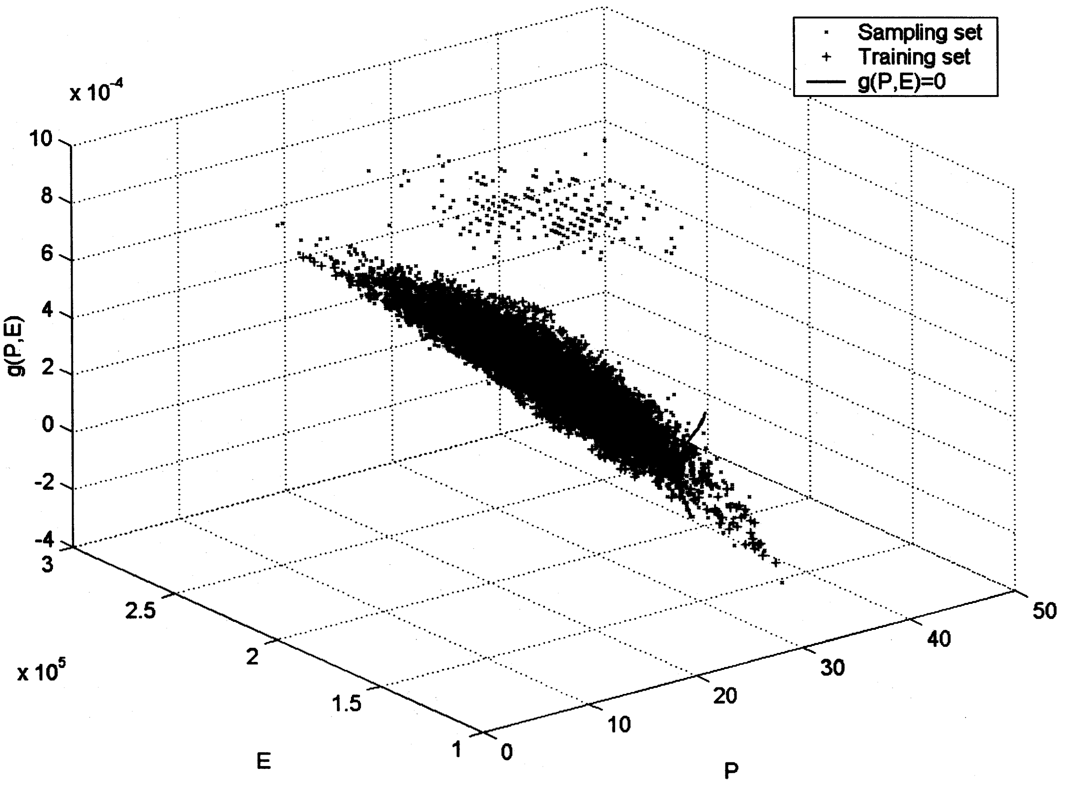

Fig. 13. Monte Carlo points and limit state function of Case No. 4.

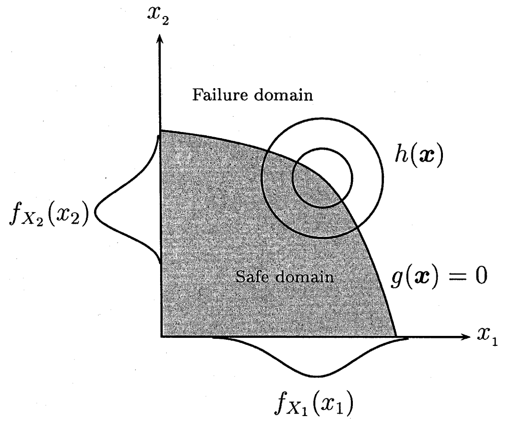

Fig. 14. IS concept.

where the $\boldsymbol{x}_i$ are now generated after the importance sampling density $h(\boldsymbol{x})$ and no longer from the actual one $f_X(\boldsymbol{x})$. This method has its roots in a simple transformation of Eq. (1), whereby the integrand is multiplied by and divided into $h(\boldsymbol{x})$. Notice that this transformation imposes no restriction on $h(\boldsymbol{x})$. This opens the way to an arbitrary assumption of its form and structure. In order to minimize the variance of the estimate (26), it has been demonstrated that the optimal value of the importance sampling density is [17]

$$h^*(\boldsymbol{x}) = \frac{1}{P_\mathrm{f}} I[g(\boldsymbol{x}) \leqslant 0] f(\boldsymbol{x}), \tag{27}$$

which implies that the target of the calculation should be known in advance for its implementation. As a consequence, it is necessary to generate an exploratory population before the final one is simulated. This, of course, leads to an increase of the computational labor required by the method, which is much bigger indeed in the case of multiple limit states and many dimensions of the input variable space, as is common in structural mechanics.

On the contrary, RBF networks are competitive in this respect, as they do not require any exploratory work and fit most naturally to the case of multiple limit states. This is due to the localization of the RBF procedure, which brings to mind that of the IS method (compare Figs. 12(b) and 14). The difference between the IS and RBF methods lies in that, while the former is aimed at sampling with a localized density, whose position and parameters require an exploratory sampling, the RBF technique privileges the points located near each center, which can be located anywhere. This suggests that a good combination for further reducing the number of finite element solver calls is to combine these two methods by establishing a further privilege to those RBF centers having a high probability content in the failure region, as the IS method does. Research in this direction is presently being done by the authors.

Another linkage of neural networks that is worth examining is DS, whose aim is to address the sampling towards the design point [8]. This technique is now applied to generate the training patterns of the neural networks in order to minimize the finite element solver calls in the frame example examined earlier. Table 12 summarizes the results obtained with seven MLP–SSE–GNLMs trained with 150 samples only, which were generated by focusing the sampling towards the design point in the manner illustrated by Fig. 15. This network type was selected in order to examine whether its behavior improves if the domain of its global mapping is reduced.

In other words, the training points are those located in the union of the hypersphere and the hypercone delimited by

$$\cos\gamma = \frac{\beta}{\sqrt{\beta^2 + \kappa^2}} \tag{28}$$

in the standard space. This is defined by the transformation

$$u_i = \frac{x_i - \mu_i}{\sigma_i}, \tag{29}$$

Table 12
Frame example: Estimates with Directional Simulation

No. of hidden neurons	$\hat{P}_{\mathrm{f}}^{*}$
6	0.0053
8	0.0053
6	0.0054
7	0.0053
6	0.0053
5	0.0053
4	0.0054

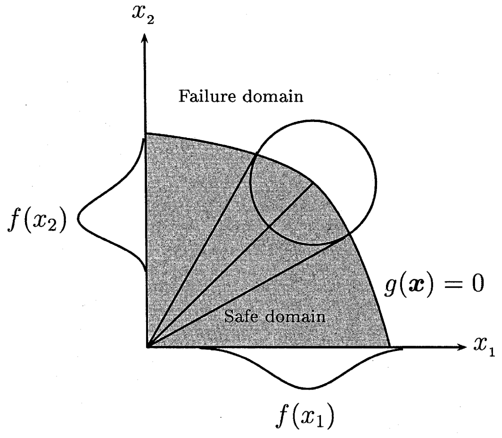

Fig. 15. DS used for Case No. 3.

wherein μ_i and σ_i are the mean and standard deviation of variable x_i. A value of $\kappa = 3$ was selected. As said before, an obstacle for a direct application of these combinations of neural networks and importance or directional sampling methods lies in the need of locating the design point. This can be assisted by several optimization techniques, such as design of experiments [4], evolutionary strategies for constrained optimization [3], and constrained random search methods [21]. This is due to the fact that, in the standard space, the location of the design point corresponds to a minimization problem with an equality constraint [17]. Since this calculation implies additional solver calls, it is necessary to apply a parsimonious method. In the quoted example, a design-of-experiments technique was applied, which required 66 samples. This means that the total number of solver calls equals 216 in this analysis, which is a reasonable quantity.

From Table 12 it is evident that the results are quite satisfactory. Therefore, it can be concluded that MLP networks are a valuable solver substitute in case their global mapping domain is localized by focusing on the design points and in combination with a variance reduction technique.

5. Conclusions

The following conclusions can be drawn from this study:

1. The use of neural networks as functional approximators seems to be superior than that for data classification.
2. The SSE leads to better results than the CE cost function.
3. The sampling methods for generating the training population seem to have no effect in the learning phase as long as the resulting sample is reasonably dense. However, the generation of the sample set with compact uniform distributions is the recommended technique.
4. The RBF networks exhibit a better performance than the MLP in terms of accuracy and stability, as measured by the training and validation errors and also by the mode of the failure probability obtained after several trials. This is due to the locality of the mapping they perform. Also, their training times are

several orders of magnitude lower than those of the MLP networks, if use is made of their EA variant. This renders possible the use of this type for calculating the confidence interval of the $\hat{P}_f^*$ estimate after the procedure of Section 3 with a very low computational effort.

5. However, care must be exerted when using RBF networks in cases showing bifurcations in the Monte Carlo clouds, due to the fact that the different receptive fields are to be assigned to similar outputs, which is in contradiction to the philosophy of the method.
6. MLPs are a valuable tool for reliability analysis mostly in case they are combined with a variance reduction procedure that reduces their global mapping to the neighborhood of the design point for a single limit state function. Further research is needed in the case of a multiple one. They also lead to good estimations when combined with basic Monte Carlo if the failure domain is small. For practical applications of MLPs, the GNLM algorithm is recommended over the classical BPX.
7. A procedure has been proposed for arriving at an end estimate of failure probability and its confidence interval without resampling, which is required by basic and advanced Monte Carlo techniques. The procedure cannot be used in connection with the other solver-substitution approach, which is the RSM, because it gives a unique mapping of a set of input data. This increases the computational advantages of the ANN–MCM combination over these conventional approaches.

Despite the fact that the pattern recognition approach, which suggests the use of ANN for data classification, has not exhibited a performance as good as the one shown by functional approximation, it is worth some further research due to the fast training it allows and its advantages for dealing with complex and highly dimensional problems. In addition, more research is necessary on the combination of neural networks (especially RBF and SSE–GNLM) with advanced Monte Carlo methods, such as directional, conditional and importance sampling.

References

[1] G.L. Ang, A.H.S. Ang, W.H. Tang, Optical importance sampling density estimator, J. Eng. Mech. 118 (1991) 1146–1163.
[2] B. Ayyub, R.H. McCuen, Simulation-based reliability methods, in: C. Sundararajan (Ed.), Probabilistic Structural Mechanics Handbook, Chapman & Hall, New York, 1995, pp. 53–69.
[3] T.T. Binh, U. Korn, Scalar optimization with linear and nonlinear constraints using evolution strategies, in: B. Reusch (Ed.), Computational Intelligence Theory and Applications, Lecture Notes in Computer Science, vol. 1226, Springer, Berlin, 1997.
[4] G.E.P. Box, N.R. Draper, Empirical Model Building and Response Surfaces, Wiley, New York, 1987.
[5] C.G. Bucher, Adaptive sampling: an iterative fast Monte-Carlo procedure, Struct. Safety 5 (1988) 119–126.
[6] O.J. Chapman, A.D. Crossland, Neural networks in probabilistic structural mechanics, in: C. Sundararajan (Ed.), Probabilistic Structural Mechanics Handbook, Chapman & Hall, New York, 1995, pp. 317–330.
[7] A. Cichocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, Wiley, Chichester, 1993.
[8] O. Ditlevsen, H.O. Madsen, Structural Reliability Methods, Wiley, Chichester, 1999.
[9] I. Enevoldsen, M.H. Faber, J.D. Sorensen, Adaptive response surface techniques in reliability estimation, in: G.I. Schuëller, M. Shinozuka, J.T.P. Yao (Eds.), Structural Safety and Reliability, Balkema, Rotterdam, 1994, pp. 1257–1264.
[10] L. Faravelli, Response-surface approach for reliability analysis, J. Eng. Mech. 115 (1989) 2763–2781.
[11] E.J. Hartman, J.D. Keeler, J.M. Kowalski, Layered neural networks with Gaussian hidden units as universal approximations, Neural Comput. 2 (1990) 210–215.
[12] K.M. Hornik, M. Stinchcommbe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[13] J.E. Hurtado, D.A. Alvarez, Reliability assessment of structural systems using neural networks, in: Proc. ECCOMAS-2000, ECCOMAS, Barcelona, 2000.
[14] J.E. Hurtado, A.H. Barbat, Monte Carlo techniques in computational stochastic mechanics, Arch. Comput. Meth. Eng. 4 (1998) 3–30.
[15] H. Ishibuchi, H. Tanaka, Fuzzy regression analysis using neural networks, Fuzzy Set Syst. 50 (1987) 257–265.
[16] S.H. Kim, S.W. Na, Response surface method using vector projected points, Struct. Safety 19 (1997) 3–19.
[17] R.E. Melchers, Structural Reliability: Analysis and Prediction, second ed., Wiley, Chichester, 1999.
[18] M. Papadrakakis, V. Papadopoulos, N.D. Lagaros, Structural reliability analysis of elastic–plastic structures using neural networks and Monte Carlo simulation, Comput. Methods Appl. Mech. Engrg. 136 (1996) 145–163.
[19] H.L. Resnikoff, R.O. Wells, Wavelet Analysis, Springer, New York, 1998.
[20] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, 1996.
[21] R.Y. Rubinstein, Monte Carlo Optimization, Simulation and Sensitivity of Queueing Network, Krieger, Malabar, 1992.
[22] J.M.F. Saraiva, N.F.F. Ebecken, Application of neural networks in structural reliability analysis, Rev. Int. Met. Num. Calc. Dis. Ing. 14 (1998) 167–180 (in Portuguese).
[23] G.I. Schuëller, R. Stix, A critical appraisal of methods to determine failure probabilities, Struct. Safety 4 (1987) 293–309.
[24] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Academic Press, London, 1999.