World Applied Sciences Journal 2 (6): 594-601, 2007
ISSN 1818-4952

An Inverse Reliability Method Using Neural Networks and Genetic Algorithms

M.A. Shayanfar, S.R. Massah and H. Rahami

Department of Civil Engineering, Iran University of Science and Technology, Tehran 16846, Iran

Abstract: In the reliability analysis of a system, generally, the designer seeks a reliability index by employing parameters such as mean and variance of the existing variables. The inverse of the above problem can be stated as: "what are the mean and variance of the variables for a given reliability index β?". Various methods have been presented in the past regarding the solution of inverse reliability problems. The objective of this article is to attain a numerical solution to the inverse reliability problems by the use of neural networks and genetic algorithms. The results obtained from the sample problems presented in this article show that the analysis of inverse problems using the aforementioned methods is highly efficient. These methods have been presented in the framework of an example and then the results for additional numerical examples are compared with the results obtained by other classical methods.

Key words: Inverse reliability method · neural networks · genetic algorithms · reliability index · design parameters

INTRODUCTION

The inverse reliability problem arises when a value for design parameters related to a specific reliability level is required. This analysis can be performed by trial and error; using a forward reliability procedure like *FORM* [1, 2] and interpolating the design parameters at the required reliability level. More efficient methods for approaching the inverse reliability problems are reported in the literature. A reliability contour method has been described by Winterstein [3], which is applied to problems in offshore environmental loads. To extend the method to general limit state functions, Der Kiureghian [4] proposed an iterative algorithm based on the modified Hasofer-Lind-Rackwits-Fiessler (HLRF) scheme. In a practical application, the design variable may also be treated as random when there is a requirement to provide tolerance intervals. In this case, the actual design parameter could be either the mean or the standard deviation of the design variable. In addition, multiple design parameters are required if multiple reliability and/or geometric constraints are specified. In the work done by Li and Foschi [5], a general inverse reliability method was presented to approach not only for the single design parameter case, but also for the multiple-parameter problems with specific constraints. In their work, in order to obtain a unique solution to the problem, it was necessary that the number of design parameters be equal to the number of geometric and/or reliability-based constraints. In the work done by Sadovsky [6], an investigation on the trend of convergence in the above research has been presented. In the work done by Palaniaappan *et al.* [7], various inverse measures and their advantages along with the methods of computing the inverse measures are described. Xiaoping *et al.* [8], proposed an integrated framework for optimization by the employment of an inverse reliability strategy that uses percentile performance for assessing both the objective robustness and probabilistic constraints.

The method proposed in this article, in addition to its simplicity, also solves the problem of non-convergency and the problem which arises from the inequality of the number of design parameters versus the number of constraints. The use of any probability distribution function and the correlation coefficient of the related variables can simply be implemented in the proposed method.

Neural networks: The basic concepts of neural networks have broadly been presented by Haykin [9], Hagan *et al.* [10] and Fausett [11]. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections

Corresponding Author: Dr. M.A. Shayanfar, Department of Civil Engineering, Iran University of Science and Technology, Tehran 16846, Iran

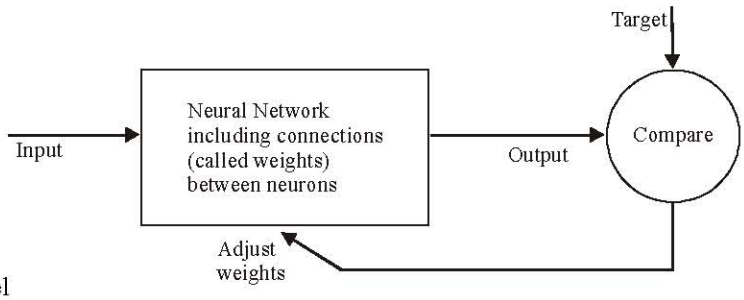

Fig. 1: A neuron model

between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Fig. 1. There, the network is adjusted based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are used in this supervised learning to train a network.

For the solution of complex engineering problems, where a mathematical approach is not available, the neural networks are generally used in order to significantly reduce the computing time. Neural networks might be time consuming at the training stage, but its testing is performed in an extraordinary short time.

Neural networks have been trained to approximate complex functions in various fields of application including pattern recognition, identification, classification, speech, vision and control systems. Generally, the neural networks are considered as strong tools in case of inverse problems. In other words, if there is a need for a linear or nonlinear mapping in a learning process, the neural networks can easily perform the task.

One of the most important neural networks is Multi-Layer Feed Forward (MLFF). The architecture of this network is composed of an input layer, an output layer and one or a number of hidden layers. Another important network is Radial Basis Function (RBF). This network is similar to the previous network, with the exception that it is composed of only one hidden layer. Radial basis networks may require more neurons than standard feed-forward networks, but often they can be designed in a fraction of the time required to train standard feed-forward networks. They work best when many training vectors are available [9-11].

Genetic algorithm: The genetic algorithm is a stochastic global search method that mimics the metaphor of natural biological evaluation. In other words, GAs are based on the mechanism of natural selection and survival of the fittest. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics.

Individuals are encoded as strings, i.e. chromosomes. The most commonly used representation in GAs is the binary alphabet {0,1}, although other representations can be used. The binary string can easily be changed into a real value, which this is known as decoding. Having decoded the chromosome representation into the decision variable domain, it is possible to assess the performance, or fitness of individual member of a population. This is done through an objective function that characterizes an individual's performance in the problem domain.

During the reproduction phase, each individual is assigned a fitness value derived from its raw performance measure given by the objective function. The recombination operator is used to exchange genetic information between pairs of individuals. The simplest recombination operator is that of single-point crossover.

A further genetic operator, called mutation, is then applied to the new chromosomes. In the binary string, mutation will cause a single bit to change its state.

The application of the above operators will eventually result in formation of a new generation of strings. The objective function improves as new generations are created. This process is repeated to the extent that convergence can be observed in the last generation [12]. This method has broadly been employed in different branches of science, e.g. the work done by Kaveh and Rahami [13, 14] on the optimization of structural trusses.

MATERIALS AND METHODS

The proposed method using the ANN and GA, is thoroughly explained in the framework of the following example.

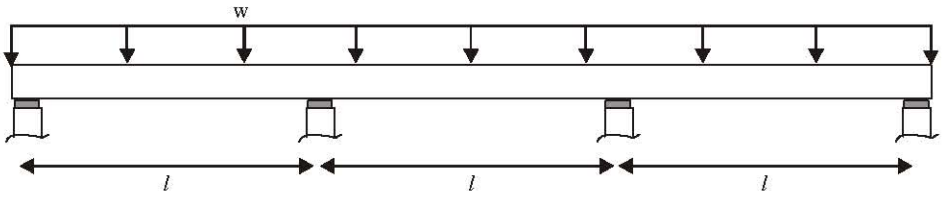

Fig. 2: A continuous beam with three equal spans (l = 5 m) under the uniformly distributed load (ω)

Table 1: Extremum parameters used in the network

	μ_ω	σ_ω	μ_E	σ_E	μ_I	σ_I	β
Min.	3.2536	0.1301	0.65e7	1.63e6	0.0003	0.049e-3	1.4289
Max.	17.4425	0.6977	3.49e7	8.72e6	0.0014	0.260e-3	3.5404

Consider a continuous beam with three equal spans (l =5m) under the uniformly distributed load *(ω)* as shown in Fig. 2. The allowable and maximum deflection of the beam are defined as *l/360* and $0.0069\omega l^4/EI$, respectively. The design variables in this problem are *ω*, *E* and *I*. It is assumed that only the span length, *l*, is deterministic. In case of considering the failure mode as the deflection mode; for a given reliability index, β, compute the mean and the variance *(μ, σ)* of the existing variables.

Neural networks: The inverse of the above problem is numerically computable. For example, if *ω(μ=10, =0.4)*, *E(2e7,0.5e7)* and *I(8e-4,1.5e-4)*, by using the *HLRF* method, a *β=3.18* is obtained. In this method, it is assumed that *ω, E* and *I* have a specified range of minimum and maximum, as illustrated in Table 1. Then, by choosing a value from the specified range for each variable, the reliability index, β, will be evaluated.

The analysis will be repeated several times (one hundred times in this example) in order to prepare the input and output data. Next, in this example, a neural network composed of three input neurons *(e.g. β, μ_ω, σ_ω)* and four output neurons *(e.g. μ_E, σ_E, μ_I, $_I$)* forms the neural architecture and then the training process begins. From amongst the one hundred inputs, *80* are selected for training and the rest are used for the testing process. The number of hidden layers and their neurons are selected randomly.

After training, for each testing data input *(β, μ_ω, σ_ω)*, four data outputs are achieved. These data outputs together with the values of μ_ω and σ_ω using the *HLRF* method, will lead to a new β. The network with the least testing error is considered the best. For example in this problem, the testing error is *2.23%* (for *MLFF*) which has been obtained by considering two layers with each having six and five neurons, respectively. Using the *RBF* network, an error of *2.26%* was obtained. Figures 3 and 4 show the results obtained from the comparison of the input β with the new β.

The time consumption of *RBF* training is significantly shorter than that by *MLFF*; as expected. The reason is that the *RBF* in its first layer is trained unsupervised to classify the input data.

Now, we shall consider a different case for a further investigation of this example. For instance, we assume that the parameters μ_E and$_I$ are given and the objective is to find the μ_I and σ_E. In this case, also, the neural network can easily lead to a reasonable solution to the problem. It is necessary to increase the inputs to five and reduce the number of outputs by two, simply by transferring two of the outputs to the inputs. Both of the networks *MLFF* and *RBF* have been tested. After the training stage, the *MLFF* network gives an error of *1.45%* for the reliability index, β, which corresponds to the testing data. Now, that we are looking for the parameters μ_I and σ_E, it is appropriate that the errors due to these two parameters be obtained without considering the β.

Figure 5 shows the comparison of the values of $_E$ obtained from the network to its corresponding real values. The maximum error in this case is *9.07%*. In Fig. 6, the same comparison is made for μ_I. Also, in this case, an error of *6.73%* was evaluated. Both of the computed errors are more than the errors corresponding to the β.

The reason is that, although for each value of μ and corresponding to the variables, one specific β can be obtained, but this is not a one-by-one mapping. In other words, it is possible that for different values of μ and σ, a same β be evaluated. Consequently, the error has slightly increased, which means that even though there is a minor variation of μ_I and σ_E, but eventually the target β with a negligible error has been calculated.

The *RBF* network, similar to the *MLFF* network, gives appropriate results for the β. Therefore, it is possible that with any combination of the inputs and the outputs, the required results be achieved.

In order to illustrate the capabilities of this method, additional examples will be presented in the following section. In these examples the selected networks are all *MLFF*. It should be noted that in the case

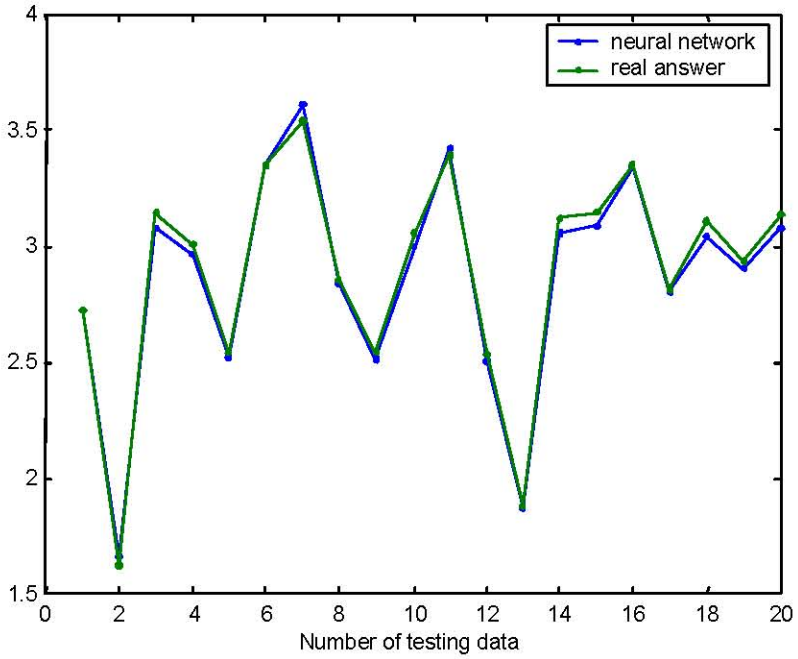

Fig. 3: Comparison of output from neural network (β; MLFF) with real answer

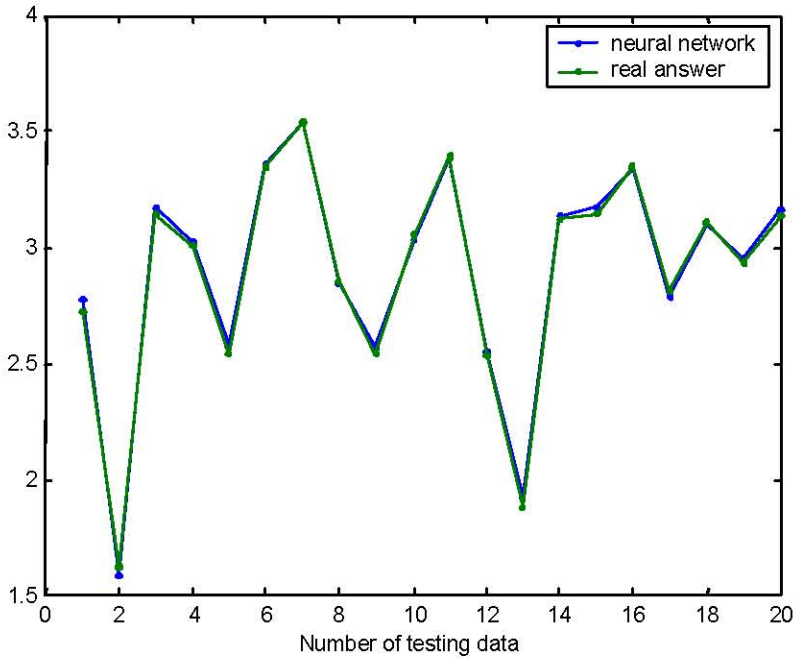

Fig. 4: Comparison of output from neural network (β; RBF) with real answer

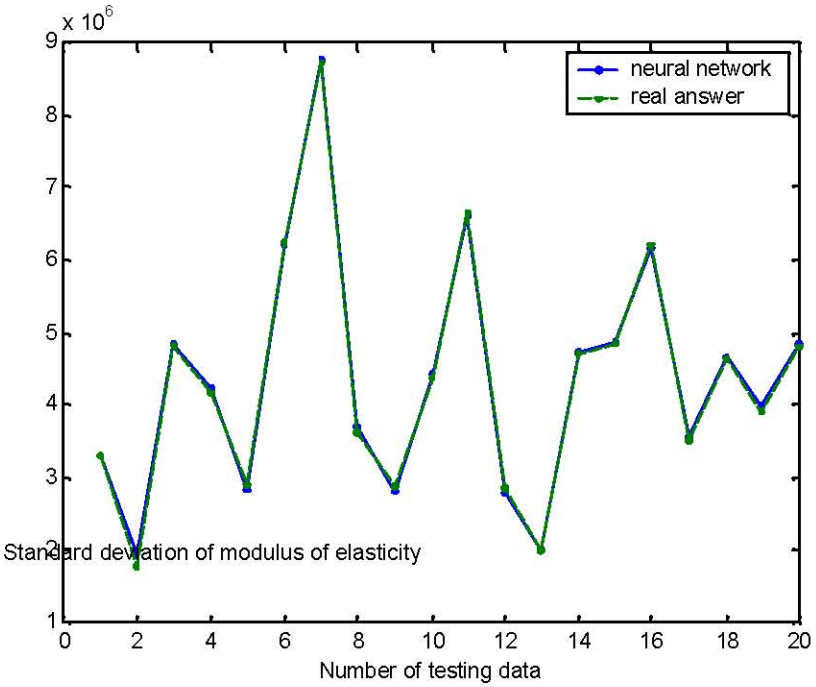

Fig. 5: Comparison of the output from neural network (s_E; MLFF) with real answer

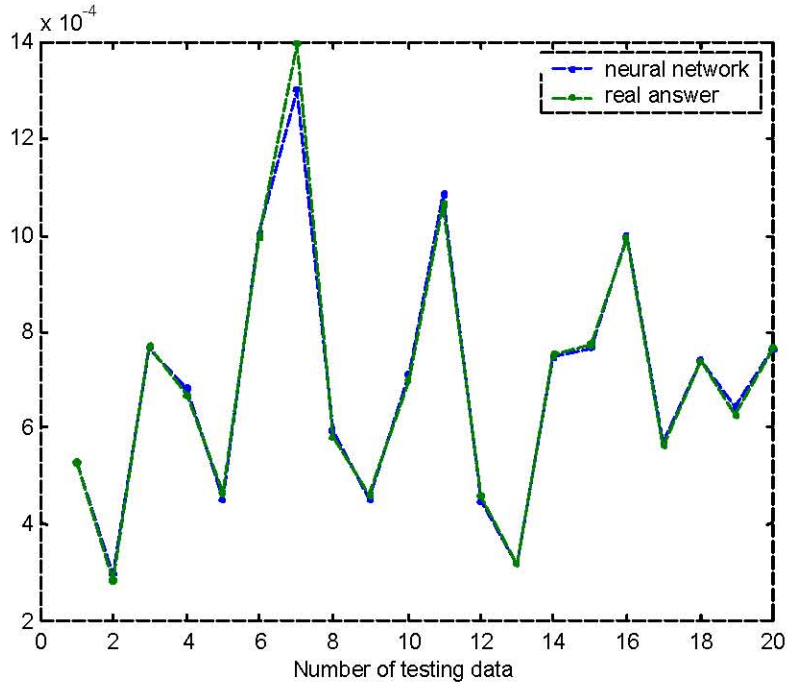

Fig. 6: Comparison of output from neural network (μ_I; MLFF) with real answer

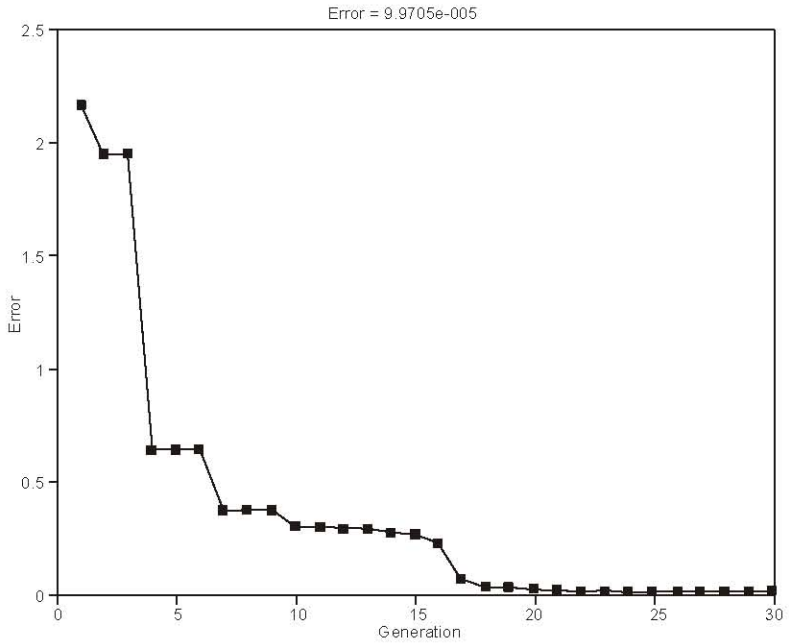

Fig. 7: The trend of minimization in GA

where numerous data are available, the performance of *RBF* is better than *MLFF*. Because, *RBF,* contrary to the *MLFF*, performs a classification on the data.

Genetic alghorithm: In this method, an objective function corresponding to the stated problem must be established. The objective function that is used is as follows:

$$\beta_a = \{ \beta_1, \beta_2, \ldots, \beta_n \} \; ; F(x) = [\text{norm}(\beta - \beta_a)]^2$$

where n is the number of limit state functions, β_a is the given target reliability index vector, x is a random variable vector (e.g. mean or/and variance), β is the reliability index corresponding to x and F being the objective function which must be minimized to zero.

At first, x is selected from a specified range on a random basis. Then, the corresponding β is determined using the classical methods discussed earlier. Therefore the objective function can be evaluated. This process is repeated, using the GA method, so that the objective function converges to zero or to a small value with a desired tolerance.

In the above example, by choosing the initial population as *30* and after *16* generations, by minimization of the objective function for $\beta_a = 3$, we will have $F = 4e\text{-}6$ corresponding to $\beta = 2.998$. If choosing $\beta_a = 3.3$, then we have $F = 2.5e\text{-}5$ corresponding to $\beta = 3.305$.

Numerical examples: Example 1. A limit state function with a single design parameter, as shown in Der Kiureghian *et al.* [4] is used to show the application of the proposed method. The function is

$$G = \exp[(-\theta\ (u_1+2u_2+3u_3)]-u_4+1.5$$

In the first case, the network is composed of one input and one output. After training and testing the network, we have: $\grave{e} = 0.3671$ and $\beta = 2.0000$. In the second case, that the random variable $\grave{e}$ having a normal Gauss distribution and a coefficient of variation of $v = 0.3$ has been considered, we will have: $\theta = 0.3725$ and $\beta = 2.0000$. The above results are in agreement with the results obtained from the method presented in the reference [5]. The same results are to be evaluated when using GA method.

Example 2. A set of three limit state functions G are given involving random variables, x_1, x_2, x_3 and x_4:

$$G1 = x_1^2 - 4x_2 - 2x_3x_4$$
$$G2 = 2\, x_1\, x_4 .\, x_2\, x_3$$
$$G3 = x_1\, x_2\, x_4 - 2x_3$$

With given target reliability indeces $\beta_1 = 3.0$, $\beta_2 = 3.5$ and $\beta_3 = 4.0$.

In the first case, the mean value for each of the variables x_1, x_2, x_3 is unknown. In the second case, where the variables are not correlated, the mean of the variables

Table 2: (Case 1) Statistics

Variable	Mean value	Coefficient of variation	Distribution type
X_1	?	0.01	Normal
X_2	?	0.2	Lognormal
X_3	?	0.1	Lognormal
X_4	1.0	0.1	Gumbel

Table 3: (Case 1) Solution

Variable	Mean value	C.o.v	β [4]	β (Neural Net.)
X_1	4.3638	0.01	3.0000	3.0002
X_2	2.1617	0.2	3.4999	3.5000
X_3	1.7831	0.1	4.0011	4.0000

Table 4: (Case 2) Statistics

Variable	Mean value	Coefficient of variation	Distribution type
X_1	6.0	?	Normal
X_2	?	0.2	Lognormal
X_3	?	0.1	Lognormal
X_4	1.0	0.1	Gumbel

Table 5: (Case 2) Solution

	ρ = 0.0	ρ = 0.8	ρ = 0.0	β [4]	β (Neural Net.)
S.D. of x_1	0.7682	0.8292	x_1	3.0009	3.0001
Mean of x_2	2.1961	3.3061	x_2	3.4999	3.5000
Mean of x_3	2.0787	1.9924	x_3	3.9996	3.9997

x_1 and x_2 and the coefficient of variation of x_3 are unknown. Tables 2-5, show a comparison of the results obtained by using the method presented in this paper and the results obtained by the other method [5].

Using the GA method in the second case for $\rho=0.0$, the trend of minimization is illustrated in the Fig. 7.

In this example, the population of the initial generation is selected to be *50*. After *30* generations, the objective function becomes $F=9.97e-5$ corresponding to $\beta = \{2.9994, 3.4942, 3.9919\}$.

DISCUSSIONS

If the range of the expected outputs be known, by training the network in the specified range, not only the convergence of the network will be enhanced, but also the reduction of the input data can be achieved. For instance, in the examples presented previously, for an unknown range of input data, the network outputs have a *1.2* percent error in β. To reduce this error, the network input data will be selected close to the input data of the previous network. Therefore, the precision of the results will significantly be improved.

Generally, the training of a network is time consuming and therefore it seems that the classical methods perform better; but in reality when numerous computations based on varying inputs are required (e.g. in optimization problems), the networks exhibit a much higher performance. The reason for this is that the networks take some time for training, but in case of having numerous input data, the network outputs will be evaluated at a fraction of a second; while in the classical methods the analysis will take a much longer time. Of course, in addition to the above, the difficulties related to the convergency confronted in the classical methods, must also be taken into account.

In order to comapre the two methods on the basis of time consumption, it can be stated that if the problem consists of several cases, ANN is better suited for application. But, for the problem with one case, GA would be more appropriate to use.

CONCLUSIONS

In this study new methods are proposed for the solution of inverse reliability problems using neural networks and genetic algorithms. This method is significantly efficient and can easily be extended for the solution of complex systems. For these problems, it is highly convenient to apply the *Monte Carlo Simulation* for creating the initial data. Some advantages of the proposed method are as follows:

- Simple to follow and not requiring the application of complex mathematical theories.
- o limitations are implemented in selecting the number and the type of inputs and outputs.
- The usual convergence difficulties present in the classical methods do not exist here.
- he distribution type and the correlation of the random variables can easily be implemented in the proposed method without the need to normalize or to uncorrelate the variables.

For future studies, it is suggested that the combination of Neural Networks and Genetic Algorithms be used in order to optimize the outputs in case of having specific constraints (e.g. for a given β, find the parameters of the variables to obtain the minimum weight).

REFERENCES

1. Hurtado, J.E., 2004. Structural reliability, Springer-Verlag, Berlin, Heidelberg,
2. Melchers, R.E., 1999. Structural reliability analysis and prediction, 2nd Edn., Wiley,
3. Winterstein, S.R., 1994. Ude TC, Cornell CA. Environmental parameters for extreme response: Inverse form with omission factors. Structural Safety and Reliability, Shinozuka and Yao editors, Rotterdam: Balkema,
4. Der Kiureghian, A., Y. Zhang and C.C. Li, 1994. Inverse reliability problem. J. Engineering Mechanics, ASCE, 120: 1154-1159.
5. Li Hong and O. Foschi Ricardo, 1998. An inverse reliability method and its application, J. Structural Safety, 20: 257-270.
6. Sadovsky Zoltan, 2000. Discussion on: An inverse reliability method and its application, J. Structural Safety, 22: 97-102.
7. Palaniappan, R., *et al.* 2006. Inverse reliability measures and reliability-based design optimization, J. Realiability and Safety, 1: 187-205.
8. Xiaoping Du *et al.*, 2006. An integrated frame work for optimization under uncertainty using inverse reliability strategy, J. Mechanical Design, 126: 562-570.
9. Haykin, S., 1999. Neural Networks: A comprehensive foundation, 2nd Edn., Prentice Hall,
10. Hagan, M.T. *et al.*, 1999. Neural network design, PWS Publishing Co.,
11. Fausett, L., 1994. Structural reliability analysis and prediction, 2nd Edn., John Wiley,
12. Goldberg, D.E., 1989. Genetic algorithms in search, optimization and machine learning, Addision-Wesley, Reading, MA,
13. Kaveh, A. and H. Rahami, 2006. Analysis, design and optimization of structures using force method and genetic algorithms, J. Numerical Methods in Engineering, 10: 1570-1584.
14. Kaveh, A and H. Rahami, 2006. Non-linear analysis and optimal design of structures via force method and genetic algorithms. J Computers and Structures, 84: 770-778.
15. MATLAB Guide, Neural Network Toolbox, The Math Works Inc., 2005.