Comput. Methods Appl. Mech. Engrg. 223–224 (2012) 70–80

Contents lists available at SciVerse ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Accelerated subset simulation with neural networks for reliability analysis

Vissarion Papadopoulos [*], Dimitris G. Giovanis, Nikos D. Lagaros, Manolis Papadrakakis

Institute of Structural Analysis and Seismic Research, National Technical University of Athens, Iroon Polytechniou 9, Zografou Campus, Athens 15780, Greece

A R T I C L E I N F O

Article history:
Received 6 January 2011
Received in revised form 17 February 2012
Accepted 21 February 2012
Available online 28 February 2012

Keywords:
Subset simulation,
Neural networks
Markov chain
Metropolis–Hastings
Reliability analysis

A B S T R A C T

Subset Simulation (SS) is a powerful tool, simple to implement and capable of solving a broad range of reliability analysis problems. In many cases however, SS leads to reliability predictions that exhibit a large variability due to the fact that the robustness of the SS prediction depends on the selection of an adequate width of the proposal distribution when applying the modified Metropolis algorithm. In this work a Neural Network-based SS (SS-NN) methodology is proposed in which NN are effectively trained over smaller sub-domains of the total random variable space which are generated progressively at each SS level by the modified Metropolis algorithm. NN are then used as robust meta-models in order to increase the efficiency of SS by increasing significantly the samples per SS level with a minimum additional computational effort. In the numerical examples considered, it is demonstrated that the training of a sufficiently accurate NN meta-model in the context of SS simulation leads to more robust estimations of the probability of failure both in terms of mean and variance of the estimator.

1. Introduction

Reliability analysis methods have been developed over the last two decades and have stimulated the interest for the probabilistic treatment of structures. Despite the theoretical advancements in the field of reliability analysis serious computational obstacles arise when treating realistic problems. In particular, reliability analysis of large-scale structural systems is an extremely computationally intensive task, which requires disproportionate computational effort for practical reliability problems, when a Monte Carlo Simulation (MCS) method is implemented. This is even more pronounced in cases where nonlinear Finite Element analysis is required for the prediction of the structural performance.

To alleviate this drawback advanced variance reduction-based simulation methods have been proposed in order to reduce the number of Monte Carlo simulations required for an accurate prediction of the probability of failure, such as importance sampling, line sampling and Subset Simulation (SS) [1–4]. However, the success of the aforementioned methodologies in accurately performing reliability analyses is not always guaranteed. In particular, SS leads to reliability predictions that in many cases exhibit a large variability due to the fact that the accuracy of SS prediction depends strongly on an arbitrary selection of the values of certain parameters for which no a priori knowledge is available.

In addition to the aforementioned methodologies, meta-models, such as Neural Networks (NN) have been successfully implemented in the framework of reliability analysis leading to cost-efficient but approximate predictions of the probability of failure [5,6]. Following these early approaches, NN approximations of the limit-state function were proposed in [7] combined with either MCS or with FORM approaches for handling the uncertainties, while similar approaches were presented in [8–10] in which NN based response surface methods were implemented in order estimate the reliability integral over the failure domain. In addition, surrogate models were implemented in [11] in order to obtain a drastic reduction of the computational labour implied by simulation techniques.

In the present study a methodology for computing the probability of structural failure by incorporating NN in the framework of the SS method is proposed. This way the computational efficiency of SS is substantially enhanced. The basic concept is to progressively train the NN at each SS level in a sequence of moving ranges (windows) defined by the upper and lower bounds of the samples generated by the Markov Chains of the modified Metropolis algorithm. This way the NN are effectively trained and used subsequently as robust meta-models in order to improve the efficiency of SS. This is accomplished by significantly increasing the samples at each SS level with a minimum additional computational effort. It is shown that the proposed approach leads to more robust estimations of the probability of failure both in terms of mean and variance, with respect to classical SS as well as with respect to NN-based brute force Monte Carlo simulation [5].

* Corresponding author. Tel.: +30 210 772 4158.
E-mail address: vpapado@central.ntua.gr (V. Papadopoulos).

0045-7825/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2012.02.013

2. The subset simulation method

The estimation of small failure probabilities P_F with the aid of MCS requires an excessive number of simulations in order to capture rare events. The basic idea of subset sampling is the subdivision of the failure event F into a sequence of M partial failure events (subsets) $F_1 \supset F_2 \supset \ldots \supset F_M = F$. The division into subsets (sub-problems) offers the possibility to transform the simulation of rare events into a set of simulations of more frequent events. The determination of the failure events F_i can be determined by presetting a series of limit values g_i, $i = 1, 2, \ldots, M$ such as [3]:

$$F_i = \{x : g(x) \leqslant g_i\} \tag{1}$$

where x being the vector of independent and identically distributed samples according to a probability density function (PDF) q, and $g(x)$ is the system performance function, usually a non-linear function of x. This enables the computation of the failure probability as a product of conditional probabilities $P(F_{i+1}|F_i)$ and $P(F_1)$ as follows:

$$P_F = P(F_M) = P(F_1) \cdot \prod_{i=1}^{M-1} P(F_{i+1}|F_i) \tag{2}$$

The determination of the failure events F_i and the partial conditional failure probabilities $P_i = P(F_{i+1}|F_i)$ strongly affects the accuracy of the simulation. Usually, the limit values $g_i|i = 1, 2, \ldots, M$ are selected in such way that nearly equal partial failure probabilities $P_i|i = 2, \ldots, M$ are obtained for each subset. However, it is difficult to specify in advance the limit values g_i according to a prescribed probability P_i. Therefore the limit values have to be determined adaptively within the simulation, as a function of the prescribed P_i [4].

2.1. Subset simulation algorithm

In the first step the probability P_1 is determined by direct Monte Carlo simulation:

$$P_1 = P(F_1) \approx \frac{1}{N} \cdot \sum_{k=1}^{N_1} I_{F_1}(x_k^{(1)}) \tag{3}$$

where $[x_k^{(1)} : k = 1, \ldots, N]$ are independent and identically distributed (i.i.d) samples simulated according to PDF q and $I_{F_1}(x_k^{(1)})$ is the indicator

$$I_{F_1}(x_k^{(1)}) = \begin{cases} 0 & \text{if } x_k^{(1)} \notin F_1 \\ 1 & \text{if } x_k^{(1)} \in F_1 \end{cases} \tag{4}$$

To obtain the conditional probabilities $P(F_{i+1}|F_i)$ of Eq. (2), the evaluation of the respective conditional probability density functions is required:

$$q(x|F_i) = \frac{I_{Fi}(x)q(x)}{P(F_i)} \tag{5}$$

Samples that follow the $q(x|F_i)$ conditional PDF can be generated numerically using a Markov Chain Monte Carlo procedure. Then the conditional probability at subset level $i + 1$ can be estimated as:

$$P(F_{i+1}|F_i) = P_{i+1} \approx \frac{1}{N_{i+1}} \sum_{k=1}^{N_{i+1}} I_{F_{i+1}}(x_k^{(i+1)}) \tag{6}$$

The samples of $i + 1$ subset are generated from the samples of subset i that are located in the failure region F_i as follows

$$x^{(i)} : g(x^{(i)}) > g_i, \quad i = 1, 2, \ldots, M-1 \tag{7}$$

The limit value g_i of the ith partial subset is determined during the simulation from a list of sorted pairs $(x_k^{(i)}, g(x_k^{(i)})), k = 1, 2, \ldots, N_i$ in ascending order, according to the values of $g(x_k^{(i)})$. The limit value g_i corresponds to the value of the performance function of the jth sorted sample given by

$$g_i = g(x_j^{(i)}), \quad j = P_i \cdot N_i \tag{8}$$

The number of samples N_i is selected in a way that partial probabilities P_i at each subset level are accurately estimated, in the context of MCS. The last failure probability $P(F_M|F_{M-1})$ can be estimated with the following expression:

$$P(F_M|F_{M-1}) \approx P_M = \frac{1}{N_M} \sum_{k=1}^{N_M} I_{F_M}(x_k^{(M)}) \tag{9}$$

The total failure probability P_F may then be computed as

$$P_F = \prod_{i=1}^{M} P_i \tag{10}$$

2.2. Markov chain Monte Carlo simulation

Markov Chain Monte Carlo Simulation (MCMCS), in particular, the Metropolis method, is a powerful simulation technique for simulating samples according to an arbitrary probability distribution function (PDF). It originates from the method developed by Metropolis and his co-workers for applications in statistical physics [12]. The advantages of MCMCS for solving reliability analysis problems is that it succeeds in simulating samples compatible to a conditional probability distribution $q(\cdot|F)$, which has been the main challenge in simulation–based reliability analysis. The algorithm used here is a modified version of the original Metropolis algorithm [4].

2.2.1. Modified Metropolis algorithm

For every $j = 1, \ldots, n$ let $p_j^*(y|x)$, called the 'proposal PDF', be a one-dimensional PDF for y centred at x with the symmetry property $p_j^*(y|x) = p_j^*(x|y)$. Generate a sequence of samples$\{x_1, x_2, \ldots\}$, from a given sample x_1 by computing x_{k+1} from $x_k = [x_k(1), \ldots, x_k(n)], k = 1, 2, \ldots$ as follows:

1. Generate a 'candidate' state x^*: for each component $j = 1, \ldots, n$, simulate y_j from $p_j^*(\cdot|x_k(j))$.

Compute the ratio $$r_j = \frac{q_j(y_j)}{q_j(x_k(j))} \tag{11}$$

Set $x^*(j) = y_j$ with probability

$$x^*(j) = \begin{cases} y_j \text{ with probability } \min\{1, r_j\} \\ x_k(j) \text{ with probability } 1 - \min\{1, r_j\} \end{cases} \tag{12}$$

1. *Accept/reject* x^*: Check the location of x^*. If $x^* \in F_i$, accept it as the next sample, i.e. $x_{k+1} = x^*$; Otherwise reject it and take the current sample as the next sample i.e. $x_{k+1} = x_k$.

$$x_{k+1} = \begin{Bmatrix} x_{k+1}^* & \text{when } x^* \in F_i \\ x_k & \text{when } x^* \notin F_i \end{Bmatrix} \tag{13}$$

Step 1 can be viewed as a 'local' random walk in the neighbourhood of the current state x_k , while Step 2 ensures that the next sample always lies in F_i, so as to produce the correct conditioning in the samples.

The choice of the proposal PDFs affects the deviation of the candidate state from the current one and controls the efficiency of the Markov Chain samples. Simulations show that the efficiency of the method is insensitive to the type of the proposal PDFs, and hence those which can be operated easily are the most preferable. For example, the uniform PDF centred at the current sample with width $2l_j$ is a good candidate for p_j^*. However, the spread of the proposal PDFs is of crucial importance since it affects the size of the region covered by the Markov chain samples, and consequently it controls the efficiency of the method. Very small spreads of the proposal PDFs tends to increase the dependence between successive samples due to their proximity, thus slowing down convergence of the estimator and occasionally causing ergodicity problems. On the other hand, excessive large spreads may reduce the acceptance rate, by increasing the number of repeated Markov Chain samples and thus slowing down convergence. The optimal choice for the spread of the proposal PDFs depends on a trade-off between acceptance rate and correlation due to proximity. Usually, the spread is chosen to be a fraction of the standard deviation of the starting samples, since no a priori knowledge is available that would lead to a convenient choice of spread [3,4].

The choice of the intermediate failure events also plays a key role in the SS robustness since it affects the values of the conditional level probabilities and hence the efficiency of the method. A commonly applied procedure is to choose a priori the intermediate threshold values g_i in eq.(1) in such way that nearly equal partial failure probabilities $P_i|i = 2, \ldots, M$ are obtained for each subset. However, as it is often difficult to specify the limit values g_i in advance according to a prescribed partial probability $P_{i,}$ these values are obtained during the SS procedure from a list of sorted pairs in ascending order so that the estimated conditional probabilities are equal to predefined values (see eq.(8)). This choice of the intermediate limit thresholds implies that the limit values g_i are no longer deterministic, since they depend on the conditional samples.

3. Artificial neural networks

Artificial Neural Networks (NNs) are information processing models configured for a specific application through a training process. Trained NNs provide with rapid mapping of a given input into the desired output quantities, thereby can be used as meta-models that enhance the computational efficiency of a numerical analysis process. This major advantage of a trained NN over a conventional numerical analysis procedures, under the provision that the predicted results fall within acceptable tolerances, is that results can be produced in a fraction of wall clock time, requiring orders of magnitude less computational effort than the conventional procedure [13–15,18].

Learning algorithms are divided into two major categories. (i) Algorithms that use global knowledge of the state of the entire network, referred as global techniques such as the direction of the overall weight update vector. In the conventional back-propagation learning algorithm, a global learning algorithm that is frequently used is the gradient descent algorithm. (ii) Local adaptation strategies, based on the temporal behaviour of the partial derivative of the corresponding weight. A local approach is more closely related to the neural network concept of distributed processing in which computations can be made independent to each other. Furthermore, it appears that for many applications local strategies achieve faster and more reliable prediction than global techniques despite the fact that they use less information [16]. In this work the Resilient (Rprop) training algorithms have been adopted which belong to the second category (local adaptation strategy) since they have been proved very efficient for training a NN [17]. In the Rprop learning algorithm the locally adaptive learning rates are bounded by upper and lower limits in order to avoid oscillations and arithmetic underflow. The NN learning process progresses iteratively, through a number of epochs. On each epoch the training patterns are submitted in turn to the network and the error is calculated by combining the actual outputs with the corresponding target values. In the present investigation a fully connected network is used (Fig. 1).

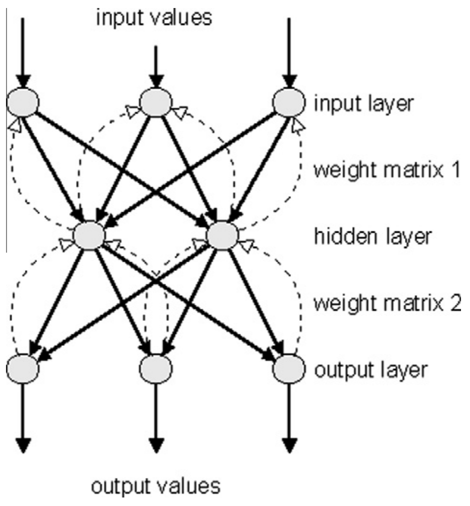

Fig. 1. Fully connected back-propagation neural network.

4. Neural networks based subset simulation (SS-NN)

The basic idea of the proposed methodology is to transfer information from the modified Metropolis algorithm to the NN training process in order to train effectively the NN in a sequence of moving ranges (windows), as these are produced by the upper and lower bounds of the random variables generated by Markov chains of the MCMCS at each successive SS level. The NN is then used as a robust meta-model in order to increase the SS efficiency by significantly enriching the sample space at each SS level with a minimum additional computational cost. In order to achieve that, a large number, with respect to standard SS procedure, of N_{MC} samples is generated at each subsequent SS level using MCMC algorithm and the corresponding failure function values are evaluated with the trained NN instead of the real numerical model.

Two alternative training strategies were implemented for the NN training procedure. The first (SS-NN1) introduces a closed multidimensional set Ω^M, subset of the total random variable space $\boldsymbol{\Theta}(\Omega^M \subset \boldsymbol{\Theta})$ defined by the upper and lower bounds of the random variables as these are generated by the increased, with respect to standard SS, space of MCMCS samples at each SS level. Subsequently, the NN is effectively trained over progressive and relatively small sets $\boldsymbol{\Omega}^M$ instead of $\boldsymbol{\Theta}$, with the minimum required training points using a Latin Hypercube Sampling (LHS) training scheme. The second strategy (SS-NN1) introduces a width l around each initial seed that the MCMCS uses as center in order to generate a random walk according to the proposal pdf $q(x|F_i)$ of eq. (5). This width defines a closed subset $\Omega^{(j)}$ around each seed j, defined by a lower and upper bound at distance $\pm l$ from the center. The NN is effectively trained at each $\Omega^{(j)}$ with a minimum computational cost using the LHS. Obviously, the subset $\boldsymbol{\Omega}^M$ in the case of SS-NN2 is constructed by the union of all $\Omega^{(j)}$ at each SS level.

The basic difference between the proposed methodology and the classical NN-based MCS [5] is that the SS-NN1 and SS-NN2 training processes are more effective since in both cases the NN is trained over smaller but sufficient progressive subsets of the total random variable space $\boldsymbol{\Theta}$. The NN is used subsequently as a more accurate meta-model, compared to a NN that is trained over the whole $\boldsymbol{\Theta}$, as in the case of classical NN-MCS [5]. The trained NN is then used to produce estimates of the failure function values required for the definition of the conditional parametric space

$F_M|F_{M-1}$ over which the probability $P(F_M|F_{M-1})$ of eq. (9) is calculated at each SS level. Detailed pseudo-algorithms of the two NN training alternatives are presented next.

4.1. First NN training approach (SS-NN1)

Step 1: At the *M*th SS level, generate a large number of N_{MC} samples for each random $x_i^{(M)}$ variable satisfying failure criterion F_{M-1}, using the Metropolis–Hastings algorithm as follows:

$$x^{(M)} = \begin{bmatrix} x_1^{(M)} \\ x_2^{(M)} \\ \vdots \\ x_n^{(M)} \end{bmatrix} \rightarrow \boxed{\text{MCMC}} \rightarrow \begin{bmatrix} x_{1j}^{(M)} \\ x_{2j}^{(M)} \\ \vdots \\ x_{nj}^{(M)} \end{bmatrix}, \quad j = 1,\ldots,N_{MC} \tag{16}$$

where $x_{ij}^{(M)}$, $i = 1,2,\ldots,n$ are the points of the MCMC random walk around $x_i^{(M)}$, n being the number of random variables.

Step 2: Define the limits of subset $\boldsymbol{\Omega}^M$ by determining its lower and upper bounds. These limits define the window in which NN is progressively trained at each SS level.

$$x_{\text{lower}}^{(M)} = \begin{bmatrix} x_{\text{lower},1}^{(M)} \\ x_{\text{lower},2}^{(M)} \\ \cdot \\ \cdot \\ \cdot \\ x_{\text{lower},n}^{(M)} \end{bmatrix} = \left\{ \min \begin{bmatrix} x_{1j}^{(M)} \\ x_{2j}^{(M)} \\ \cdot \\ \cdot \\ \cdot \\ x_{nj}^{(M)} \end{bmatrix}, \quad j = 1,\ldots,N_{MC} \right\} \tag{17a}$$

$$x_{\text{upper}}^{(k)} = \begin{bmatrix} x_{\text{upper},1}^{(M)} \\ x_{\text{upper},2}^{(M)} \\ \vdots \\ x_{\text{upper},n}^{(M)} \end{bmatrix} = \left\{ \max \begin{bmatrix} x_{1j}^{(M)} \\ x_{2j}^{(M)} \\ \vdots \\ x_{nj}^{(M)} \end{bmatrix}, \quad j = 1,\ldots,N_{MC} \right\} \tag{17b}$$

Step 3: Train the Neural Network within the multidimensional space $\boldsymbol{\Omega}^M$ using a LHS scheme in the following range

$$[x_{\text{lower}}^{(M)}, x_{\text{upper}}^{(M)}] = [\{x_{\text{lower},i}^{(M)}\}, \{x_{\text{upper},i}^{(M)}\}], \quad i = 1,\ldots,n \tag{18a}$$

In case that the successive training windows overlap, NN is trained over the remaining non-overlapping domain, which is defined as follows:

$$\Omega^M - (\Omega^M \cap \Omega^{(M-1)}) \tag{18b}$$

A schematic representation of the aforementioned procedure for the progressive NN training windows in two subsequent subset levels is presented in the following Fig. 2 for an ideal case of two random variables.

The number of points used for the NN training at the *M*th SS level spans the dashed area in Fig. 2. This number is equal to the number of the required analyses of the detailed model (or function evaluations) and hence the computational effort involved at each SS level.

Step 4: Estimate the analysis results (function evaluations) at each one of the N_{MC} sample points of $\boldsymbol{\Omega}^M$ generated in Eq. (23) using the trained NN with practically zero additional computational cost.

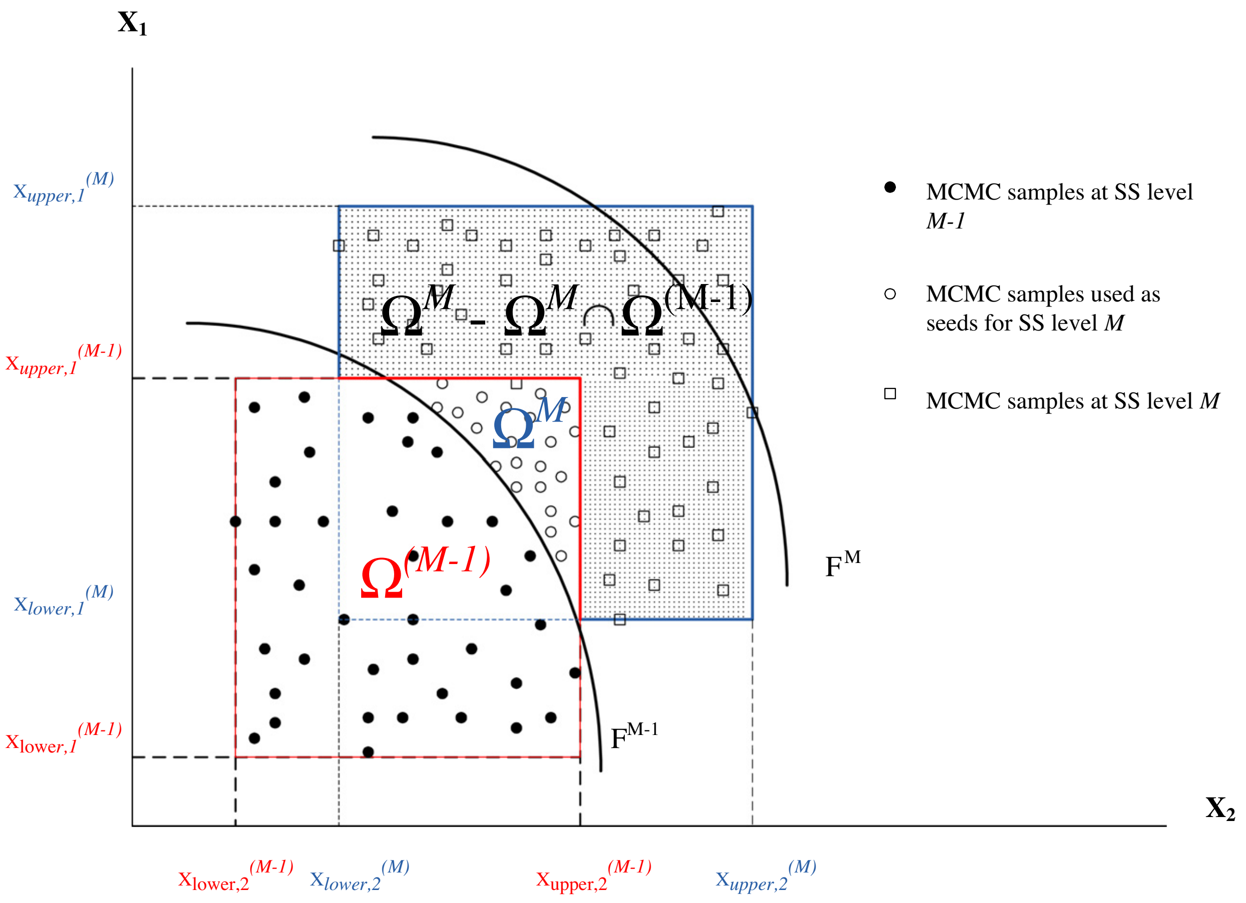

Fig. 2. Schematic representation of progressive NN training windows in two subsequent SS levels.

$$X_j^{(M)} = \begin{bmatrix} x_{1j}^{(M)} \\ x_{2j}^{(M)} \\ \vdots \\ x_{nj}^{(M)} \end{bmatrix} \rightarrow \boxed{\text{NN}} \rightarrow y^{(M)} = Y_j^{(M)}(x_j^{(M)}), \quad j = 1, \dots, N_{MC}. \tag{19}$$

Step 5: Estimate the failure criterion F_M according to eq.(9) and return to Step 1 for the subsequent SS Level. This is done by specifying intermediate target probabilities $P(F_M|F_{M-1})$ at each SS level (usually set at 10^{-1}) and identifying the failure event F_M by sorting x_j and $y^{(M)}$ according to the magnitude of the elements of $y^{(M)}$.

4.1.1. LHS based training

The Resilient Back Propagation (Rprop) algorithm is used for the NN training. The appropriate selection of I/O training data is an important factor for a successful training since the training set must include data over the entire range of the output space. Although the number of training patterns plays its own role in the accuracy of the predictions, the distribution of samples is of greater importance. The selection of the I/O training pairs is based on the requirement that the full range of possible results should be represented in the training procedure.

In the present study the sample space for each random variable is divided into equally spaced distances for the application of the NN simulation and for the selection of the suitable training pairs. The range for the random vector $\boldsymbol{x}$ is defined as follows:

$$\left.\begin{aligned} x_{\text{lower},i}^{(M)} &= \text{minimum}(x_{ij}^{(M)}) \\ x_{\text{upper},i}^{(M)} &= \text{maximum}(x_{ij}^{(M)}) \end{aligned}\right| \begin{aligned} i &= 1, \dots, n \\ j &= 1, \dots, N_{MC} \end{aligned} \tag{20}$$

where n is the number of random variables, N_{MC} is the total number of MCMC samples generated at Mth SS level. The samples used at each SS level for the NN training are generated with the Latin Hypercube Sampling method within the range $[x_{\text{lower},i}^{(M)}, x_{\text{upper},i}^{(M)}]$ for each random variable. As mentioned previously, NN is trained over the non-overlapping domains of the windows in two subsequent SS levels. A schematic representation of the LHS training scheme for an ideal case of two random variables and three subsequent SS levels is shown in Fig. 3.

The convergence of the training process is controlled by the prediction error. This is done either with a direct comparison of the predicted results with the target values computed by the conventional numerical procedure denoted as "exact", or by means of the root mean square error e_{RMS}, which gives a measure of the difference between predicted at each NN cycle and "exact" values.

$$e_{\text{RMS}}(\%) = \frac{|Out_{\text{Real}} - Out_{\text{NN}}|}{Out_{\text{Real}}} \cdot 100 \tag{21}$$

4.2. Second NN training approach (SS-NN2)

The pseudo-algorithm of the second NN training strategy is as follows:

Step 1: At the first SS level, generate N_{MC} function evaluations, sort the samples in ascending order and determine the failure criterion for the first subset level and the samples which will be used as seeds for the next subset level. The number of the samples will be $P_0 \cdot N_{MC}$, where P_0 is the target probability of the intermediate failure events usually taken as 0.1.

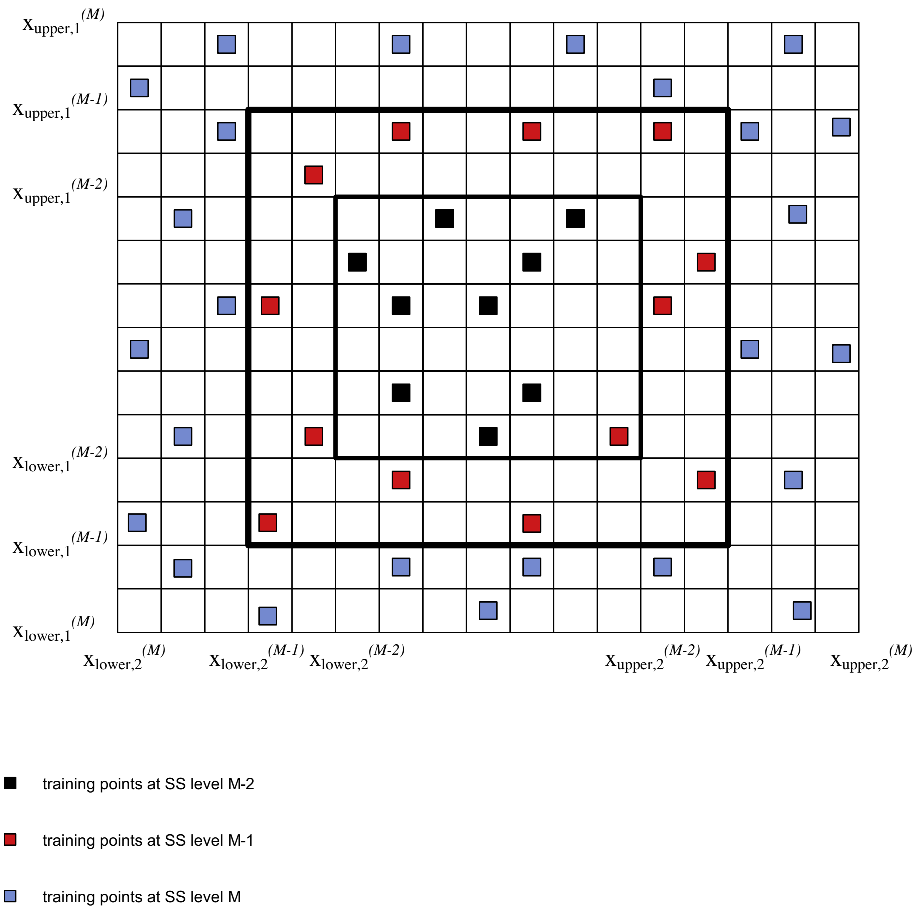

Fig. 3. LHS training scheme for two random variables and three subsequent SS levels.

Step 2: For every seed $x^{(j)}_{\text{center}} = (x^{j}_{\text{center},1}, \dots, x^{j}_{\text{center},n})$ satisfying the first failure criterion ($>F_1$), train a NN within subset $\Omega^{(j)} \subset \Omega^M \subset \Theta$ defined as:

$$\Omega^{(j)} = [x^{(j)}_{\text{lower}}, x^{(j)}_{\text{upper}}] = [\{x^{(j)}_{\text{center}} - l\}, \{x^{(j)}_{\text{center}} + l\}],$$
$$j = 1, \dots, P_0 \cdot N \tag{22a}$$

where l is a width that defines $\Omega^{(j)}$. Subset Ω^{M} in this case is given by the following union:

$$\Omega^M = \bigcup_{j}^{P_0 \cdot N} \Omega^{(j)} \tag{22b}$$

The total number of the NN training points at this step will be

$$N_{\text{train}} = T \cdot P_0 \cdot N_{MC} \tag{23}$$

where T is the number of points generated by LHS. A schematic representation of steps 1 and 2 is shown in Fig. 4.

Step 3: Use the trained Neural Network to perform N_{NN} ($N_{\text{NN}} \geqslant \text{N}_{\text{train}}$) function evaluations in the context of SS procedure and modified Metropolis algorithm. Estimate the failure criterion of the second subset and obtain the samples $P_0 \cdot N_{NN}$ used as seeds for the next SS level.

Step 4: Among the $P_0 \cdot N_{NN}$ samples select only N_{train} which will be used for the NN training. The selection is made stepwise in order to achieve the best representation of the parametric space (Fig. 5). In order to train the NN accurately and effectively, replace the values of the performance function for the N_{train} points with a real function evaluations.

Step 5: Train the Neural Network and generate N_{NN} MCMC samples.

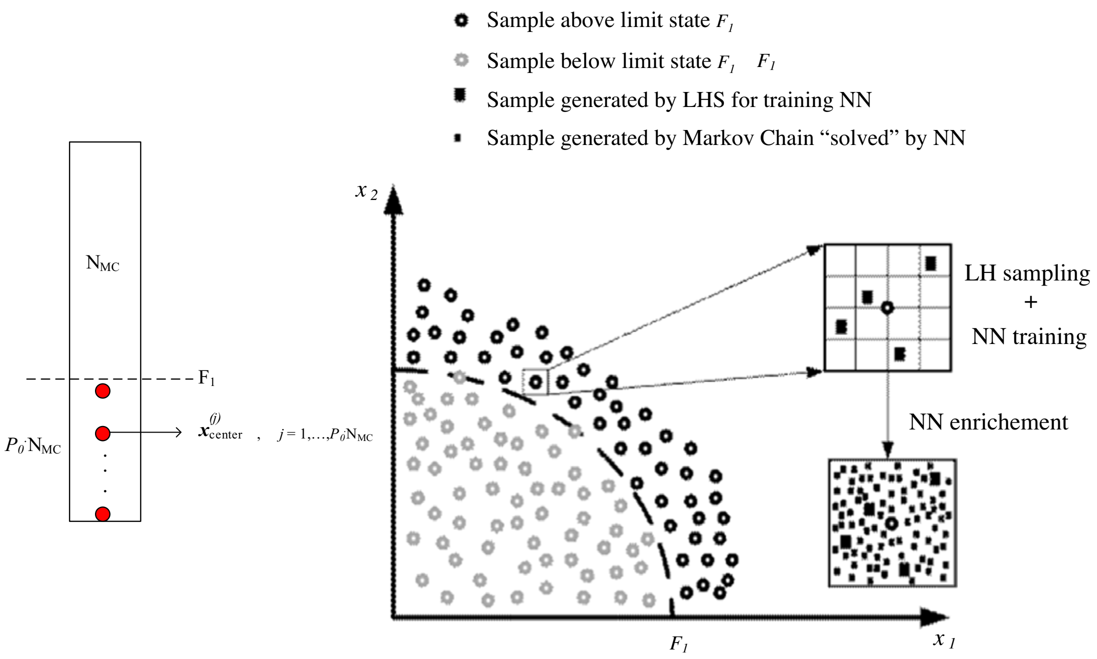
Sample above limit state F_1
Sample below limit state F_1 F_1
Sample generated by LHS for training NN
Sample generated by Markov Chain "solved" by NN
N_{MC}
F_1
$P_0 N_{MC}$
$x^{(j)}_{center}$, $j = 1,\dots,P_0 N_{MC}$
x_2
x_1
F_1
LH sampling + NN training
NN enrichement

Fig. 4. Schematic representation of steps 1 and 2 for the second NN training approach.

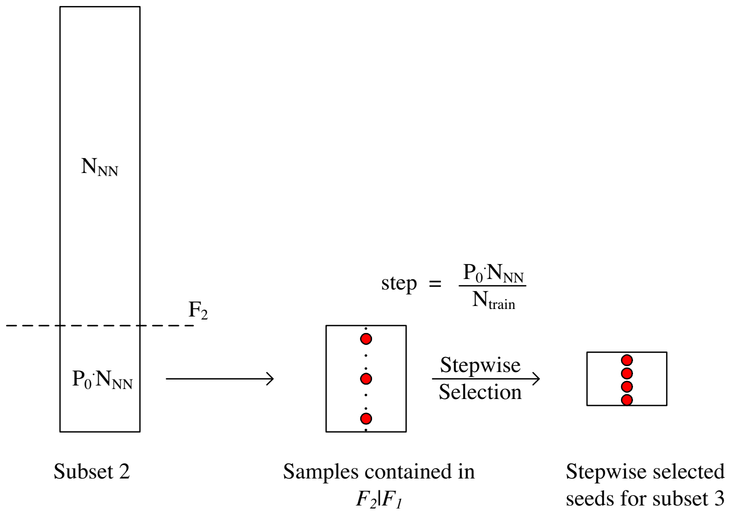
N_{NN}
F_2
$P_0 N_{NN}$
step = $\frac{P_0 N_{NN}}{N_{train}}$
Stepwise Selection
Subset 2
Samples contained in $F_2|F_1$
Stepwise selected seeds for subset 3

Fig. 5. Schematic representation of step 4.

Step 6: Repeat steps 4 and 5 until the target failure criterion is achieved.

4.2.1. Choice of witdh for the NN training in SS-NN2

It must be mentioned here that the selection of an appropriate width l for the NN training in SS-NN2, is of crucial importance since it affects the ability of the NN to accurately predict the values of the failure function within Ω^M. This is due to the fact that the NN is not trained over the total subspace that covers the random walk around each seed of the MCMCS, as in the case of Ω^Min SS-NN1. Therefore, the accuracy of SS-NN2 relies on the aliasing of the various $\Omega^{(j)}$ so that samples of the random walk around one seed that violate the limits of eq. 22, i.e. plus and minus the width of the proposal pdf, are expected to fall in to the training window of an adjacent seed. The selection of an appropriate width is not a priori known but it seems that a convenient choice is $l = 2$.

5. Numerical examples

In order to assess the performances of the SS-NN methods, two test examples are considered, one based on mathematical models and one based on a structural analysis problem. For the mathematical models, all random variables $x^{(j)}, j = 1, \ldots, n$ are assumed to follow the Normal distribution with mean value $\mu_x = 0$ and standard deviation $\sigma_x = 1$. The partial failure probabilities are predefined to $P_i = P_0 = 0.1$ for each subset. For this example a number of $n = 3$, 10 and 100 random variables have been used in order to test the efficiency of the proposed method and its sensitivity to the dimensionality of the problem. The target probability of failure is taken $P_F = 10^{-4}$.

5.1. Example 1: Mathematical model 1

The proposed method is first implemented in a simple linear analytic performance function, given by the following expression:

$$Y_1(x) = \sqrt{\sum_{i=1}^{n} (x^{(j)})^2}, \quad j = 1, \ldots, n \tag{24}$$

Table 1
SS estimation of failure probability P_F.

Subset simulation (10 simulation runs)			
Failure function		$Y = \sqrt{\sum_{i=1}^n x_i^2}$	
Random variables	Function evaluations per subset	Mean P_f	COV
3	1.000	9.89×10^{-5}	0.40
	10.000	9.96×10^{-5}	0.15
10	1.000	9.59×10^{-5}	0.34
	10.000	9.94×10^{-5}	0.15
100	1.000	9.89×10^{-5}	0.29
	10.000	9.98×10^{-5}	0.09

5.1.1. Subset Simulation

A standard SS is initially performed generating 1.000 and 10.000 samples per subset level. The spread of the uniform proposal PDF is taken $l = 2$. Table 1 depicts the mean value and COV of the estimated probability of failure, which corresponds to the reference failure criterion for the cases of 3, 10 and 100 random variables. The reference failure criterion which corresponds to the target probability of $P_F = 10^{-4}$, is calculated in all cases considered by means of brute force Monte Carlo Simulation with 10^6 samples.

From this Table it can be seen that the efficiency of the Subset Simulation is highly depended on the number of the samples generated in each Subset and less in the dimensionality of the problem. The coefficient of variation for the estimated probability significantly reduces when increasing the number of analyses and reaches a 10–20% value only when 10.000 samples per ubset are used. These results tend to improve when higher dimensionality problems are investigated (10 and 100 random variables). The bias of SS in the prediction of the mean P_F, is 1.1%, 4.1% and 1.1% for the 1.000 function evalutations per subset levels and 3, 10 and 100 random variables, respectively, while the same results for 10.000 function evaluations per SS level are 0.4%, 0.6% and 0.2%.

5.1.2. Proposed SS-NN methodologies

The same problem is now solved with the proposed SS-NN approaches using the procedures described in section 4.1 and 4.2. Tables 2–4 present the results for the two approaches for the cases of n=3, 10 and 100 random variables, respectively.

Form these Tables it can be seen that the performance of the two NN training alternatives is almost equivalent in terms of the COV of their estimations. For SS-NN1, the bias for the cases of 3, 10 and 100 variables in the mean P_F is 1%, 1% and 5%, respectively, while SS-NN2 produces improved predictions with a corresponding bias of 0.2%, 0.2% and 0.7%. In all cases examined, very good predictions of the mean P_F with a relatively small COV (10–20%) where achieved with the proposed approach using 1.000 training points and 10^6 MCMC samples per SS level, which is an order of magnitude less computational effort with respect to the 10.000 number of function evaluations per subset required by standard SS in order to reach the same accuracy.

5.2. Example 1: Mathematical model 2

The proposed method is also implemented in a non-linear analytic performance function, given by the following expression:

$$Y_1(x) = \sum_{i=1}^{n} X_i^5 - 5 \cdot \sum_{i=1}^{n} X_i + 12, \quad j = 1, \ldots, n \tag{25}$$

The results of the standard Simulation method and of the proposed methodologies are presented in the next Tables.

From Table 5 it can be seen that the bias of SS in the prediction of the mean P_F, is 2.5%, 2% and 1.1% for the 1.000 function evalutations per subset levels and 3, 10 and 100 random variables, respectively, while the same results for 10.000 function evaluations per SS level are 0.7%, 0.4% and 2%. Form Tables 6–8 it can be concluded

Table 2
SS-NN estimation of failure probability for the case of three random variables.

Failure function			$Y = \sqrt{\sum_{i=1}^n X_i^2}$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
3	1.000	10^4	9.89×10^{-5}	0.40	9.86×10^{-5}	0.20	1.09×10^{-4}	0.14
		10^5			9.94×10^{-5}	0.18	1.01×10^{-4}	0.07
		10^6			1.01×10^{-4}	0.11	9.98×10^{-5}	0.06

Table 3
SS-NN estimation of failure probability for the case of 10 random variables.

Failure function			$Y = \sqrt{\sum_{i=1}^{n} X_i^2}$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC Samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
10	1.000	10^4	9.59×10^{-5}	0.34	9.84×10^{-5}	0.24	9.93×10^{-5}	0.19
		10^5			1.07×10^{-4}	0.15	1.01×10^{-4}	0.12
		10^6			1.01×10^{-4}	0.12	9.98×10^{-5}	0.09

Table 4
SS-NN estimation of failure probability for the case of 100 random variables.

Failure function			$Y = \sqrt{\sum_{i=1}^{n} X_i^2}$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC Samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
100	1.000	10^4	9.89×10^{-5}	0.29	9.85×10^{-5}	0.23	9.81×10^{-5}	0.21
		10^5			9.91×10^{-5}	0.16	9.94×10^{-5}	0.13
		10^6			1.05×10^{-4}	0.12	9.93×10^{-5}	0.12

Table 5
SS estimation for 3, 10 and 100 random variables for the second mathematical function.

Subset simulation (10 simulation runs)			
Failure function		$Y_1(x) = \sum_{i=1}^{n} X_i^5 - 5 \cdot \sum_{i=1}^{n} X_i + 12$	
Random variables	Function evaluations per subset	Mean P_f	COV
3	1.000	9.75×10^{-5}	0.42
	10.000	9.93×10^{-5}	0.17
10	1.000	9.82×10^{-5}	0.32
	10.000	9.96×10^{-5}	0.15
100	1.000	9.89×10^{-5}	0.28
	10.000	1.02×10^{-4}	0.11

Table 6
SS-NN estimation of failure probability for the case of three random variables.

Failure function			$Y_1(x) = \sum_{i=1}^{n} X_i^5 - 5 \cdot \sum_{i=1}^{n} X_i + 12$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC Samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
3	1.000	10^4	9.75×10^{-5}	0.42	9.85×10^{-5}	0.21	1.09×10^{-4}	0.17
		10^5			9.90×10^{-5}	0.19	1.04×10^{-4}	0.12
		10^6			9.89×10^{-5}	0.13	9.92×10^{-5}	0.08

Table 7
SS-NN estimation of failure probability for the case of 10 random variables.

Failure function			$Y_1(x) = \sum_{i=1}^{n} X_i^5 - 5 \cdot \sum_{i=1}^{n} X_i + 12$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
10	1.000	10^4	9.82×10^{-5}	0.32	1.08×10^{-4}	0.25	9.89×10^{-5}	0.18
		10^5			9.88×10^{-5}	0.15	9.92×10^{-5}	0.14
		10^6			1.03×10^{-4}	0.10	9.96×10^{-5}	0.09

that again the performance of the two NN training alternatives is almost equivalent in terms of the COV of their estimations. For SS-NN1, the bias for the cases of 3, 10 and 100 variables in the mean P_F is 1%, 3% and 6%, respectively, while SS-NN2 produces improved predictions with a corresponding bias of 0.8%, 0.4% and 0.5%. Thus, in all cases examined, very good predictions of the mean P_F with a relatively small COV (10–20%) where achieved with the proposed approach using 1.000 training points and 10^6 MCMC samples per SS level, which is an order of magnitude less computational effort with respect to the 10.000 number of function evaluations per subset required by standard SS in order to reach the same accuracy.

Table 8
SS-NN estimation of failure probability for the case of 100 random variables.

Failure function			$Y_1(x) = \sum_{i=1}^{n} X_i^5 - 5 \cdot \sum_{i=1}^{n} X_i + 12$					
			SS		SS-NN1		SS-NN2	
Random variables	Function evaluations per subset	MCMC samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
100	1.000	10^4	9.89×10^{-5}	0.28	9.75×10^{-5}	0.22	9.81×10^{-5}	0.21
		10^5			9.88×10^{-5}	0.15	9.90×10^{-5}	0.13
		10^6			1.06×10^{-4}	0.12	9.95×10^{-5}	0.11

5.3. Example 2: 6-storey reinforced concrete building

The third test example considered in this study is the evaluation of the failure probability of a 6-storey reinforced concrete building of Fig. 4. The base nodes are fixed while the slabs are considered to act as diaphragms. The building is analyzed with a Non linear Static Pushover (NSP) analysis. A lateral load distribution corresponding to the fundamental mode is implemented [7,19]. The building has been designed to meet the Eurocode requirements of EC8 [19] and EC2 [20] design codes. Concrete of class 16/20 (nominal cylindrical strength of 16 MPa) and class *S*500 steel (nominal yield stress of 500 MPa) is assumed. The base shear is obtained from a response spectrum of soil type B characteristic periods: (T_B = 0.15 s, T_C = 0.50 s, T_D = 2.00 s) while the Peak Ground Acceleration (PGA) considered is equal to 0.31g. Moreover, the importance factor γ_I was taken equal to1.0, while the damping correction factor is equal to 1.0, since a damping ratio of 5% has been considered. The slab thickness is equal to 15 cm and is considered to contribute to the moment of inertia of the beams with an effective flange width. In addition to the self weight of beams and slabs, a distributed permanent vertical load of 2 KN/m^2 due to floor finishing partitions and an imposed load with nominal value of 1.5 KN/m^2, is considered (Fig. 6). Following EC8 for the seismic loading combination, dead loads are considered with their nominal values, while live loads with 30% of their nominal value.

Beam-column members are modelled with the inelastic force-based fibber element [21]. Three random variables were introduced, namely the concrete compression strength *fc,* the steel tensile strength *fs* and the steel Young modulus *Es* which were considered to differ in each storey, resulting a total number of 18 random variables. All random variables were assumed to follow the Gaussian distribution with parameters presented in Table 9.

Again, the target P_F is set at 10^{-3} which corresponds to a reference failure criterion g = 1.7219 of the top storey drift. As long as the network architecture is concerned, the Rprop and the Levenberg- Marquardt training algorithms with 10 hidden layers were chosen. Both the approaches of the proposed methodology were studied. The reference value for the failure criterion was obtained via crude MCS with 10^5 samples.

The same problem is solved using the proposed SS-NN method And 10^4, 10^5 and 10^6 samples generated by the NN in order to enrich the sampling space at each subsequent SS step, while 1.000 training points are used in each subset in order to train the NN with a total of 4000 NSP Analyses. Table 10 depicts the mean and COV of P_F obtained for the SS and the two NN training approaches. Again SS-NN2 produces better predictions, with respect to SS-NN1, on the mean P_F, with a bias of 2% instead of 3% and a COV. of 0.08 instead of 0.13 estimated through SS-NN1. The corresponding COV of the SS is 0.38.

5.4. Neural Network prediction error

As mentioned above, the key point of the proposed SS-NN approach is that it exploits the ability of NN to accurately predict the system response if trained in a properly selected sub-domain of the total space of the random variables. This is demonstrated in the following by computing the average e_{RMS} error of the NN prediction in the context of a SS-NN procedure. This error corre-

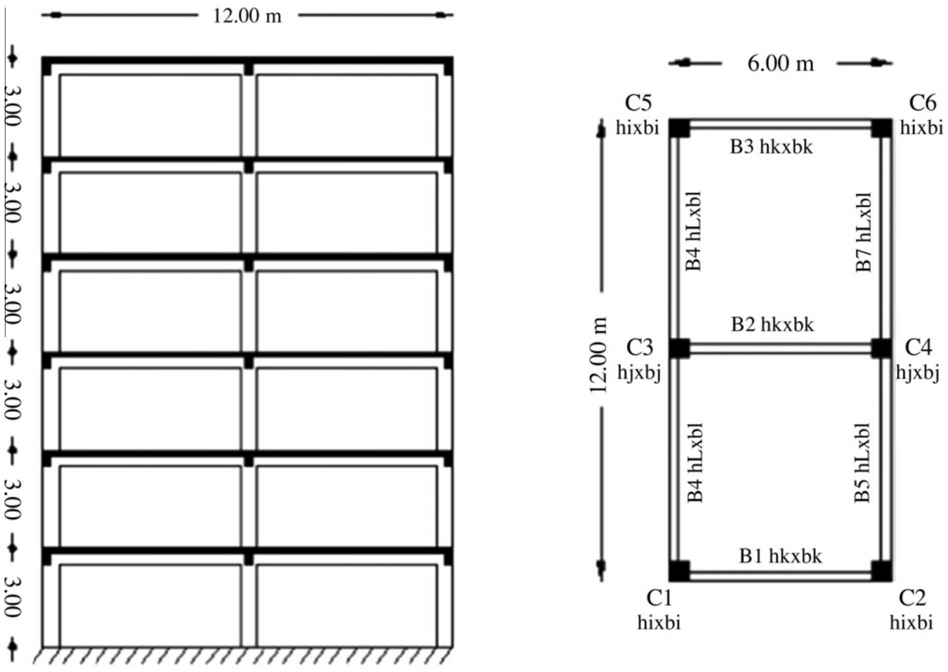

Fig. 6. Views of the 6-storey RC building.

Table 9
Random variable parameters of the Gaussian distribution, for the 6-storey RC building.

Random variables	Mean value	COV
Es	220 GPa	0.05
fs	500.000 kPa	0.05
fc	20.000 kPa	0.15

sponds to the mean error of 1.000 randomly selected MCMC samples, in the case of the mathematical function of Eq. (33). Fig. 7 presents the computed error for the case of 3, 10 and 100 random variables for different number of hidden layers for the two approaches of the proposed methodology in the estimation of the performance function. From this figure, it can be seen that in the first approach and for the case of 10 hidden layers and three random variables the NN prediction reaches a relative error of 2%. However, in the case of 10 and 100 random variables, the prediction error increases to 11%, and 45% respectively. For the case of 30 hidden layers the prediction errors reduces to 0.8%, 3.2% and 23% for the cases of 3, 10 and 100 random variables respectively. For the second approach the NN prediction error is slightly smaller. For the case of 3 random variables and 10 hidden layers it reaches 1% and for the cases of 10 and 100 random variables the prediction error increases to 9% and 45% respectively. Finally, for the case of 30 hidden layers the error is 0.5%, 2.2% and 21% for the cases of 3, 10 and 100 random variables respectively. These results show us the dependency of the quality of the results to the architecture of the Neural Network.

From the above Figure it can be observed that the relative error decreases with the increase of the number of the hidden layers of the Neural Network. In additioni, the error increases with the increase of the number of random variables. The latest obviously imposes the following limitation of the proposed approaches: The larger the dimensionality of the problem becomes, the more the quality of the NN results becomes poor requiring more training points for a succesful prediction.

5.5. SS-NN versus NN-MCS

The results of the SS-NN2 procedure, considered to be more robust, are further compared against results obtained via classical NN-based brute-force Monte Carlo Simulation (NN-MCS) as this was proposed in [5,6]. In order to perform a fair comparison between the two approaches, the total number of required detailed numerical analyses (or function evaluations) is kept the same for both methodologies. In the case of NN-MCS the training set of the random variable space is obtained with the previously described LH training scheme in the range $[-5\sigma, +5\sigma]$, σ being the standard deviation of the random variables. The brute-force MCS was performed for 10^6 samples. The results are obtained for the cases of 3 and 100 random variables and presented in Tables 11 and 12, respectively for the mathematical models of examples 1 and 2. At this point it is reminded that in all cases, considered the target 'exact' probability of failure is set at 10^{-4}. The COV for the Monte Carlo Simulation is

$$\delta = \sqrt{\frac{(1-p)}{N \cdot p}}. \tag{26}$$

From these Tables it can be seen that SS-NN2 produces significantly improved predictions of the mean P_F, with a bias of less than 1% in all cases considered. The corresponding NN-MCS bias of the

Table 10
SS-NN estimation of the failure probability P_F for the 6-storey RC building.

Failure function			Non-linear static pushover analysis (NSP)					
			SS		SS-NN1		SS-NN2	
Random variables	NPS per subset	MCMC samples per subset for the NN	Mean P_f	COV	Mean P_f	COV	Mean P_f	COV
18	1.000	10^4	9.86×10^{-4}	0.38	9.81×10^{-4}	0.22	9.91×10^{-4}	0.19
		10^5			9.94×10^{-4}	0.19	1.03×10^{-3}	0.11
		10^6			1.03×10^{-3}	0.13	1.02×10^{-3}	0.08

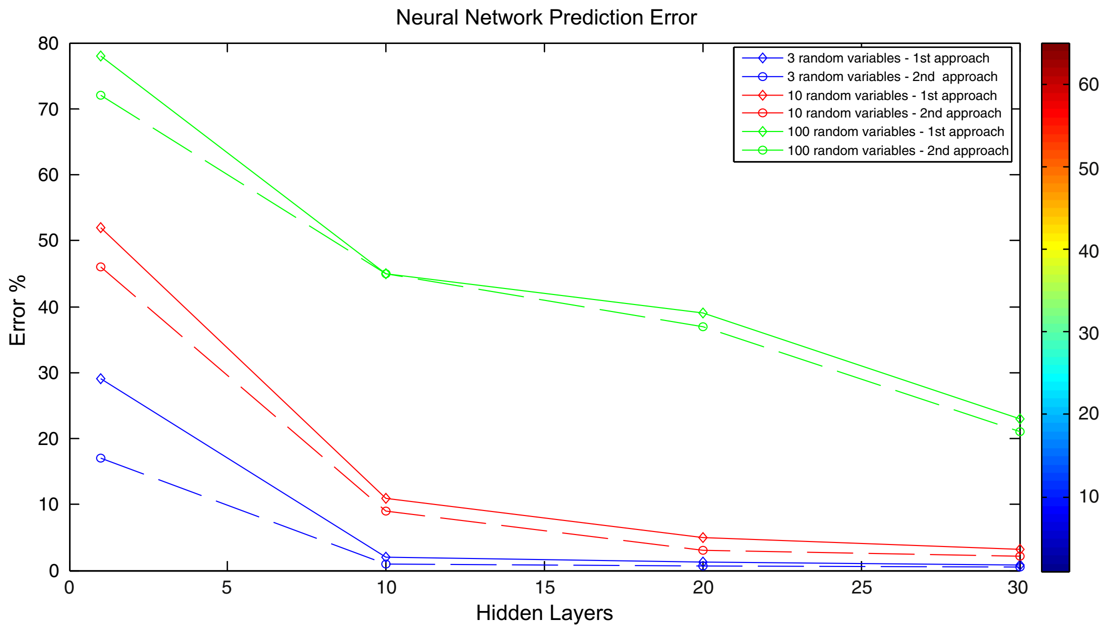

Fig. 7. NN prediction error for 3, 10 and 100 random variables.

Table 11
SS-NN vs NN-MCS estimation of failure probability for the case of 3 random variables.

Failure function		$Y=\sqrt{\sum_{i=1}^{n}X_i^2}$			
		NN-MCS		SS - NN2	
Random variables	Total function evaluations	Mean P_f	COV	Mean P_f	COV
3	4.000	8.98×10^{-5}	0.10	9.98×10^{-5}	0.06
	Failure function	$Y=\sum_{i=1}^{n}X_i^5-5\cdot\sum_{i=1}^{n}X_i+12$			
		NN-MCS		SS-NN2	
		Mean P_f	COV	Mean P_f	COV
	4.000	8.72×10^{-5}	0.10	9.93×10^{-5}	0.08

Table 12
SS-NN2 vs NN-MC estimation of failure probability for the case of 100 random variables.

Failure function:		$Y=\sqrt{\sum_{i=1}^{n}X_i^2}$			
		NN & MC		SS-NN2	
Random variables	Total function evaluations	Mean P_f	COV	Mean P_f	COV
100	4.000	8.98×10^{-5}	0.10	1.005×10^{-4}	0.12
	Failure function	$Y=\sum_{i=1}^{n}X_i^5-5\cdot\sum_{i=1}^{n}X_i+12$			
		NN & MC		SS-NN2	
		Mean P_f	COV	Mean P_f	COV
	4.000	8.72×10^{-5}	0.10	9.95×10^{-5}	0.11

estimation is more than 10%. Thus, the proposed methodology leads to a more than one order of magnitude reduction in the bias of the P_F estimation.

6. Conclusions

In this work a methodology is proposed for accurate and efficient system reliability analysis using a Neural Network-based Subset Simulation approach. This methodology takes advantage of the special characteristics of the SS in order to exploit the capability of a NN to efficiently approximate limit state structural performance, provided that it is trained over a properly selected domain. The basic idea is to progressively train the NN at each SS level in a sequence of moving windows defined by the upper and lower bounds of the samples generated by the Markov Chains of the modified Metropolis algorithm. This way the NN are effectively trained and used subsequently as robust meta-models in order to improve the efficiency of SS. This is accomplished by significantly increasing the samples at each SS level with a minimum additional computational effort. It is shown that the proposed approach improves the efficiency of classical SS reaching the same accuracy, both in terms of mean and COV of the estimation, with one order of magnitude less computational effort. In addition it leads to more robust predictions, with respect to classical NN-based brute force Monte Carlo simulation, with more than one order of magnitude less error in the estimation of the mean probability of failure.

References

[1] C.G. Bucher, Adaptive sampling – an iterative fast Monte Carlo procedure, Struct. Saf. 5 (1998) 119–126.
[2] G.I. Schuëller, H.J. Pradlwarter, Benchmark study on reliability estimation in higher dimensions of structural systems – an overview, Struct. Saf. 29 (2007) 167–182.
[3] S.K. Au, J.L. Beck, A new adaptive importance sampling scheme, Struct. Saf. 21 (1999) 135–158.
[4] S.K. Au, J.L. Beck, Estimation of small failure probabilities in high dimensions by subset simulation, Probab. Eng. Mech. 16 (4) (2001) 263–277.
[5] M. Papadrakakis, V. Papadopoulos, N.D. Lagaros, Structural reliability analysis of elastic-plastic structures using neural networks and Monte Carlo simulation, Comp. Meth. Appl. Mech. Eng. 136 (1996) 145–163.
[6] J.E. Hurtado, D.A. Alvarez, Neural–network-based reliability analysis: A comparative study, Comput. Methods Appl. Mech. Engrg. 191 (1–2) (2002) 113–132.
[7] N.D. Lagaros, A.Th. Garavelas, M. Papadrakakis, Innovative seismic design optimization with reliability constraints, Comput. Methods Appl. Mech. Engrg. 198 (1) (2008) 28–41.
[8] E. Mesbahi, Y. Pu, Application of ANN-based response surface method to prediction of ultimate strength of stiffened panels, Journal of Structural Engineering 134 (10) (2008) 1649–1656.
[9] J. Cheng, Q.S. Li, Reliability analysis of structures using artificial neural network based genetic algorithms, Comput. Methods Appl. Mech. Engrg. 197 (45–48) (2008) 3742–3750.
[10] C. Bucher, T. Most, A comparison of approximate response functions in structural reliability analysis, Probab. Engrg. Mech. 23 (2–3) (2008) 154–163.
[11] J.E. Hurtado, Filtered importance sampling with support vector margin: A powerful method for structural reliability analysis, Struct. Saf. 29 (1) (2007) 2–15.
[12] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, Equations of state calculations by fast computing machines, J Chem Phys 21 (6) (1953) 1087–1092.
[13] N.D. Lagaros, M. Papadrakakis, Improving the condition of the jacobian in neural network training, Adv. in Engng Soft. 35 (1) (2004) 9–25.
[14] M. Papadrakakis, N.D. Lagaros, Y. Tsompanakis, Structural optimization using evolution strategies and neural networks, Comput. Methods Appl. Mech. Engrg. 156 (1998) 309–333.
[15] D.J.C. MacKay, A practical Bayesian framework for backprop networks, Neural Comput. 4 (1992) 448–472.
[16] W. Schiffmann, M. Joost, R. Werner, Optimization of the back-propagation algorithm for training multi-layer perceptrons, University of Koblenz, Institute of Physics, Technical Report, 1993.
[17] M. Riedmiller, H. Braun, A direct adaptive method for faster back-propagation learning: the RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks (ICNN), San Francisco, 1993, pp. 586–591.
[18] M. Riedmiller, Advanced supervised learning in multi-layer perceptrons – from back-propagation to adaptive learning algorithms, Int. Journal of Computer Standards and Interfaces 16 (1994) 265–278.
[19] EN 1998–1(2003): Eurocode 8: Design of Structures for Earthquake Resistance. Part 1: General rules, seismic actions and rules for buildings. Commission of the European Communities, European Committee for Standardization, October 2003.
[20] PrEN 1992-1-1(2002): Eurocode 2: Design of Concrete Structures – Part 1: General Rules and Rules for Buildings. Commission of the European Communities, European Committee for Standardization, December 2002.
[21] I. Papaioannou, M. Fragiadakis, M. Papadrakakis, Inelastic analysis of framed structures using the fibre approach, in: Proceedings of the 5th International Congress on Computational Mechanics (GRACM 05), Limassol, Cyprus, June 29–July 1.