SLOPE FAILURE PROBABILITY AT CRITICAL VALUE OF SAFETY FACTOR

Angela NEAGOE[1], Radu POPA[2]

A very efficient methodology to obtain the slope failure probability at minimum value of 1 for safety factor is presented. Using a genetic algorithm (GA), the minimum safety factor at specified values of geometrical and soil data is derived. This GA model produces lessons for learning and validation of an artificial neural network (ANN), able to give the correct value of minimum safety factor at imposed values of uncertain soil parameters. The ANN is then used in a probabilistic analysis based on the subset simulation (SS) and Markov chain Monte Carlo simulation (MCMCS) to provide the answer for the question of interest. The Latin hypercube sampling method has been preferred in the key steps of methodology. Numerical simulations using various data sets confirm the expected results. These results can be taken into account in any reliability analyses of such slope design problem.

Keywords: small slope failure probability, genetic algorithm, artificial neural network, subset simulation, Markov chain Monte Carlo simulation.

1. Introduction

Due to the uncertainties associated with soil properties and their spatial variability, a minimum value of 1.5 is accepted for safety factor in the slope design. However, this practice offers no information concerning the reliability of the structure.

Knowing that the real failure occurs at a safety factor value less than or equal to the critical value of 1, any risk analysis requires the occurrence probability evaluation of such situation.

Although the use of stochastic concepts in structural safety started over half of century ago (e.g. Freudenthal, [1]), the computational difficulties of probabilistic analysis for rare events prevented, for a long time, getting some concluding results.

Only after the year 2000, efficient techniques for small failure probabilities estimation were developed (e.g. Au and Beck, [2]; Cho, [3]; Zio and Pedroni, [4]). They are based on new sampling methods, on improvement of the standard Monte

[1] Lecturer, Power Engineering Faculty, University POLITEHNICA of Bucharest, Romania, e-mail: angela_neagoe@yahoo.com
[2] Professor, Power Engineering Faculty, University POLITEHNICA of Bucharest, Romania

Carlo simulation (MCS) by subset simulation (SS) and using Markov chains, and on artificial intelligence methods adapted to this domain, as well.

Regarding the slope stability problem, it is known that, for a given geometry and specified soil proprieties, obtaining the minimum safety factor and the corresponding sliding surface is – in fact – a nonlinear and hard optimization problem. Including this task in any type of Monte Carlo analysis is prohibitive as computation time.

Thus, the present study proposes a genetic algorithm (GA) model, applied in the vertical slices method (Fellenius, [5]) in order to find the minimum safety factor in deterministic conditions, for a specified slope geometry.

Using the Latin hypercube sampling method (Stein, [6]), sets of representative samples (containing vectors of uncertain parameters) were then generated. With the mentioned GA model, the minimum safety factors were obtained for each such vector and sets of lessons for learning, and for validation of an artificial neural network (ANN) were formed. After validation, this ANN provides directly and simple the output quantity (the minimum safety factor), for specified values of input quantities (vector of uncertain parameters). Cho [3] mentions the use of an ANN, but for which the correct relationship between inputs and output was obtained with a different slope stability analysis method, using a commercial software package.

Finally, ANN was used in the evaluation of small failure probabilities using subsets Monte Carlo simulation, with Markov chains. The standard Monte Carlo simulation is ineffective even in these conditions because the required number of minimum safety factor values is inversely proportional to the failure probability when this is very small. On the contrary, in the subsets simulation (SS), the initial probabilistic domain of uncertain variables is successively reduced. At each intermediate level, the new samples of vectors with uncertain parameters are generated using Monte Carlo simulation with Markov chains, according to the conditional probability of being placed in this subset. Thus, ANN will supply values of the minimum safety factor corresponding only to rare events, possible in this probabilistic subset.

It is true that, for static or time-invariant reliability problems, sampling procedures have been developed (e.g. the importance sampling method) which give good results if the number of uncertain parameters is reduced (e.g. Rubinstein, [7]; Melchers, [8]).

Still, the coupling between the GA model in order to train an ANN and use of the ANN in MCS subsets with Markov chains, seems to be the most efficient way for a suitable evaluation of small slope failure probabilities.

2. GA model for the minimum safety factor evaluation in deterministic conditions

Although there are more sophisticated methods for slope stability analysis (e.g. Griffiths, [9]; Matsui, [10]), here was adopted, for illustration, the classical method of vertical slices in the following assumptions: i) homogeneous and isotropic soil, having the cohesion c, friction angle φ and unit weight γ as uncertain parameters, but having normal probability density functions (p.d.f.) – $f_j(x_j)$, j = 1, 2, 3 – with mean and standard deviations known from data measurements; ii) soil is unsaturated and the effect of pore pressure is neglected; iii) bottom ground is rigid, and the height H and slope m are known; iv) sliding surface is a circular arc which passes through the tail of slope.

By choosing the origin of the axes in C point, like in figure 1.a., the safety factor is given by relation:

$$S = \frac{\mathrm{tg}\varphi \sum N_i + cL_{AC}}{\sum T_i} \tag{1}$$

where $T_i = G_i \sin\alpha_i$ and $N_i = G_i \cos\alpha_i$ are the tangential and normal components of the soil weight G_i in slice i, having an unit depth and an area of a_i ($G_i = a_i\gamma$). The angle between the vertical direction and the radius from *O* to the slice i middle is α_i, and L_{AC} is the length of circular slip surface.

In deterministic conditions (H, m, c, φ and γ specified) the equilibrium state occurs for the safety factor minimum value, but the position of slip circle center is unknown.

Considering the above, the problem to be solved can be formulated as:

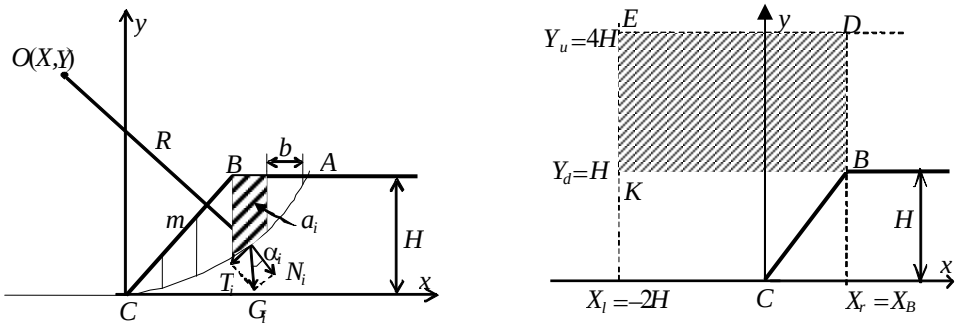
$$\min\left[F(X,Y)\right] \tag{2}$$

with *F* being here a performance function ($F \equiv S$), and restricted by:

$$\begin{aligned} X_l \le X \le X_r \\ Y_d \le Y \le Y_u \end{aligned}, \tag{3}$$

where the allowed domain for the *O* center position was limited like in figure 1.b. by: $X_l = -2H$, $X_r = X_B$, $Y_d = H$ şi $Y_u = 4H$, according to various references (e.g. Priscu, [11]).

There are many studies in the literature about GAs and their applications (e.g. Michalewicz, [12]; Goldberg, [13]; Chambers, [14], De Jong, [15]) so here is presented only a brief description of the implementation used in this problem.



a) **b)**

Fig.1.a) Sketch of slope defining; b) The permitted domain for the center position *O*

The GA works simultaneously with populations having *N* real-coded solutions (X_j, Y_j), $j = 1, N$. In the initial population, the pairs X_j, Y_j are randomly generated from the domain specified by the constraints (2). For each j solution the safety factor S_j can be computed using relation (1) and then its fitness function $E_j = 1/S_j$ - that has to be maximized.

To obtain the next population /generation, a random selection procedure is applied *N* times, so that the individuals with good fitness have a higher probability of selection. The normalized geometric ranking was used here as selection method (Houck et al, [16]). After each two such selections, the pair of parent solutions generates two children new solutions by crossover and mutation operators. Crossover is randomly performed by arithmetic crossover operator with probability of $p_c = 0.9$, while the nonuniform mutation operator was applied with a probability of $p_m = 0.05$.

The solutions X_j, Y_j, $j = 1, N$ from the new generation are evaluated by computing S_j and E_j, and it is expected that they will be better than the ones from the previous population.

The selection, crossover and mutation operations are repeated obtaining a new generation which is then evaluated, and the process stops after a specified number of generations, or due to a lack of improvement in the best solution over a specified number of generations. Here *N*=80 solutions per generation are used and the runs were stopped after 15 000 generations.

The geometrical slope data are supposed *H*=9 m and *m*=2, while for the normal p.d.f. of uncertain variables the following parameters are considered:

$\bar{c} = 18.4\,\text{kPa}$ and $\sigma_c = 2.76\,\text{kPa}$; $\bar{\varphi} = 14^{\text{o}}$ and $\sigma_\varphi = 2.1^{\text{o}}$; $\bar{\gamma} = 18\text{kN/m}^3$ and $\sigma_\gamma = 1.44\text{kN/m}^3$.

From several GA runs for the same values of c, φ and γ, it could be seen that the safety factor $S_{\min}$ resulted identically in each run. The corresponding positions of *O* center are approximatively the same, but so that the radius *R* remains identical. A similar behaviour was observed for different uncertain variabiles vectors, concluding that the proposed GA model is suitable for this problem.

In order to prepare for ANN learning and validation, four data sets were generated, each including 50 vectors (c, φ, γ).The Latin hypercube sampling method was used, which assures a representative covering of the entire probabilistic domain, defined here as $\mu_i \pm 3\sigma_i$, i=1,2,3 for each uncertain variable (with μ - the mean). All the four sets (D1, D2, D3 and D4) contain different samples, but probabilistic distributed after the same methodology.

Each data set *D* was run with the GA model which provided the 50 values of $S_{\min}$ corresponding to vectors (c, φ, γ) of the set. These data represent a lesson set.

In order to diversify the available material for ANN stage, the four sets of lessons were concatenated two by two, and so 12 sets, each having 100 lessons (*D*12, *D*13,..., *D*43) were obtained.

3. ANN for deriving the relationship $S_{\min}$– uncertain parameters

A classical architecture was adopted for ANN, of multi-layers feed-forward type, having an input layer, one hidden layer and an output layer, as shown in figure 2.

The neurons from the input layer introduce into the network the values of uncertain parameters, and the output neuron provides the corresponding value of the minimum safety factor. For the hidden layer 7 neurons were accepted, the recommended number to obtain a good approximation (Hornik et al., [17]).

between each input neuron and each hidden neuron, w_h, and respectively between hidden neurons and the output one, w_e, are initially randomly generated. The weighted signals entered in the hidden neurons are processed using a transfer function to generate signals towards the output neuron. This processes identically the received weighted signal and provides the value of $S_{\min}$. Only the logistic sigmoid transfer functions were used.

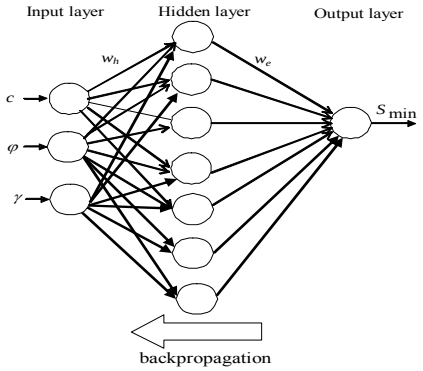


Fig.2. Architecture of the ANN model

In the learning phase, the weight connection values are iteratively adjusted, in order to minimize the mean-squared error between the ANN response and that known as the correct one from each lesson of the trained set. The back-propagation is used to adjust the weights by a gradient method, starting from the output towards the input layer.

After the learning phase, the ANN must be validated using a different set of lessons, having the known input and output values. In both phases, the ANN ability to perform a good approximation is proved by some statistical parameters. In this application, the number of ANN responses having an absolute deviation less than 0.01,…, 0.05 (about 1-5%) from the correct values is accepted. Also, for each set of 100 lessons created with GA, there were determined: absolute average deviation on the set, maximum absolute deviation, average percentage deviation (%), average absolute percentage deviation (%) and standard residual deviation.

For illustration, in table 1 the statistical parameters obtained after the training of ANN with some sets of lessons and with runs made under the same conditions are specified.

Table 1

Statistical results after training

Used set	*Number of data with deviation (from 100)*					*AvAD*	*MxAD*	*AvPD*	*AvAPD*	*SRD*
	≤0.01	≤0.02	≤0.03	≤0.04	≤0.05					
D12	60	90	96	98	98	0.011	0.091	0.089	0.661	0.017
D13	67	92	99	99	99	0.01	0.124	-0.122	0.633	0.016
D23	71	95	98	99	99	0.008	0.055	-0.155	0.553	0.011
D24	56	90	98	98	100	0.01	0.044	0.047	0.648	0.013
D32	71	95	98	99	99	0.008	0.053	-0.022	0.563	0.011
D34	57	95	98	99	100	0.009	0.05	0.129	0.598	0.012
D42	55	95	97	98	100	0.01	0.046	-0.097	0.631	0.013
D43	68	96	97	99	99	0.009	0.052	-0.173	0.586	0.011

As it can be seen in table 1, the use of sets $D\,24$, $D\,34$ and $D\,42$, resulted in responses different from the correct ones by less than 0.05, for all the 100 data sets. Because the set $D\,24$ recorded the minimum absolute deviation (0.044), in table 2 are included the same statistical parameters obtained at validation of other data sets, using the weights found with this set.

Table 2

Statistical results at validation, with given weight

Verified set	*Number of data with deviation (from 100)*					*AvAD*	*MxAD*	*AvPD*	*AvAPD*	*SRD*
	≤0.01	≤0.02	≤0.03	≤0.04	≤0.05					
D12	64	94	98	98	98	0.011	0.143	-0.009	0.674	0.023
D13	63	94	99	99	99	0.01	0.143	0.104	0.649	0.018
D14	63	94	99	99	99	0.01	0.143	0.104	0.649	0.018
D23	57	91	99	99	100	0.01	0.042	0.137	0.683	0.012
D34	61	93	99	99	100	0.009	0.044	0.127	0.589	0.012

It is easy to observe that the ANN validation using the weights found with $D24$set, maintain or even improve the performance of the other sets, comparing with the training phase. The only exception appears at maximum absolute deviation which is higher for the first 3 validation sets.

Considering the diversity of the learning sets content obtained with GA and the results from the tables 1 and 2, it can be considered that ANN responds to the imposed accuracy requirements. In the following sections, is used the ANN with the weights obtained in the learning phase by $D\,24$ set.

4. The mechanism of subset simulation (SS)

Further, only for convenience, F is considered the failure event and the corresponding probabilistic domain of the uncertain parameters (Au and Beck, [2]). Also, S is considered the safety factor minimum value (instead of $S_{\min}$ from the previous Sections).

Let F_i, $i=\overline{1,n}$ be a sequence of failure events, having the probabilistic spaces with decreasing sizes, so that $F_1 \supset F_2 \supset ... \supset F_n$ and $F_k = \bigcap_{i=1}^{k} F_i$, $k=1,...,n$.

Using the definition of conditional probability it can be written that

$$P(F_n)=P\left(\bigcap_{i=1}^{n}F_i\right)=P\left(F_n\Big|\bigcap_{i=1}^{n-1}F_i\right)\cdot P\left(\bigcap_{i=1}^{n-1}F_i\right)=P(F_n|F_{n-1})\cdot P\left(\bigcap_{i=1}^{n-1}F_i\right)= \\ =...=P(F_1)\cdot\prod_{i=1}^{n-1}P(F_{i+1}|F_i) \tag{4}$$

which expresses that the failure probability at the probabilistic subspace corresponding to F_n event is given by the product of a sequence of conditional probabilities $P(F_{i+1}|F_i)$, $i=1,...,n-1$ and of the $P(F_1)$.

If S is the quantity influenced by the uncertain parameters, then the sequence of the failure events F_i can be defined by $F_i=\{S<S_i\}$, where $S_1>S_2>...>S_n$.

The idea of SS for small failure probabilities evaluation, for example $P(F_n)\cong 10^{-4}$ is to choose $n=4$ intermediate failure events so that $P(F_1)$ and $P(F_{i+1}|F_i), i=1,2,3$ to be in the range of 10^{-1}.

While the estimation of $P(F_1)$ is possible with standard MCS (eventualy with Latin hypercube sampling method), obtaining the conditional failure probabilites from eq.(4) is a very dificult task. Generating samples of vectors with uncertain parameters which have p.d.f. restricted by their affiliation to the F_i event, is inefficient if the classical MCS is used.

But the Markov chain MCS (MCMCS) method overcomes this inconvenience, being a powerful technique for simulating samples with any given p.d.f., in particular according to the conditional p.d.f. $f(\mathbf{X}|F)$, where $\mathbf{X}$ is the vector of the uncertain quantities. A short description of the method is included in Section 5.

In the actual SS implementation, the correspondence between the intermediate threshold values S_i of the output performance (the minimum safety factor) and the intermediate failure events F_i are taken into account. These thresholds S_i are introduced only for computational reasons in SS, without any physical interpretation about the degradation process.

Because of an arbitrarily choosing of the sequence $\{S_i; i=1,2,...,n\}$, the values of the conditional probabilities $P(F_{i+1}|F_i)$ are affected and difficult to be controlled. Thus, the intermediate threshold values are chosen adaptively: the conditional failure probabilities are imposed to a fixed value p_0, and then the corresponding S_i values are obtained.

The steps of SS algorithm are as follows:

- M vectors $\left\{\mathbf{X}_0^k; k=1,2,...,\mathrm{M}\right\}$ are sampled by standard MCS or by the Latin hypercube sampling technique. These samples correspond to "conditional level 0". Using ANN, the corresponding values of the response $\left\{S\left(\mathbf{X}_0^k\right), k=1,2,...,\mathrm{M}\right\}$ are computed and ranked in an increasing list. The first intermediate threshold value S_1 is chosen as the $p_0\mathrm{M}$-th value in this list; in this way, the sample estimation of $P(F_1)=P(S<S_1)$ is equal to p_0. There are now $p_0\mathrm{M}$ samples among all M vectors $\left\{\mathbf{X}_0^k; k=1,2,...,\mathrm{M}\right\}$, whose S values lie in $F_1=\{S<S_1\}$. These are placed at "conditional level 1" and distributed as $f\left(\cdot|F_1\right)$;

- starting from each one of these samples, the MCMCS is used to generate $(1/p_0-1)$ new conditional samples – i.e. on the entire $(1/p_0-1)\mathrm{M}$ samples - to complete at M the number of conditional samples $\left\{\mathbf{X}_1^k; k=1,2,...,\mathrm{M}\right\}\in F_1$ at "conditional level 1". The intermediate threshold value S_2 is chosen as the p_0M-th value from the increasing list of the responses $\left\{S\left(\mathbf{X}_1^k\right), k=1,2,...,\mathrm{M}\right\}$, so defining the failure region $F_2=\{S<S_2\}$. The sample estimation of $P\left(F_2|F_1\right)=P\left(S<S_2|S<S_1\right)$ is also equal to p_0;

- the $p_0\mathrm{M}$ samples lying in F_2 are used as seeds for sampling $(1-p_0)\mathrm{M}$ additional conditional samples distributed as $f\left(\cdot|F_2\right)$, to complete a total of M conditional samples $\left\{\mathbf{X}_2^k; k=1,2,...,\mathrm{M}\right\}$ at "conditional level 2".

This procedure is repeated until the samples at "conditional level *n*-1" are generated to obtain the threshold value S_n as the $p_0\mathrm{M}$-th value in the increasing list of $\left\{S\left(\mathbf{X}_{n-1}^k\right), k=1,2,...,\mathrm{M}\right\}$. The failure region $F_n=\{S<S_n\}$ is then defined and the sample estimation of $P\left(F_n|F_{n-1}\right)$ is also equal to p_0.

Idea of this SS implementation is detailed described in Au and Beck, [18], Zio and Pedroni, [4], etc.

5. Markov chain Monte Carlo simulation (MCMCS)

In Fishman, [19] is presented the Metropolis method from which originates the MCMCS. Au and Beck, [2] include a modified Metropolis algorithm, more suitable when random vectors with many independent components are to be sampled.

Let a seed sample $\mathbf{X}$ in the failure region F_i. For every $j=1,2,3$ uncertain parameters (c,φ,γ) it can be considered a "proposal" p.d.f. $f_j^*(y|x)$ as a one – dimensional p.d.f. for y centered at x and having the symmetry property, i.e. $f_j^*(y|x)=f_j^*(x|y)$.

With seed sample $\mathbf{X}=\mathbf{X}_1=\left\{x_1^1,x_2^1,x_3^1\right\}$, the Markov chain $\mathbf{X}_2,...,\mathbf{X}_k,...$ is generated according to the following scheme:

1. Let $\mathbf{X}_k=\left\{x_1^k,x_2^k,x_3^k\right\}$ the current state
2. Generate a candidate state $\widetilde{\mathbf{X}}_{k+1}=\left\{\widetilde{x}_1^{k+1},\widetilde{x}_2^{k+1},\widetilde{x}_3^{k+1}\right\}$
   For each component $j=1,2,3$:
   - generate a pre-candidate y_j^{k+1} from $f_j^*\left(\bullet|x_j^k\right)$
   - compute the acceptance ratio

$$r_j=\frac{f_j\left(y_j^{k+1}\right)f_j^*\left(x_j^k\left|y_j^{k+1}\right.\right)}{f_j\left(x_j^{k+1}\right)f_j^*\left(y_j^{k+1}\left|x_j^k\right.\right)} \tag{5}$$

   - set $\widetilde{x}_j^{k+1}$ according to

$$\widetilde{x}_j^{k+1}=\begin{cases}y_j^{k+1} & \text{with probability } \min\left(1,r_j\right)\\ x_j^k & \text{with probability } 1-\min\left(1,r_j\right)\end{cases} \tag{6}$$

3. Accept / reject candidate state $\widetilde{\mathbf{X}}_j^{k+1}$:
   - if no pre-candidate components were accepted, then go to step 4
   - else, check the location of $\widetilde{\mathbf{X}}_j^{k+1}$ by computing with ANN the output performance S and

- if S is less than the threshold value S_i, then $\widetilde{\mathbf{X}}_j^{k+1} \in F_i$ and accept it as the next state (i.e. set $\mathbf{X}_{k+1} = \widetilde{\mathbf{X}}_j^{k+1}$);
- else the current pre-candidate is rejected.

4. Repeat from step 2 for an imposed number of attempts, at rejection from step 3. If no accepting condition persists after 5 attempts, set $\mathbf{X}_{k+1} = \mathbf{X}_k$.

The last operation is here proposed to decrease the number of identical samples at "conditional level *i*" and so, to provide a more accurate sample estimation of the next intermediate threshold value S_{i+1}.

Referring to the "proposal" p.d.f.s f_j^*, these affect the deviation of the candidate state from the current one. The spread of f_j^* determines the covering of failure region by Markov chain samples. Small spread increases the dependence between successive samples, while too large spread reduces the acceptance rate, increasing the number of identical samples in Markov chain. The type of the chosen proposal p.d.f. is less important and an uniform or normal p.d.f. centered at current sample may be used. In this work, the last one is adopted, with the same spreading as the original p.d.f., s f_j.

6. Numerical results

A first problem in SS algorithm implementation is choosing the number of vectors M, with values of uncertain parameters and the fixed value p_0, for the failure conditional probabilities. In addition, $p_0 M$, has to be integer and, also, $(1/p_0 - 1)$ - the number of new conditional samples generated by MCMCS from each sample maintained at any level – to be integer.

For the "conditional level 0", by the Latin hypercube sampling technique a more suitable filling of the probabilistic space is obtained, comparatively to the standard MCS. This is based upon stratified sampling, with random selection within each stratum, and 1000 samples obtained by such a method will produce comparable estimations to these of about 5000 samples generated by standard MCS.

Two files, *V1* and V2, each having $M = 1000$ different vectors (c, φ, γ) were created using this method. Then, a new file, *V*, with $M = 2000$ vectors, is obtained by concatenation of the previous two.

Referring to the p_0 value, Au and Beck, [2], empirically found that a good efficiency is obtained if p_0=0.1, but in this application such value seems to be too small (critical condition $S=1$ appears inside the conditional level 1). Some other values should be 0.2; 0.25 and 0.5, with only 4, 3 and 1 new generated samples from each seed vector in MCMCS procedure.

Because all seed samples maintained at any level *i* are in the probabilistic region F_i (and therefore distributed as $f(\mathbf{X}|F_i)$, so are the subsequent samples generated from each seed and the Markov chain is in a stationary state. This means that the length of the Markov chain obtained from each seed becomes less important. The following results were computed using p_0=0.5 and p_0=0.2, with 1 and 4 new samples in each Markov chain.

To illustrate the SS idea, figure 3.a presents the $M=1000$ points (c,φ,γ) from *V1* at conditional level 0, and then, figures 3.b, c and d present all the 500 seed states and 500 generated states at conditional levels 1, 2 and 3.

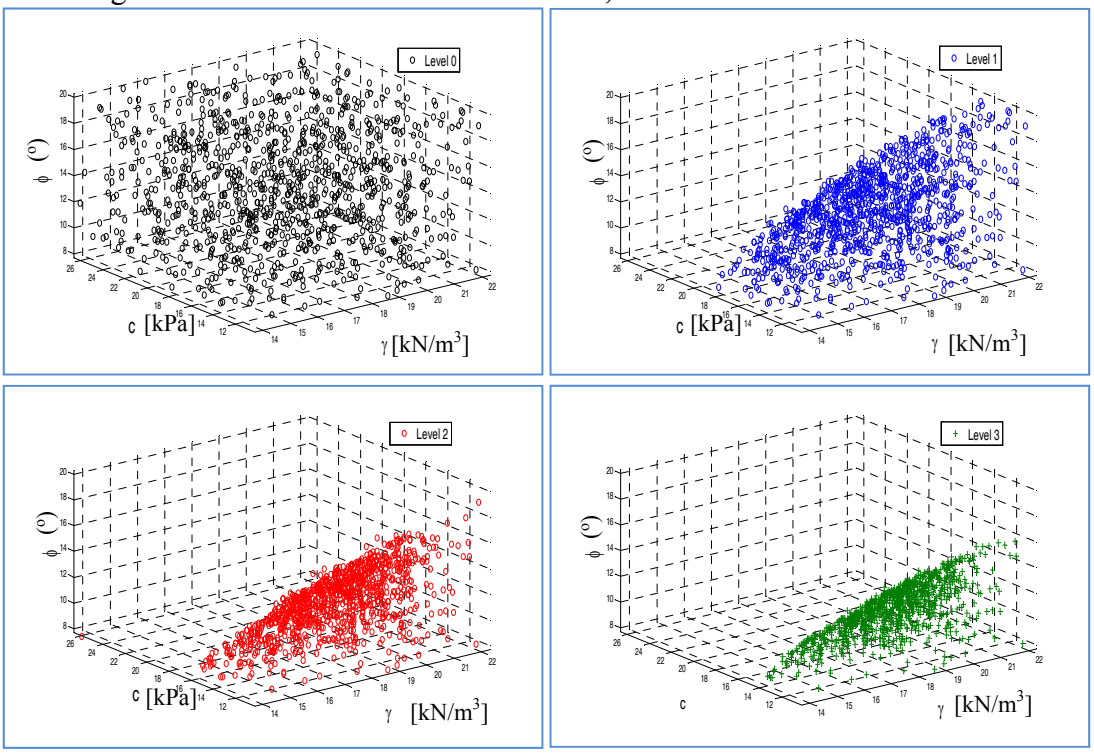


Fig.3. The points of M = 1000 sample vectors in probabilistic domain: a) level 0; b) level 1($S_{\min} \le 1.4906$); c) level 2($S_{\min} \le 1.3229$); d) level 3 ($S_{\min} \le 1.2207$)

Table 3 includes some results obtained for p_0=0.5, and $M=1000$. For each initial vector *V1* and *V2* are presented the threshold values S_i from one run, and the average values from 3 different runs.

One ascertains that threshold values S_i at different values of the non-exceeding probability are almost equals. This is true between individual runs with initial vectors *V1* and *V2*, as well for the mean values from 3 runs with *V1* and *V2* respectively.

Table 3

Threshold values S_i, so that $P(S \le S_i)$, $p_0 = 0.5$

$P(S \le S_i)$	0.5^1	0.5^2	0.5^3	0.5^4	0.5^5	0.5^6	0.5^7	0.5^8
V1	1.4906	1.3229	1.2207	1.1508	1.0993	1.0588	1.0213	0.9953
$V1_{ave}$	1.4906	1.3210	1.2196	1.1508	1.0984	1.0568	1.0217	0.9931
V2	1.4953	1.3253	1.2244	1.1423	1.0990	1.0508	1.0209	0.9824
$V2_{ave}$	1.4953	1.3268	1.2247	1.1462	1.0978	1.0521	1.0216	0.9860

Using the last two average *S* values obtained with *V1* and linear interpolation, the probability for critical state $S \le 1$ results as $P(S \le 1)$=0.00485, i.e. less than 0.5%.

A similar analysis was developed for the fixed probability p_0=0.2. In table 4 are presented the threshold values S_i, obtained in 3 runs with initial vectors *V1*, *V2* and *V* (*M*=2000 data sets) respectively.

Table 4

Threshold values S_i, so that $P(S \le S_i)$, $p_0 = 0.2$

$P(S < S_i)$	*V1*			*V2*			*V*		
0.2^1	1.2365	1.2365	1.2365	1.2473	1.2473	1.2473	1.2431	1.2431	1.2431
0.2^2	1.0949	1.0945	1.1019	1.0873	1.0899	1.1001	1.0837	1.0942	1.0888
0.2^3	1.0137	1.0078	1.0126	0.9942	1.0104	1.0017	1.0024	1.0094	1.0014
0.2^4	0.9377	0.9377	0.9527	0.9413	0.9465	0.9413	0.9365	0.9365	0.9387

By a simple inspection, the same general conclusions as above can be derived. At any non-exceeding probability level, in all runs were obtained almost the same values of the threshold S_i.

For evaluation the critical state probability $P(S \le 1)$, the last two lines from table 4 are to be used within the linear interpolation. The mean *S* values from the 3 runs with *V1* are 1.0114 and 0.9427 (at $P = 0.2^3 = 0.008$ and $P = 0.2^4 = 0.0016$ respectively), resulting $P(S \le 1) = 0.00694$, while from runs with V vectors are obtains the mean S values 1.0044 and 0.9373 (at the same P values), and $P(S \le 1) = 0.00758$ (i.e. less than 0.7% and 0.8% respectively).

The linear interpolation was used above only for convenience.

A more appropriate way to evaluate the non-exceeding probability $P(S \le 1)$ should be to obtain the rank of the first value with S=1in the increasing list at the last level having the $p_0 M$ values of S less than 1. Then, the value of the rank divided by M is exactly the conditional probability as $S \le 1$ at the last level.

With initial V vectors ($M = 2000$) and fixed probability $p_0 = 0.2$, a run that uses this scheme gives $P(S \le 1) = 0.00516$ (while using linear interpolation the result was 0.00636) and the average value from 3 runs resulted as 0.0060. With $p_0 = 0.5$ an average probability $P(S \le 1) = 0.0045$ is derived.

However, taking into account the observation about using $p_0 = 0.1$ in this application, 3 runs were finally accomplished with initial V vectors. The non-exceeding probability $P(S \le 1)$ is obtained as: 0.0135; 0.01295 and 0.0129, having an average value of 0.0131 (i.e. about 1.3%).

The small differences between the results obtained at the same value p_0 are perfectly well explained by the probabilistic nature of the analysis. Concerning the observed differences at various values of p_0, these are rather related to the characteristic feature of this particular problem, besides the probabilistic aspects.

At any rate, an estimation of the non-exceeding probability for critical state $S \le 1$ as being less than 1.5% seems to be reasonable and this value may be accepted in any reliability study.

7. Conclusions

Efficient computation of slope failure probability at critical value $S = 1$ for the minimum safety factor is here obtained using: 1) a genetic algorithm (GA) model to find this minimum safety factor when geometrical and soil data are specified; 2) an artificial neural network (ANN) able to directly give this factor, for specified values of uncertain parameters; 3) subset simulation (SS) to express the small failure

probability as a product of larger conditional failure probabilities; 4) Markov chain Monte Carlo simulation (MCMCS) to easily generate samples conditional on intermediate failure region.

By Latin hypercube sampling method, a lot of vectors with uncertain parameters were derived and then used in GA model to obtain the training and validation data sets for ANN. All the following probabilistic analyses were accomplished only by using this ANN, which ensures the efficiency of proposed methodology.

It should be noted that – in this numerical example – the chance to appear a minimum safety factor less than a value of 1.5 (standard value in design) is about 50%, but the same chance for an *effective failure* (at critical value $S \leq 1$) does not exceed about 1.5%. This aspect must be taken into account for any risk analysis and possible optimization of structure design.

R E F E R E N C E S

[1]. *A. M. Freudenthal,* Safety and the probability of structural failure; Trans. Am. Soc. Civ. Eng., 121, p. 1337-1397, 1956.

[2]. *S. K. Au., J. L. Beck,* Estimation of small failure probabilities in high dimensions by subsetsimulation; Probabilistic Engineering Mechanics, Elsevier, 16, p. 263 – 277, 2001.

[3]. *S. E. Cho,* Probabilistic stability analyses of slope using the ANN – based response surface;Computer and Geotechnics; 36, p. 787-797, 2009

[4]. *E. Zio , N. Pedroni,* Subset simulation and line sampling for advanced Monte Carlo reliability analysis; in Proc. of the ESREL Conf, p. 687 – 694, 2009.

[5]. *W. Fellenius,* Calculation of stability of earth dams; Transactions of Second Congress on Large Dams; 4; p. 445-459, 1936.

[6]. *M. L. Stein,* Large sample properties of simulations using Latin Hypercube sampling;Technometrics, 29; p. 143 – 151; 1987.

[7]. *R .Y. Rubinstein,* Simulation and the Monte Carlo method; Wiley, N.Y, 1981.

[8]. *R. E. Melchers,* Importance sampling in structural systems; Struct. Safety, 6, p. 3 – 10, 1989.

[9]. *D. V. Griffiths, P. A. Lane,* Slope stability analysis by finite elements; Geotechnique, 49(3), p. 387 – 403, 1999.

[10]. *T. Matsui, K. C. San*, Finite element slope stability analysis by shear strength reduction technique; Soil Found; 32 (1); p.59-70 , 1992.

[11]. *R. Priscu* – Constructii hidrotehnice (Hydrotechnical Structures), Editura Didactica si Pedagogica, Bucuresti, 1974.

[12]. *Z. Michalewicz,* Genetic algorithms + data structures = evolution programs, Springer-Verlag, N.Y., 1994.

[13]. *D. E. Goldberg,* Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Massachusetts, 1989.

[14]. *L. Chambers (ed.)*, Practical Handbook of Genetic Algorithms: New Frontiers, Volume II,CRC Press, Boca Raton, Florida,1995.
[15]. *K. A. De Jong,* Evolutionary computation. A unified approach; Bradford Books, N.Y., 2002.
[16]. *C. R. Houck, J. A. Joines, M. G. Kay,* A genetic algorithm for function optimization. A Matlab implementation; ACM Trans on Mathematical Software, 22, p 1 -14, 1996.
[17]. *K. Hornik, M. Stinchcombe, H. White,* Multi - layer feed – forward networks are universal approximators; Neural Networks, 2 (5), p. 395 – 368, 1989.
[18]. *S. K. Au, J. L. Beck,* Subset simulation and its application to seismic risk based on dynamic analysis; Journal of Engineering Mechanics, ASCE, 129 (8), p.901 – 917, 2003.
[19]. *G. S. Fishman,* Monte Carlo: Concepts, algorithms and applications; Springer – Verlag, N.Y., 1996.