Confiabilidade estrutural utilizando o método de Monte Carlo e redes neurais

Anderson Henrique Barbosa
UFOP - Escola de Minas - Departamento de Engenharia Civil, Mestrado em Construção Metálica, Laboratório de Mecânica Computacional
E-mail: anderhb@bol.com.br

Marcílio Sousa da Rocha Freitas
UFOP - Escola de Minas - Departamento de Engenharia Civil, CEP 35400-000, Ouro Preto, MG, Brasil
E-mail: marcílio@em.ufop.br

Francisco de Assis das Neves
UFOP - Escola de Minas - Departamento de Engenharia Civil, CEP 35400-000, Ouro Preto, MG, Brasil
E-mail: fassis@em.ufop.br

Resumo

A análise de confiabilidade estrutural em geral, por envolver um grande número de variáveis aleatórias ou exigir uma grande quantidade de simulações, se depara com a questão do custo computacional. Duas técnicas utilizadas para essa avaliação são o método de simulação de Monte Carlo e os métodos analíticos do tipo FORM/SORM. Os métodos analíticos FORM e SORM podem apresentar problema de precisão no cálculo da probabilidade de falha. Em relação ao método de Monte Carlo, embora sejam de fácil implementação e absolutamente geral, o grande número de simulações pode exigir um tempo de processamento elevado, o que pode tornar sua aplicação inviável. Nesse trabalho, foi utilizada uma rede neural treinada para substituir a solução do problema estrutural necessário a cada simulação de Monte Carlo, com o objetivo de reduzir o custo computacional requerido na análise. As aplicações realizadas proporcionaram bons resultados, com baixo custo computacional, o que atesta a viabilidade de sua aplicação.

Palavras-chave: Confiabilidade estrutural, método de Monte Carlo, redes neurais.

Abstract

Structural reliability analysis due to the great number of random variables or large number of simulations needed may result in a high computational cost. Two techniques largely used for structural reliability assess are Monte Carlo Simulation and the analytic methods FORM/SORM. These may present some inaccuracy in the assessment of the probability of failure. The Monte Carlo Method is easy to implement and absolutely general, but the great number of required simulations may result in high computational cost making the application impracticable. This work used a trained neural network to substitute the structural analysis needed for each Monte Carlo Simulation, in order to reduce the computational cost. The applications produced good results with low computational cost, certifying its application viability.

Keyswords: *Structural reliability; Monte Carlo Method; Neural Networks.*

1. Introdução

A incerteza dos parâmetros envolvidos na análise estrutural é conhecida como um fator importante que influencia a segurança estrutural (Pulido et al., 1992). A existência de incertezas nesses parâmetros contribui para que se tenha uma probabilidade não nula de que a estrutura não atenda aos objetivos para os quais ela fora concebida. Essa probabilidade é definida como probabilidade de falha e pode ser determinada pelos métodos de análise de confiabilidade estrutural.

A teoria da confiabilidade é uma ferramenta que proporciona ao engenheiro, a partir do conhecimento das incertezas inerentes às variáveis de projeto, por meio de suas distribuições de probabilidade, a determinação da probabilidade de a estrutura falhar e, também, parâmetros que fornecem a importância de cada variável nessa probabilidade. Essas informações são, seguramente, de fundamental importância na tomada de decisões que envolvam a segurança da estrutura.

A análise de confiabilidade estrutural, por envolver um grande número de variáveis aleatórias ou exigir uma grande quantidade de simulações, se depara com a questão do custo computacional. Duas técnicas utilizadas para essa avaliação são o método de simulação de Monte Carlo e os métodos analíticos do tipo FORM / SORM. Os métodos analíticos FORM e SORM apresentam alguns problemas em função da complexidade da análise, que gera dificuldades na determinação dos pontos de mínimo. Em relação ao método de Monte Carlo, embora seja de fácil implementação e absolutamente geral, o grande número de simulações pode exigir um tempo de processamento elevado, o que pode tornar sua aplicação inviável. Esse problema tem sido resolvido através de técnicas de redução de variância (Papadrakakis et al., 1996 e Papadrakakis & Lagaros, 2002).

Uma rede neural artificial é uma estrutura computacional que procura imitar o comportamento do cérebro humano em desempenhar tarefas particulares, que dependem, principalmente, da capacidade de aprender. A eficiência desta em produzir resultados satisfatórios passa pela escolha de alguns parâmetros escolhidos via testes numéricos, que são de suma importância para a sua performance e podem exercer influência no seu tempo de treinamento.

Nesse artigo, são analisados problemas de confiabilidade estrutural com a utilização de redes neurais artificiais, aplicadas em conjunto com o método de simulação de Monte Carlo (Barbosa, 2004). Para tal aplicação, será utilizado o software Matlab (Demuth & Beale, 1998), que possui um ambiente de redes neurais já implementado.

2. Método de Monte Carlo

O método de Monte Carlo é um método de amostragem artificial utilizado na solução de experimentos aleatórios onde se tem conhecimento das distribuições de probabilidade das variáveis envolvidas. Tem sido utilizado para determinar a confiabilidade de sistemas estruturais (Pulido et al., 1992).

A sua utilização requer a geração de N amostras independentes do vetor das variáveis aleatórias X obtidas a partir da função densidade de probabilidade conjunta $f_X(\mathbf{X})$. A probabilidade de falha pode ser expressa, utilizando o método de Monte Carlo, partindo da integral definida em (1):

$$P_f = \int_{G_X(\mathbf{X})\le 0} \int \ldots \int f_X(\mathbf{X}) d\mathbf{X} \tag{1}$$

ou em (2):

$$P_f = \int_{\forall \mathbf{X}} \int \ldots \int I[G_X(\mathbf{X}) \le 0] f_X(\mathbf{X}) d\mathbf{X} \tag{2}$$

onde $G_X(\mathbf{X})$ é a função de falha, que relaciona as variáveis envolvidas na análise, e a função I[.] é um indicador que corresponde aos valores apresentados em (3):

$$I[.] = \begin{cases} 1 \text{ se } G_X(\mathbf{X}) > 0 \\ 0 \text{ se } G_X(\mathbf{X}) \le 0 \end{cases} \tag{3}$$

Pode-se reescrever a expressão (2) como em (4):

$$P_f = \sum_{i=1}^{N} I\left[G_X(\mathbf{X}^i) \le 0\right] \cdot \frac{1}{N} \tag{4}$$

onde $\mathbf{X}^i$ representa a i-ésima amostra do vetor das variáveis X geradas a partir da função densidade de probabilidade $f_X(\mathbf{X})$. O valor de P_f passa a ser determinado pela expressão (5):

$$P_f = \frac{\text{Nº de Simulações em que } G_X(\mathbf{X}) \le 0}{N} \tag{5}$$

A variância, para valores pequenos da probabilidade de falha, é expressa em (6):

$$\text{Var}(P_f) = P_f \cdot \frac{(1 - P_f)}{N} \cong \frac{P_f}{N} \tag{6}$$

Para as estruturas usuais, a probabilidade de falha P_f é pequena, geralmente da ordem de 10^{-3} a 10^{-5}, e, como a sua variância é expressa de forma inversamente

proporcional ao número total de simulações, o valor de N deve ser elevado para que se possa obter aproximações aceitáveis de $\mathbf{P}_f$. Por esse motivo, o método de Monte Carlo é, freqüentemente, utilizado para checar outras técnicas aproximadas (Pulido et al., 1992).

3. Redes neurais artificiais

Com o processo evolutivo das máquinas, um grande objetivo do homem é a construção de uma máquina que possa executar tarefas sem depender do controle humano, que tenha a capacidade de aprender e de interagir com ambientes que lhe são desconhecidos, podendo ser denominada de inteligente. Ela teria uma grande habilidade de aprendizado de afazeres não facilmente manipulados pelas máquinas atuais e continuaria a se adaptar e a realizar tarefas com maior eficiência.

Redes neurais artificiais podem ser caracterizadas como modelos computacionais baseados em processamento distribuído paralelo com propriedades particulares como habilidade para aprender, generalizar, classificar e organizar dados (Gomes & Awruch, 2004).

3.1 Modelo matemático

O primeiro modelo de neurônio artificial foi desenvolvido por McCulloch e Pitts em 1943 e representava uma simplificação do que era conhecido na época sobre o funcionamento do neurônio biológico. Esse neurônio pode ser observado na Figura 1 e descrito pela seguinte formulação matemática mostrada em (7):

$$y = f\left(\sum_{i=1}^{n} W_i x_i - b\right) = f\left(\mathbf{W}^t \mathbf{x} - b\right) \quad (7)$$

onde:
x = vetor com n sinais de entrada.
W^t = matriz com os pesos de cada neurônio acoplado.
b = valor do limiar (threshold), também conhecido como bia.
f(.) = função de ativação.
y = sinal de saída.

A forma com que os neurônios de uma rede estão interligados está diretamente ligada ao algoritmo de aprendizado, que será utilizado para treinar a rede. Como exemplo, pode-se citar as redes multilayer feedforward sem realimentação, que são redes progressivas que possuem uma ou mais camadas ocultas, redes estas utilizadas nas aplicações posteriores, apresentada na Figura 2.

3.2 Características das redes neurais

A capacidade de uma rede neural depende, principalmente, da sua estrutura paralela distribuída e de sua habilidade de aprender e, como conseqüência, generalizar. Algumas das características importantes das redes neurais são:

- Tolerância a falhas, que permite que a rede continue a apresentar resultados aceitáveis, no caso de falha de algum neurônio. A informação contida na rede está distribuída por todos os seus elementos, possibilitando que, mesmo que parte da rede seja destruída, a informação esteja contida nos elementos restantes e possa ser recuperada.
- Generalização, que possibilita à rede obter saídas adequadas como resposta a dados de entrada desconhecidos, ou seja, não pertencentes ao conjunto de treinamento.

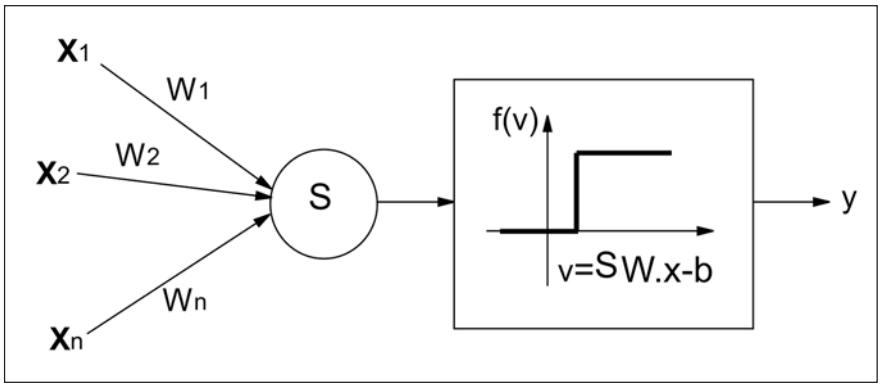

Figura 1 - Modelo matemático do neurônio.

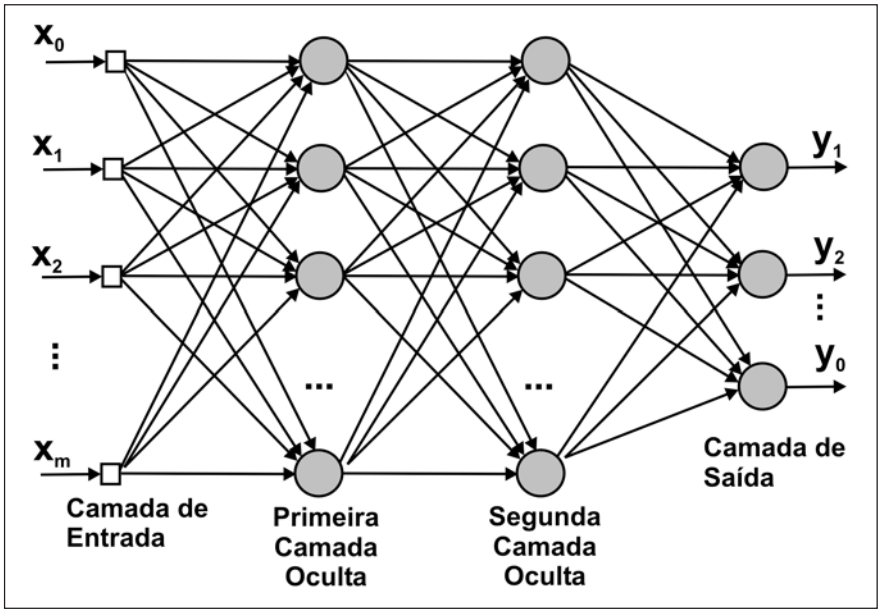

Figura 2 - Rede multilayer feedforward.

- Capacidade de aprendizagem, processo que envolve a modificação dos pesos sinápticos de uma rede através da aplicação de um conjunto de pares de treinamento, para os quais se conhece, previamente, a saída que se deseja obter. O treinamento é repetido até que a rede atinja um nível em que não haja mudanças significativas nos pesos.
- Habilidade de aproximação - Dada a capacidade de aprendizado, a rede tem a possibilidade de encontrar qualquer mapeamento entrada/saída, e, desde que os dados sejam representativos do processo do que se esteja tratando e desde que sejam adequadamente escolhidos a arquitetura de rede e o seu algoritmo de treinamento, as redes são capazes de aproximar funções contínuas de ordem qualquer.

3.3 Algoritmo Backpropagation

O algoritmo backpropagation é o principal algoritmo de treinamento de rede utilizado e é, freqüentemente, usado como o algoritmo de aprendizado em redes neurais estruturadas em camadas, isto devido à sua eficiência (Rumelhart et al., apud Hirose et al., 1991). Embora existam diversos algoritmos de treinamento de redes disponíveis, o algoritmo backpropagation tem apresentado excelentes resultados na análise e na resolução de problemas de confiabilidade estrutural (Saraiva, 1997).

O algoritmo backpropagation baseia-se no princípio do aprendizado por correção de erro, no qual o erro é retropropagado da camada de saída para as camadas intermediárias da rede neural. Pode-se dividir o algoritmo em dois passos: o direto e o reverso. No passo direto, o vetor de entrada é aplicado e seu efeito se propaga através das camadas da rede neural, produzindo uma saída, com os pesos todos fixos. No passo reverso, os pesos são reajustados de acordo com a regra de aprendizado por correção de erro. A resposta fornecida pela rede é subtraída da resposta desejada, produzindo um sinal de erro, sendo esse sinal propagado de volta através dos mesmos neurônios do passo direto, mas no sentido contrário do fluxo de sinais das conexões, daí o nome backpropagation. Os pesos são ajustados de forma que a resposta a ser produzida pela rede se aproxime da resposta esperada.

Através do conjunto de treinamento, mostrado em (8), composto por um vetor x com n entradas e um vetor de saída exato y_e:

$$\psi = \{x_i, y_{ei}\} \quad i = 1,..,n \tag{8}$$

pode-se definir o erro quadrático médio sobre este em (9):

$$E_R = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{ei})^2 \tag{9}$$

onde y_i é a saída fornecida pela rede e y_{ei} é o valor exato correspondente à saída da rede.

A expressão (9) pode ser reescrita em função dos pesos, como apresentado em (10):

$$E_R = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{W}^T.x) - y_{ei})^2 \tag{10}$$

Visando a determinar o conjunto de pesos que gere o mínimo erro para a função descrita em (10), existem vários algoritmos de otimização que são aplicados em conjunto com o algoritmo backpropagation. Destacam-se os algoritmos de Levenberg - Marquardt e o do gradiente descendente com momentum, também conhecido como regra delta generalizada ou algoritmo backpropagation tradicional, que serão utilizados nesse trabalho.

3.3.1 Algoritmo do Gradiente Descendente com Momentum

Esse algoritmo busca minimizar o erro quadrático, que é expresso em função dos pesos, de modo a se obter um conjunto de pesos otimizado que encerrará o processo de treinamento, tornando a rede apta a produzir padrões de saída aceitáveis. Por se tratar de um problema de otimização, torna-se necessária a determinação do gradiente do erro em relação aos pesos, processo este que pode ser inviabilizado devido ao tamanho do vetor de entrada, principalmente nos problemas de engenharia que, na maioria dos casos, engloba um grande número de variáveis.

A minimização do erro quadrático pode ser obtida utilizando o processo conhecido por regra delta. De acordo com essa regra, sendo **w**(k) um ponto sobre a superfície de erro, o ajuste a ser aplicado a esse ponto é expresso (11):

$$\Delta\mathbf{W}(k) = \eta\frac{\partial E_R(\mathbf{W})}{\partial\mathbf{W}} \tag{11}$$

onde η é uma constante positiva denominada taxa de aprendizado.

Efetuado o processo de ajuste, o valor atualizado do peso é descrito em (12):

$$\mathbf{W}(K+1) = \mathbf{W}(k) - \Delta\mathbf{W}(k) \tag{12}$$

É de particular interesse a determinação do parâmetro η, que é diretamente responsável pela rapidez do processo de aprendizado. O algoritmo backpropagation provê uma aproximação da trajetória de movimento sobre a superfície de erro, a qual, a cada ponto da superfície, segue a direção do ponto mais íngreme em busca do ponto de mínimo global. Quanto menor for a taxa de aprendizado, menores vão ser as correções a serem aplicadas aos pesos entre cada iteração, ocasionando um processo de convergência lento. Caso contrário, se o valor desse parâmetro for alto, pode-se obter uma aceleração no processo de convergência, mas pode-se tornar o algoritmo instável pela oscilação em torno de um ponto de mínimo local.

Uma forma simples de garantir a estabilidade e acelerar a convergência é a utilização da regra delta acrescida do fator de momento. Essa é representada em (13):

$$\Delta\mathbf{W}(k) = \beta\Delta\mathbf{W}(k-1) + \eta\frac{\partial E_R(\mathbf{W})}{\partial\mathbf{W}} \tag{13}$$

onde β é denominado constante de momento e possui a variação $0 < \beta < 1$. O efeito dessa constante é aumentar a velocidade na direção do ponto de mínimo. O que se deseja com a inserção do termo de momento é a redução no tempo de treinamento, a melhora na estabilidade do processo e, com isso, aumentar a possibilidade de encontrar o mínimo global.

3.3.2 Algoritmo de Levenberg - Marquardt

Esse algoritmo é considerado o método mais rápido para treinamento de redes *feedforward backpropagation*, que possui uma quantidade moderada de pesos sinápticos. Ele se baseia, para a aceleração do treinamento, na determinação das derivadas de segunda ordem do erro quadrático em relação aos pesos, diferindo do algoritmo backpropagation tradicional que considera as derivadas de primeira ordem.

O algoritmo de Levenberg - Marquardt se baseia no método de otimização de Newton, que faz uso da matriz Hessiana H . No método de Levenberg - Marquardt se faz uma aproximação para essa matriz, mostrada em (14), determinada em função da matriz Jacobiana, que contém as primeiras derivadas dos pesos em função dos pesos sinápticos, expressa em (15):

$$\mathbf{H} = \frac{\partial^2 E_R(\mathbf{W})}{\partial \mathbf{W}^2} \tag{14}$$

$$\mathbf{J} = \frac{\partial e(\mathbf{W})}{\partial \mathbf{W}} \tag{15}$$

onde e(**W**) é definido conforme a expressão (16):

$$e(\mathbf{W}) = \sum_{i=1}^{n} (y_i - y_{ei}) \tag{16}$$

A determinação da matriz Jacobiana é muito mais simples que a determinação da matriz Hessiana. Como, para uma rede neural, a performance de treinamento é expressa em função da soma dos erros quadráticos, a matriz Hessiana pode ser expressa pela expressão (17):

$$\mathbf{H} = \mathbf{J}^T(\mathbf{W}) \cdot \mathbf{J}(\mathbf{W}) \tag{17}$$

O método de Newton atualiza os pesos segundo (18):

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \mathbf{H}^{-1}.g_k \tag{18}$$

onde g_k pode ser escrito conforme (19):

$$g_k = 2\mathbf{J}^T(\mathbf{W}).e(\mathbf{W}) \tag{19}$$

O algoritmo de Levenberg - Marquardt procede a atualização dos pesos baseado na mesma expressão do método de Newton (18), realizando as modificações para a determinação da matriz Hessiana, mostrada em (20):

$$\mathbf{W}(k+1) = \mathbf{W}(k) - [\mathbf{J}^T(\mathbf{W}).\mathbf{J}(\mathbf{W}) + \mu_k \mathbf{I}]^{-1}.\mathbf{J}^T(\mathbf{W}).e(\mathbf{W}) \tag{20}$$

onde: I é a matriz identidade e μ_K é a constante do método de Levenberg - Marquardt.

O parâmetro μ_K funciona como um fator de estabilização do treinamento, ajustando a aproximação de forma a utilizar a rápida convergência do método de Newton e evitando passos muito grandes que possam levar a um erro de convergência.

Esse método apresenta convergência em menos iterações, mas requer mais cálculos por iteração devido ao cálculo de matrizes inversas. Apesar do grande esforço computacional, ele segue sendo o algoritmo de treinamento mais rápido para redes neurais, quando se trabalha com um número moderado de parâmetros na rede. Se esse número é elevado, a utilização desse algoritmo é pouco prática.

4. Aplicações

As redes avaliadas nas aplicações que seguem são todas do tipo Multilayer Perceptron. Ao longo das aplicações, tais redes receberão denominações com duas terminações, LM e GDM, que denotarão a utilização dos algoritmos de Levenberg - Marquardt e do gradiente descendente com momentum, respectivamente, posto para facilitar a distinção entre os dois métodos.

Considerou-se como critério de parada para o treinamento das redes avaliadas nessas aplicações o número máximo de épocas igual a 1000, ou o erro esperado de 1e-10, e a matriz de pesos das redes neurais foi gerada de forma aleatória pelo programa Matlab.

4.1 Aplicação 1

Essa aplicação se refere a uma treliça isostática representada na Figura 3, onde são consideradas como variáveis básicas a carga aplicada P e a resistência R, cujas características estatísticas estão apresentadas na Tabela 1. Esse é um problema bastante citado na literatura, tendo sido analisado por diversos pesquisadores.

Nesse problema, é considerado o efeito da carga P contra a resistência R, sendo a função de falha definida em (21):

$$G(\mathbf{X}) = R - P \tag{21}$$

A probabilidade de falha para esse caso pode ser calculada, para uma barra qualquer da treliça, utilizando-se (22):

$$P_f(\text{Barra i}) = P[(R - P) \le 0] = \int_{-\infty}^{\infty} F_R(\mathbf{X}).f_P(\mathbf{X}).d\mathbf{X} \tag{22}$$

onde:

$F_R(X)$ = Função de distribuição acumulada da variável R.

$f_p(X)$ = Função densidade de probabilidade da variável P.

Tendo essas variáveis distribuições normais, a integral definida em (22) pode ser calculada através de (23):

$$P_f(\text{Barra i}) = \Phi\,(-(\mu_R - \mu_{Pi})/(S^2_R + S^2_{Pi})^{1/2}) \quad (23)$$

onde:

μ_R, μPi são as médias das variáveis R e P na barra i;.

σ_R, σ_{Pi} são os desvios padrões de R e P na barra i.

Φ é a função de distribuição acumulada normal padrão.

Fazendo uso de (23), chega-se à probabilidade de falha para as barras 1 e 2 $P_f = 0.039848$ e para a barra 3 $P_f = 3.24e\text{-}6$.

Os resultados deste trabalho são comparados com os apresentados por Saraiva (1997), cuja rede neural, denominada de MCRN1, possui as seguintes características:

- Rede *Feedforward* sem realimentação com técnica *Backpropagation*.
- Número de camadas intermediárias = 1.
- Número de neurônios na camada intermediária = 5.
- Taxa de aprendizagem $\eta = 0.1$.
- Fator de "Momentum" $\beta_m = 0.1$.

Além dessa rede, Saraiva (1997) utilizou os métodos de Monte Carlo Clássico e o método de Monte Carlo com a técnica de redução de variância Variáveis Antitéticas, denominado de MCRV, que possibilitarão uma comparação mais precisa dos valores fornecidos pelas redes estudadas.

Para a utilização das redes, aplicou-se a técnica de redução de variância esperança condicionada, na qual se escolhe a variável com maior dispersão e a explicita em função das outras variáveis. O conjunto de treinamento utilizou 100 pares, que consistem em valores da resistência R, para a entrada, e a saída fornece valores da função de distribuição acumulada da variável R.

As redes neurais configuradas nessa aplicação, denominadas de RNTRELM e RNTREGDM, possuem os parâmetros listados na Tabela 2, sendo todas elas do tipo Feedforward Backpropagation.

As redes RNTRELM1 e RNTREGDM1 foram utilizadas, para a determinação de P_f das barras 1 e 2, e RNTRELM3 e RNTREGDM3, para a determinação de P_f na barra 3 da treliça.

Os resultados da análise da probabilidade de falha da barra 1 estão mostrados na Tabela 3, que mostra os valores obtidos para 100, 300, 500, 1000 e 3000 simulações.

O gráfico da Figura 4 ilustra o erro gerado pelos métodos apresentados na Tabela 3. Comprovam-se os bons resultados fornecidos pelas redes neurais RNTRELM1 e RNTREGDM1, com erros abaixo de 1% para o caso de 3000 simulações, e apresentando erros pequenos para um número menor de simulações.

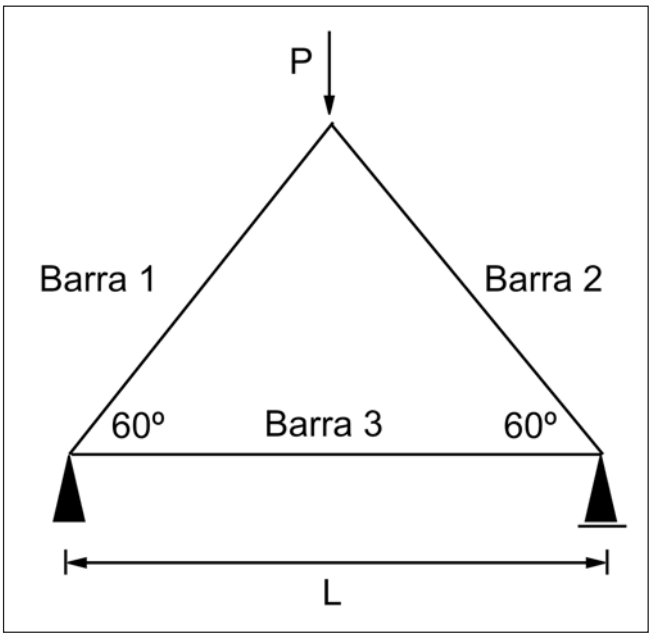

Figura 3 - Treliça isostática.

Tabela 1 - Características estatísticas das variáveis.

Variável	Distribuição	μ	σ
Carga (P)	Normal	14	1.25
Resistência (R)	Normal	11	1.50

Tabela 2 - Configuração das redes utilizadas.

	Nci	Nnc	η	β_m	μ_k	F. transferência
RNTRELM1	1	5	-	-	0.01	Tangente – Linear
RNTRELM3	1	5	-	-	0.01	Tangente – Linear
RNTREGDM1	1	8	0.1	0.1	-	Tangente – Linear
RNTREGDM3	1	8	0.1	0.1	-	Tangente – Linear

A Tabela 4 mostra os valores da probabilidade de falha obtidos utilizando as redes RNTRELM e RNTREGDM para a barra 3 da treliça isostática, cujo valor exato é $P_{f3} = 3.24e\text{-}6$.

O gráfico apresentado na Figura 5 mostra a variação do erro para a determinação, pela rede, da probabilidade de falha para a barra 3 da treliça isostática.

Os valores mostrados na Tabela 4 e na Figura 5 indicam erros de 1.88% e 5.14% em relação à resposta exata para o problema, o que revela a boa performance dos dois algoritmos para a aproximação da resposta. Os resultados apresentados nas Tabelas 3 e 4 mostram que as redes fornecem valores da probabilidade de falha com erros aceitáveis, atestando a viabilidade de sua aplicação. Essa característica se deve ao bom treinamento e à escolha correta dos parâmetros da rede neural.

4.2 Aplicação 2

Seja a viga biapoiada representada na Figura 6, onde são considerados, como variáveis aleatórias básicas, a carga concentrada aplicada P e o módulo de elasticidade do material E.

Os dados referentes às variáveis aleatórias estão mostrados na Tabela 5. O critério de falha está relacionado com

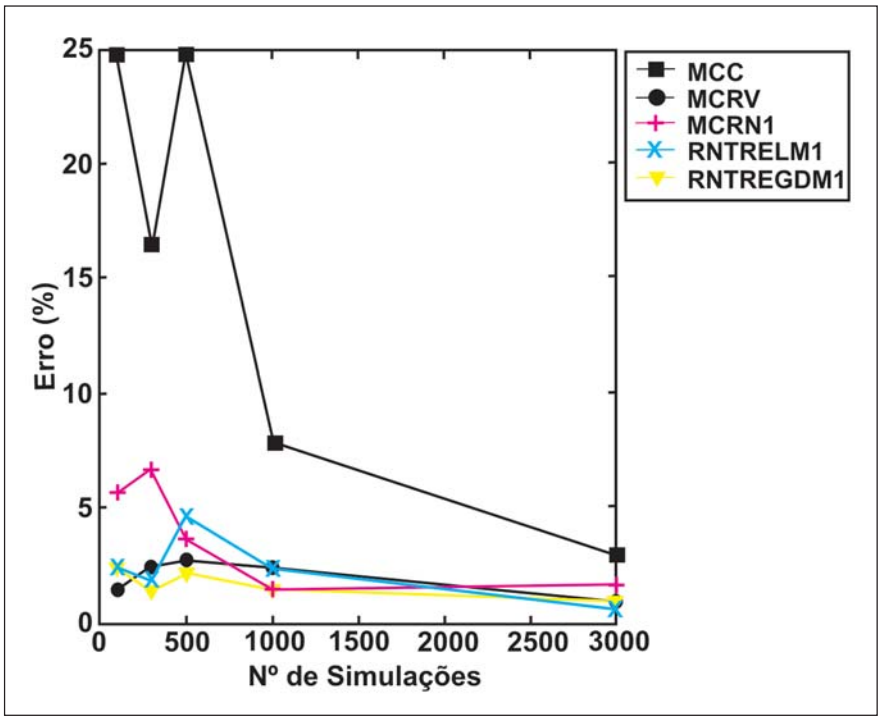

Figura 4 - Comparação dos erros de P_f das barras 1 e 2.

Tabela 3 - Valores de P_f para as barras 1 e 2.

	Número de Simulações				
	100	**300**	**500**	**1000**	**3000**
MCC	0.0300	0.0333	0.0300	0.0430	0.0410
MCRV	0.0404	0.0408	0.0409	0.0408	0.0402
MCRN1	0.0376	0.0425	0.0413	0.0404	0.0392
RNTRELM1	0.0389	0.0391	0.0380	0.0389	0.0396
RNTREGDM1	0.0408	0.0393	0.0407	0.0404	0.0402

Tabela 4 - Valores de P_f para a barra 3.

	Nº de Simulações				
	100	**300**	**500**	**1000**	**3000**
RNTRELM3	3.1091e-6	3.2950e-6	3.3013e-6	3.1456e-6	3.1791e-6
RNTREGDM3	2.9551e-6	3.0943e-6	3.0644e-6	3.3064e-6	3.0736e-6

o deslocamento do ponto A, segundo o eixo y, tomando-se como valor-limite d = 4,8 cm, sendo a função de falha expressa em (24):

$$G(\mathbf{X}) - 0.048 - d(\mathbf{X}) \qquad (24)$$

onde d(X) é o deslocamento vertical do ponto A.

Novamente, os resultados serão comparados com os apresentados em Saraiva (1997), cujas redes possuem as seguintes características:

- Rede neural do tipo Feedforward Backpropagation.
- Número de camadas ocultas = 1.
- Número de neurônios na camada oculta = 6.
- Taxa de aprendizagem $\eta = 0.1$.
- Fator de "Momentum" $\beta_m = 0.1$.
- Funções de transferência = Sigmoidal.

Essas redes foram denominadas de MCRN2 e MCRN2*, ambas possuindo as mesmas características anteriormente citadas, e tem como principal diferencial a forma de calcular a probabilidade de falha: MCRN2 utiliza a técnica de redução de variância Amostragem por Importância, enquanto MCRN2* gera os valores dos deslocamentos e calcula a P_f através do método de Monte Carlo Clássico. Determinaram-se, também, os valores de Pf para o método de Monte Carlo Clássico sem redes neurais, denominado de MCC.

O conjunto de treinamento, para o referido exemplo, consta de 80 pares. Na Tabela 6, estão apresentadas as configurações das redes neurais analisadas nessa aplicação.

Utilizaram-se rotinas auxiliares para a determinação da probabilidade de falha, visto que a rede é treinada para fornecer apenas os deslocamentos do ponto A.

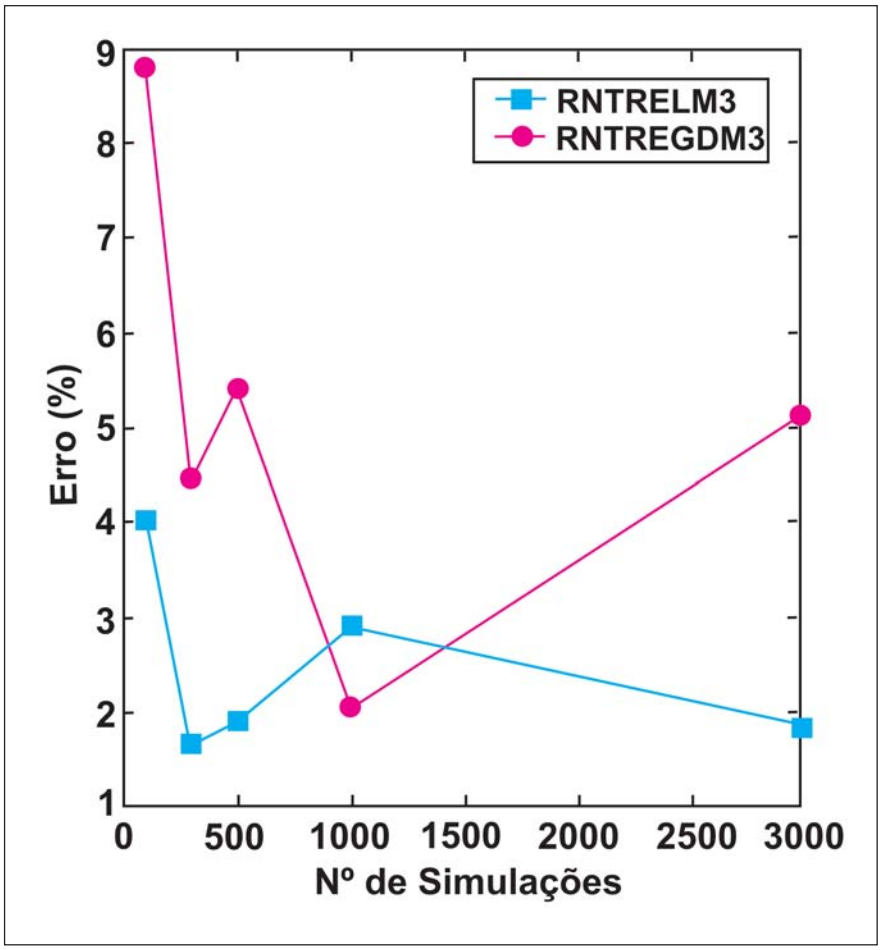

Figura 5 - Comparação dos erros de P_f da barra 3.

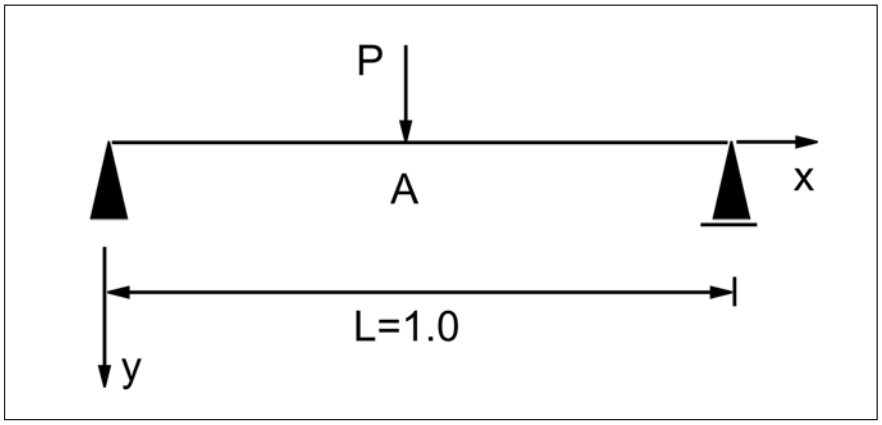

Figura 6 - Viga isostática com carga concentrada.

Tabela 5 - Características das variáveis aleatórias.

Variável	Distribuição	μ	σ	COV	λ	ξ
$P(X_1)$	Lognormal	0.16667	$1.67x10^{-2}$	0.1	-17.967	0.0998
$E(X_2)$	Lognormal	2400	240	0.1	77.782	0.0998

Tabela 6 - Configuração das redes neurais.

	Nci	Nnc	η	β_m	μ_k	F. transferência
RNVCCLM	1	3	-	-	0.1	Tangente – Linear
RNVCCGDM	1	6	0.1	0.1	-	Tangente – Linear

Tabela 7 - Valores de P_f para a viga.

	Número de Simulações				
	500	**1000**	**3000**	**5000**	**10000**
MCRN2	0.0200	0.0151	0.0126	0.0108	0.0101
MCRN2*	0.0040	0.0040	0.0143	0.0110	0.0106
MCC	0.0040	0.0050	0.0090	0.0098	0.0092
RNVCCGDM	0.0080	0.0100	0.0107	0.0108	0.0106
RNVCCLM	0.0120	0.0100	0.0090	0.0102	0.0095

A Tabela 7 apresenta a comparação dos resultados obtidos para a referida aplicação. A utilização do Método de Monte Carlo Clássico visa a apresentar uma estimativa do que seria o valor exato de P_f para servir de comparação para os resultados obtidos pelas redes estudadas.

Dos resultados apresentados na Tabela 7, verifica-se que a rede analisada foi capaz de produzir valores da probabilidade de falha com aproximações razoáveis, acompanhando a tendência do Método de Monte Carlo. Nota-se, ainda, que a rede treinada com o algoritmo de Levenberg - Marquardt apresentou bons resultados para 10000 simulações, quando comparado com o Método de Monte Carlo Clássico. Já a rede treinada com o algoritmo do gradiente descendente leva a resultados comparáveis aos obtidos por Saraiva (1997).

5. Conclusões

Os resultados apresentados nesse trabalho mostram que a aplicação das redes neurais, para a determinação da probabilidade de falha, quando da substituição de partes do Método de Monte Carlo, produz bons resultados, apresentando convergência para um número de simulações relativamente baixo. O fato de os métodos de análise de confiabilidade apresentarem algumas restrições, no processo de solução, ou exigirem um tempo computacional demasiado, para a sua convergência, torna os resultados obtidos da solução com as redes neurais uma boa alternativa.

A aplicação das redes neurais se revelou um método passível de ser utilizado, em conjunto com o Método de Monte Carlo, proporcionou ótimos resultados e tendo, como conseqüência, um baixo esforço computacional, fato que era objetivado nesse trabalho.

6. Agradecimentos

Os autores agradecem à USIMINAS.

7. Referências bibliográficas

BARBOSA, A. H. ***Análise de confiabilidade estrutural utilizando o Método de Monte Carlo e redes neurais***. Ouro Preto: Programa de Pós-Graduação em Engenharia Civil, UFOP, 2004. (Dissertação de Mestrado).

DEMUTH, H., BEALE, M. ***Neural networks toolbox user's guide - MATLAB***. Versão 3.0. 1998.

GOMES, H. M., AWRUCH, A. M.. Comparison of response surface and neural network with others methods for structural reliability analysis. ***Structural safety***, v. 26, p. 49-67, 2004.

HIROSE, Y., YAMASHITA, K., HIJIYA, S. Back-Propagation algorithm which varies the number of hidden units. ***Neural Networks***, v. 4, p. 61-66, 1991.

PAPADRAKAKIS, M., PAPADOPOULOS, V., LAGAROS, N. D. Structural reliability analysis of elasti-plastic structures using neural networks and Monte carlo simulation. ***Computer Methods in Applied Mechanics Engineering***, v. 136, p. 145-163, 1996.

PAPADRAKAKIS, M., LAGAROS, N. D. Reliability-based structural optimization using neural networks and Monte carlo simulation. ***Computer Methods in Applied Mechanics Engineering***, v. 191, p. 3491-3507, 2002.

PULIDO, J. E., JACOBS, T. L., PRATES DE LIMA, E. C. Structural reliability using Monte carlo simulation with variance reduction techniques on elastic-plastic structures. ***Computer and Structures***, p. 419-430, 1992.

SARAIVA, J. M. F. ***A utilização de redes neurais em conjunto com o Método de Monte Carlo na análise da confiabilidade de estruturas***, Rio de Janeiro: Programa de Pós-Graduação em Engenharia Civil, COPPE/UFRJ, 1997. (Tese de Doutorado).

Artigo recebido em 10/12/2004 e aprovado em 28/07/2005.