Smart Mater. Struct. 4 (1995) 126–133. Printed in the UK

# Neural network representation of fatigue damage dynamics

**Chen-Jung Li and Asok Ray†**
Mechanical Engineering Department, The Pennsylvania State University, University Park, PA 16802, USA

Received 12 August 1994, accepted for publication 6 December 1994

**Abstract.** Recently, a model of fatigue damage dynamics has been reported, which allows the damage information on critical plant components to be integrated with the plant dynamics for both on-line life prediction and off-line control synthesis. This paper proposes a neural network implementation of the fatigue damage model in order to alleviate the problem of slow computation via conventional numerical methods. The results of simulation experiments reveal that a neural network algorithm could be used as an intelligent instrument for on-line monitoring of fatigue damage and also as a tool for failure prognostics and service life prediction.

## 1. Introduction

A major objective in control of complex mechanical systems such as electric power plants, advanced aircraft, and spacecraft is to achieve the mission objectives with increased reliability, availability, component durability, and maintainability. Therefore, performance, service life, and cost of maintenance and operation should be taken into consideration during the design process. In view of high performance requirements and availability of improved materials that may have significantly different damage characteristics relative to conventional materials, the lack of appropriate knowledge about the properties of these materials will lead to either of the following: (i) less than achievable performance due to overly conservative design; or (ii) over-straining of the mechanical structures leading to unexpected failures and drastic reduction of the service life.

Ray *et al* (1994a, b) have proposed a concept for damage-mitigating control of complex structures, in which the dynamics of fatigue damage are modeled in continuous time in contrast to the usual notion of expressing the fatigue damage rate relative to the number of cycles. A unique advantage of this damage model in the continuous-time setting is that it can be directly incorporated within the control system structure to provide the necessary information for on-line damage-mitigating control as well as for off-line synthesis of a control law (Ray *et al*, 1994c). However, for on-line damage-mitigating control, traditional numerical computation of cumulative damage at several critical points may not satisfy the stringent requirements of data processing time because the damage model involves many complex non-linear calculations. This problem can be circumvented by neural computation (Troudet and Merrill, 1990).

This paper demonstrates the feasibility of the artificial neural network concept for real-time implementation of the continuous-time fatigue damage model. While the feedforward layered neural network is multiplexed for on-line calculations of fatigue damage at several critical points of the mechanical structure, the recursive error back-propagation concept is used to train this network. This class of neural networks has the capacity of representing any function to a desired degree of accuracy by a sufficiently large number of neurons (Hornik *et al*, 1989). The paper is organized in five sections including the Introduction. Section 2 reviews the continuous-time fatigue damage model based on the work of Ray and Wu (1994a). Section 3 describes the back-propagation neural network where the momentum method and the adaptive learning rate update rule are used to accelerate the convergence rate of the network during its training phase. Section 4 presents implementation of the neural network representation of the fatigue damage model and reports the results of simulation experiments. Finally, the paper is summarized and concluded in section 5 along with recommendations for future work.

## 2. The continuous-time fatigue damage model

The fatigue damage model in the continuous-time setting, reported by Ray and Wu (1994a), is first derived based on linear damage accumulation using a combination of Coffin–Manson and Basquin relationships (Bannantine *et al*, 1990). Then this linear damage model is modified following the damage curve approach (Bolotin, 1989) to account for dependence of the damage rate on the current level of accumulated damage. Only

† Corresponding author.

0964-1726/95/020126 + 08 $19.50 

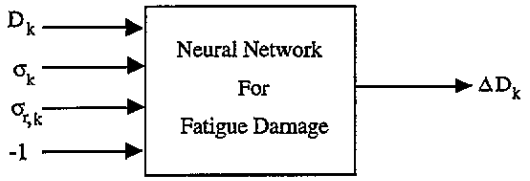


**Figure 1.** Representation of fatigue damage dynamics.

the essential features of this fatigue damage model are presented in this section.

Converting the strain amplitudes into stress amplitudes from the cyclic stress-strain curve, the rates of both elastic damage $\delta_e$ and plastic damage $\delta_p$ are computed through differentiation as:

If $\sigma \geq \sigma_r$, then

$$\frac{d\delta_e}{dt} = 2\frac{d}{d\sigma}\left(\left(\frac{\sigma-\sigma_r}{2(\sigma_f'-\sigma_m)}\right)^{-\frac{1}{b}}\right)\frac{d\sigma}{dt}$$

$$\times \text{ and } \frac{d\delta_p}{dt} = 2\frac{d}{d\sigma}\left(\frac{1}{\epsilon_f'}\left(\frac{\sigma-\sigma_r}{2K'}\right)^{\frac{1}{n'}}\left(1-\frac{\sigma_m}{\sigma_f'}\right)^{-\frac{c}{b}}\right)^{-\frac{1}{c}}\frac{d\sigma}{dt} \quad (1)$$

otherwise,

$$d\delta_e/dt = 0 \quad \text{and} \quad d\delta_p/dt = 0$$

where the current stress $\sigma$ and the stress rate $d\sigma/dt$ are obtained from the structural model; $\sigma_r$ is the reference stress obtained using the rainflow method (Dowling, 1983; Rychlik, 1993); $\sigma_m = (\sigma+\sigma_r)/2$ is the mean stress; and $\sigma_f'$, $\epsilon_f'$, $b$, $c$, $K'$, $n'$ are the material parameters (Bannantine *et al*, 1990) under cyclic operations. The damage rate $d\delta/dt$ is obtained as the weighted average of the elastic and plastic damage rates such that

$$\frac{d\delta}{dt} = w\frac{d\delta_e}{dt} + (1-w)\frac{d\delta_p}{dt} \quad (2)$$

where the weighting function, $w$, is selected as the ratio of the elastic strain amplitude and total strain amplitude. Equations (1) and (2) are then used to obtain the damage rate at any instant. Since mechanical structures are generally subjected to loads of varying amplitude, equation (1), which is based on the linear rule of damage accumulation, will lead to erroneous results due to the sequence effect (Suresh, 1991). Therefore, the linear damage is modified via a non-linear damage rule as follows:

$$D = (\delta)^{\gamma(\sigma_a, D)} \quad (3)$$

where $D$ and $\delta$ are the current states of non-linear and linear damage accumulation, respectively, and $\sigma_a$ is the stress amplitude. Equation (3) generates the following incremental change:

$$\Delta D = \gamma D^{\frac{\gamma-1}{\gamma}}\Delta\delta + \frac{D(\ln D)}{\gamma}\Delta\gamma. \quad (4)$$

It follows from a crack propagation model such as the Paris model (Paris and Erdogan, 1963) that the crack growth rate is dependent not only on the stress amplitude but also on the current crack length. Since the characteristics of $\gamma$ in (4) may strongly depend on the type of the material, availability of pertinent experimental data for the correct material is essential for damage-mitigating control. An approach to evaluate $\gamma$ at selected discrete levels of stress amplitude by interpolation based on the experimental data of Swain *et al* (1990) for the material AISI 4340 steel is given in Ray and Wu (1994a, b).

Figure 1 shows a neural network representation of the fatigue damage model having four inputs and one output. It follows from (1) to (4) that the damage increment $\Delta D_k$ at the time instant $t_k$ depends on the corresponding damage accumulation $D_k$, the current stress $\sigma_k$, and the reference stress $\sigma_{r,k}$. The input $-1$, used as the bias or threshold term (Haykin, 1994), provides an additional degree of freedom such that the process of training the neural network is expedited. The training data set is generated from the solution of the governing equations (1) to (4) of fatigue damage.

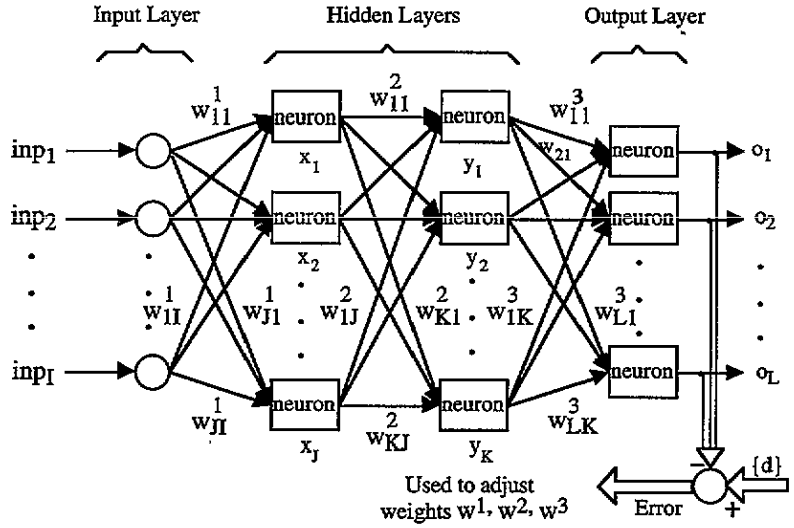


**Figure 2.** Structure of the error back-propagation neural network.

### 3. Neural network training via recursive back-propagation

The proposed neural network is fed with sets of input and desired output patterns during the iterative process of training. Random weights are provided to initialize the network, and learning is accomplished by successively adjusting these weights based on the set of input patterns and the corresponding set of desired output patterns which, in this paper, are generated from the fatigue damage model described in section 2. Let an $I$-dimensional input pattern $\{inp_i : i = 1, 2, \ldots, I\}$ be presented to the network and propagated forward through the layers of neurons of the network, and the resulting output signal be an $L$-dimensional pattern $\{o_l : l = 1, 2, \ldots, L\}$. Note that $I = 4$ and $L = 1$ in the network shown in figure 1. However, this section presents a general procedure for neural network implementation of fatigue damage, which is not restricted to the single output configuration.

Figure 2 exhibits the structure of the proposed three-layer feedforward neural network which consists of two hidden layers and the output layer in addition to the input layer. The error between the actual output $o_l$ of the neural network and the desired output $d_l$ is back-propagated through the network as a feedback signal to adjust the weights, $\{w_{ji}^1\}$, $\{w_{kj}^2\}$, and $\{w_{lk}^3\}$ of the network shown in figure 2. Each neuron forwards the weighted sum of the previous layer's outputs into an activation function which is represented by sigmoid functions at the two hidden layers and the output layer as follows:

$$x_j = f(\theta_j^1) = \left[1 + \exp(\theta_j^1)\right]^{-1} \quad \theta_j^1 = \sum_{i=1}^{I} w_{ji}^1 inp_i \quad (5a)$$

$$y_k = f(\theta_k^2) = \left[1 + \exp(\theta_k^2)\right]^{-1} \quad \theta_k^2 = \sum_{j=1}^{J} w_{kj}^2 x_j \quad (5b)$$

$$o_l = f(\theta_l^3) = \left[1 + \exp(\theta_l^3)\right]^{-1} \quad \theta_l^3 = \sum_{k=1}^{K} w_{lk}^3 y_k \quad (5c)$$

where the weights $\{w_{ji}^1\}$, $\{w_{kj}^2\}$, and $\{w_{lk}^3\}$ from the unit $i$ of the input layer to the unit $l$ of the output layer are to be optimized during the training phase of the neural network. The rules of updating the weights for a pattern pair are derived by minimizing the squared Euclidean norm of the error function at the $n$th iteration via the steepest descent method (Rumelhart and McClelland, 1986). The squared norm $E(n)$ is defined as

$$E(n) = \tfrac{1}{2}\left(\sum_{l=1}^{L}(d_l(n) - o_l(n))^2\right) = \tfrac{1}{2}\|d(n) - o(n)\|^2 \quad (6)$$

where $(d_l(n) - o_l(n))$ is the error signal for unit $l$ of the output layer at the $n$th iteration. The weights at the first and second hidden layers and the output layer in figure 2 are updated as:

$$w_{lk}^3(n+1) = w_{lk}^3(n) + \Delta w_{lk}^3(n+1) = w_{lk}^3(n) - \eta(n)\frac{\partial E(n)}{\partial w_{lk}^3(n)} \quad (7a)$$

for the output layer,

$$w_{kj}^2(n+1) = w_{kj}^2(n) + \Delta w_{kj}^2(n+1) = w_{kj}^2(n) - \eta(n)\frac{\partial E(n)}{\partial w_{kj}^2(n)} \quad (7b)$$

for the second hidden layer, and

$$w_{ji}^1(n+1) = w_{ji}^1(n) + \Delta w_{ji}^1(n+1) = w_{kj}^1(n) - \eta(n)\frac{\partial E(n)}{\partial w_{kj}^1(n)} \quad (7c)$$

for the first hidden layer, where $\eta$ is the learning rate parameter of the back-propagation algorithm, which is selected by the designer. The back-propagation algorithm for training the neural network starts at the output layer by passing the error signals backward through the network, layer by layer, and recursively computing the local gradient for each neuron (Haykin, 1994). This iterative learning process is continued until the network output signals satisfy the criterion of $E \leq \xi$, where $\xi$ is the specified tolerance. A typical value of $\xi$ is 0.0012 in this application.

Since the error surfaces possess properties that might slow down convergence of the steepest descent procedure (Jacobs, 1988), a variety of techniques have been reported in literature to accelerate the convergence rate of network training. Four heuristics were proposed by Jacobs that provide guidelines to achieve rates of convergence faster than those obtained by steepest descent techniques. A brief discussion follows.

First, every weight should have its own individual learning rate. The rationale is that the step size for adjusting any one weight is not necessarily appropriate for adjusting other weights. Second, every learning rate should be allowed to vary over time. It is common for error surfaces to possess different properties in different regions. In order to take an appropriate step size, the learning rate needs to be varied. Third, when the partial derivative of the error function with respect to one weight has the same sign for several consecutive time steps, the learning rate for that weight should be increased. As the sign of the derivative behaves in this manner, the error surface at the current point is likely to have a small curvature, and therefore, the slope continues in the same direction for a significant distance. By increasing the learning rate for this weight, the number of time steps required to traverse this distance can be reduced. Fourth, when the sign of the partial derivative of the error function with respect to one weight alternates for several consecutive time steps, the learning rate for that weight should be decreased. When the sign of the derivative behaves in this manner,

the error surface at the current point is likely to have a large curvature, and therefore, the slope of this area of the error surface may quickly change sign. In order to prevent oscillations of the weight, the corresponding learning rate should be decreased.

Several possible implementations of the heuristics described above have been reported (Jacobs, 1988;) Vogl *et al*, 1988). The momentum method and the learning rate update rule have been adopted in this paper.

### 3.1. The momentum method

The momentum method implements the heuristics by adding a new term to the weight update equations (7). At the $n$th iteration, each weight $w(n)$ of the network, at the two hidden layers and the output layer, is updated according to the following rule:

$$\Delta w(n) = -(1-\alpha)\eta\frac{\partial E(n)}{\partial w(n)} + \alpha\Delta w(n-1)$$

$$\text{with} \quad \Delta w(n-k) = 0 \quad \text{for } k \geq n \tag{8}$$

By iteration, (8) is modified to (9) as follows:

$$\Delta w(n) = -(1-\alpha)\eta(n)\sum_{k=0}^{n-1}(\alpha)^k\frac{\partial E(n-k)}{\partial w(n-k)} \tag{9}$$

where $\alpha \in [0,1)$ is the momentum factor that determines the contribution of the current and past partial derivatives relative to the current weight change. This is achieved by the exponentially weighted sum of the weight's current and past partial derivatives in which $\alpha$ is the base and the time from the current step is the exponent. The parameter $\alpha$ is to be specified by the user and a typical value of $\alpha$ is 0.9.

The momentum is considered to be an implementation of the heuristics for the following reasons. If the consecutive derivatives of a weight are of the same sign, the exponentially weighted sum grows large in magnitude and the weight is adjusted accordingly. Similarly, if consecutive derivatives of a weight are of opposite sign, then this sum becomes small in magnitude and the weight is adjusted by a small amount.

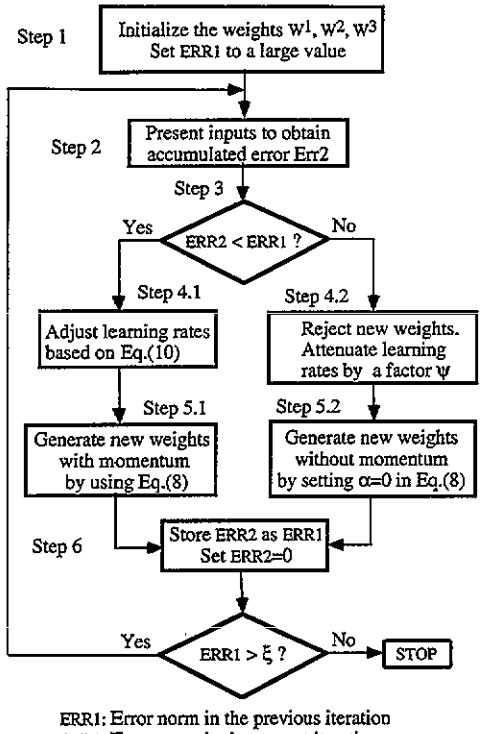


**Figure 3.** Flowchart of the modified back-propagation algorithm.

### 3.2. Modified back-propagation algorithm

Jacobs (1988) derived a learning rate update rule that performs steepest descent on an error surface defined over the learning rate parameter space. In this approach, the learning rate is updated at each iteration by minimizing the norm, $E(n)$, of the error function relative to the learning rate $\eta_{kj}(n)$ in contrast to a single learning rate, $\eta(n)$ for all weights in (9) for the momentum method. However, as pointed out by Jacobs, this learning rate update rule becomes unpractical if the local curvature of the error surface is very small or very high, which is encountered in the training patterns for the fatigue damage model. Therefore, we propose a modified back-propagation algorithm in which the momentum method is modified by including a learning rate update rule. This algorithm combines the accelerating methods with part of the back-propagation modifications as suggested by Vogl *et al* (1988). The modified back-propagation rule is delineated below for the three-layer network in figure 3:

$$\Delta\eta_{jk}(n+1) = \begin{cases} \beta\eta_{jk}(n) & \text{if } \bar{\varphi}_{jk}(n-1)\varphi_{jk}(n) \geq 0 \\ -\phi\eta_{jk}(n) & \text{if } \bar{\varphi}_{jk}(n-1)\varphi_{jk}(n) < 0 \end{cases} \tag{10}$$

where the non-negative scalars $\beta$ and $\phi$ are user-selected learning-rate adjustment parameters; and

$$\varphi_{jk}(n) = \frac{\partial E(n)}{\partial w_{jk}(n)} \quad \text{and} \quad \bar{\varphi}_{jk}(n) = (1-\theta)\varphi_{jk}(n) + \theta\bar{\varphi}_{jk}(n-1) \quad \text{for } \theta \in [0,1).$$

Figure 3 shows the flowchart of the modified back-propagation algorithm. First the weights, $\{w^1_{ji}\}$, $\{w^2_{kj}\}$, and $\{w^3_{lk}\}$ of the network shown in figure 2 are initialized in step 1, and the error norm in the previous step is set to a large value. Now the learning process begins to feed an input pattern into the network to accumulate the squared errors of the corresponding output pattern until every training pattern is presented in step 2. If the norm of the current error in step 3 is less than the previous error, the algorithm switches to step 4.1 to adjust the learning rates based on (10). Then step 5.1 calculates the new weights by using the momentum method in (8). If the current error in step 3 exceeds the previous error, then the algorithm switches to step 4.2 to adjust the learning rates that are attenuated by a factor $\psi < 1$. Then step 5.2 calculates the new weights by setting $\alpha = 0$ in (8), i.e. without using the momentum term. The iterative procedure of learning is continued until the error

**Table 1.** The user-selected parameters.

| | |
|---|---|
| Momentum parameter | $\alpha = 0.9$ |
| Learning-rate adjustment parameters | $\beta = 0.05$ |
| | $\phi = 0.13$ |
| Attenuation parameter in step 4.2 of figure 3 | $\psi = 0.7$ |
| Error criterion in step 6 of figure 3 | $\xi = 0.0012$ |

criterion $\xi$ is satisfied in step 6. The user-selected parameters are listed in table 1.

## 4. Simulation results and discussion

This section presents a neural network structure of fatigue damage dynamics while focusing on simulation experiments to demonstrate efficacy of the network. As indicated in figure 4, the neural network model of fatigue damage dynamics consists of two functional blocks: a rainflow counting block and a neural network block. First, a load sequence $\{\sigma_k\}$ acting on a critical component of the mechanical structure is injected into the rainflow counting block to identify a series of time-dependent stresses and the corresponding reference stresses. The stress $\sigma_k$, reference stress $\sigma_{r,k}$, the total damage $D_k$ at the instant $k$, and a bias $-1$ are propagated forward to the three-layer neural network block as four inputs to obtain the single output, namely, the damage increment $\Delta D_k$. The rationale for using the bias $-1$ as an additional input is to provide a threshold to each neuron in the network (Haykin, 1994). The instantaneous cumulative fatigue damage $D_{k+1}$ is fed back into the input stage of the neural network block as the input for the next time instant. The damage rate is approximated by the first-order forward differential equation for the sampling time $\Delta t$ as:

$$\dot{D}_k \approx \frac{D_{k+1} - D_k}{\Delta t} \qquad (11)$$

The training set consists of a total of 1260 input-output patterns generated from the fatigue damage model in section 2. Note that the input patterns are uniformly mapped into or are shifted to a suitable range for training, and that the output patterns are taken in the logarithm scale to circumvent the numerical resolution problem. The range of the data set for

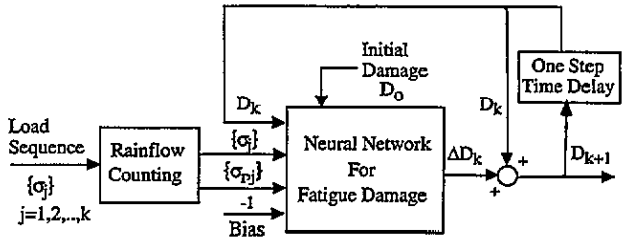


**Figure 4.** Neural network model of fatigue damage dynamics at the $k$th instant.

training the neural network was selected corresponding to an early stage of crack initiation as listed below:

Range of input patterns:

- $D_k \in [0.01, 0.5]$ with an increment of 0.005 is scaled over [0.1, 0.9]
- $\sigma_k \in [70, 90]$ ksi which implies that $\sigma_{r,k} \in [70, 90]$ and $\sigma_k \in [\sigma_{r,k}, 90]$

To facilitate implementation of the activation functions in (5), ranges of $\sigma_{r,k}$ and $\sigma_k$ are shifted as follows:

- $\sigma_{r,k} \in [1, 21]$ with an increment of 4
- $\sigma_k \in [\sigma_{r,k}, 21]$ with an increment of 0.4.

Range of output patterns:

- Logarithm of $\Delta D_k \in [2.173 \times 10^{-30}, 8.149 \times 10^{-15}]$ is taken and then scaled over [0.1, 0.9].

The number of neurons in the hidden layers is an important parameter in the back-propagation network. Lippmann (1987) has discussed how decision regions are formed by single- and multi-layer perceptrons with one and two hidden layers and two inputs. This discussion is primarily based on the networks in which hard limiting non-linearities are used. Since it is difficult to analyze the neural network problems with sigmoid activation functions, a trial and error approach has been used to decide the number of neutrons. The results of three different configurations are shown in table 2. The configuration column shows the number of neurons in each layer of the networks. The number of parameters in the networks with different configurations are different as shown in the column of the number of weights. Mean and standard deviation of epochs until a solution is reached for each case are listed in table 2, in which the initial weights in fifteen tests are randomly chosen. It follows from table 2 that the number of iterations needed to reach a solution, satisfying the error criteria in table 1 on the average, decreases with an increase in the number of neurons. A

**Table 2.** Comparison of results of back-propagation simulation.

| | | | Epochs until solution | |
|---|---|---|---|---|
| Configuration | Number of weights | Number of tests | Mean | Standard deviation |
| Case A 4-20-16-1 | 416 | 15 | 6664.0 | 6244.3 |
| Case B 4-18-12-1 | 300 | 15 | 8033.1 | 8174.9 |
| Case C 4-13-11-1 | 206 | 15 | 12070.3 | 10447.3 |

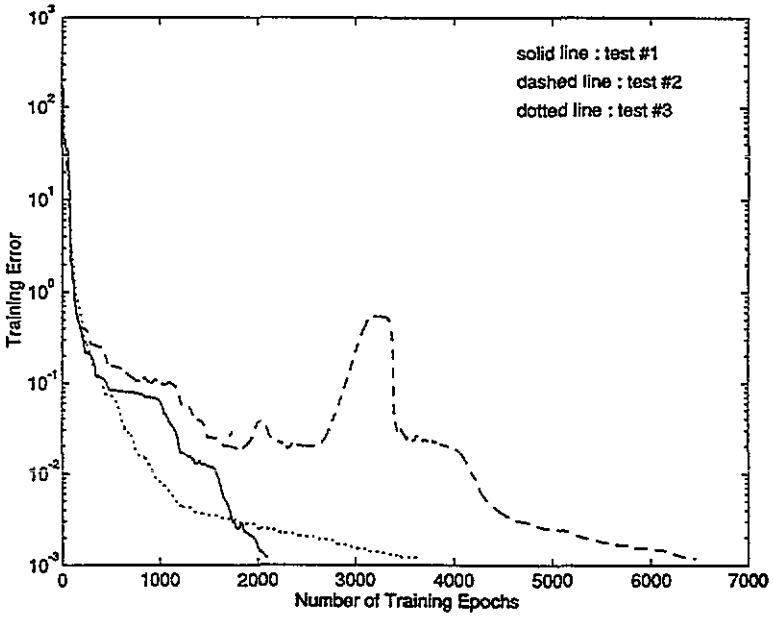


**Figure 5.** Profile of transient errors during training of the neural network.

large number of neurons generate more local minima in the error surface of the network, which might satisfy the error criterion and, therefore, it is more likely that the network would find a solution in fewer iterations. However, further research is needed to establish general validity of this observation.

Figure 5 exhibits the transient errors during the training phase until solutions are reached for three different tests corresponding to case A in table 2. It is the randomly chosen initial weights that result in different transient paths and different total number of epochs. Figure 6 shows the relative error of 1000 training patterns, which is defined as the difference of damage increments between desired solutions and neural network solutions. The relative error of training results are largely confined within 5 percent. The error threshold $\xi$ listed in table 1 was used to terminate the training of the network as seen in the flowchart of figure 3. Other approaches such as the radial basis function technique may lead to faster convergence of the network weights.

Figure 7 shows a comparison of the results of the continuous-time damage model with those of the trained neural network model. A sequence of 20 000 random inputs of load (i.e. stress) was fed into both the continuous-time damage model presented in section 2 and the neural network model of figure 4. The relative error of the neural network model with respect to the analytical model is within 10 percent for most of

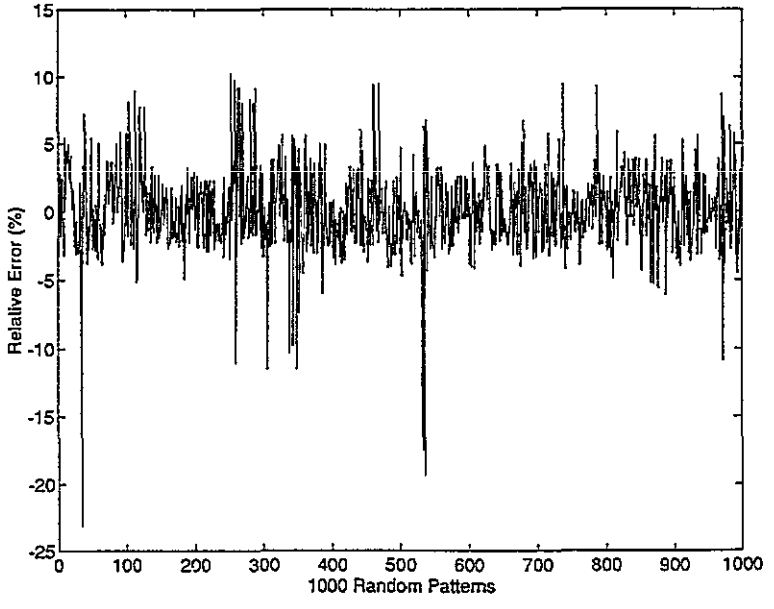


**Figure 6.** Relative error of 1000 random patterns for testing the training results.

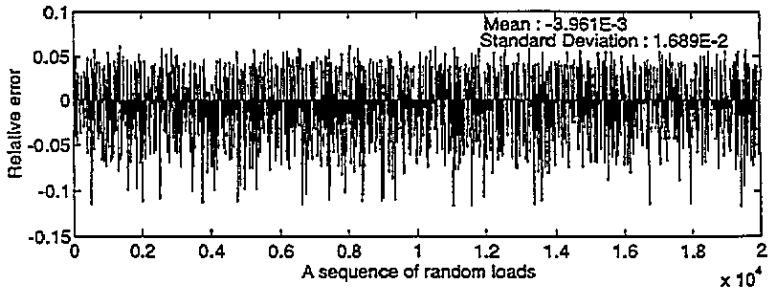


**Figure 7.** Comparison of the results generated from analytical and neural network models.

the random load, and the mean and standard deviation of the relative error are $-3.961 \times 10^{-3}$ and $1.689 \times 10^{-2}$, respectively. Ray and Wu (1994b) have reported that mean and standard deviation of relative error of the analytical model, described in section 2, is of the order of $10^{-2}$ and $10^{-1}$, respectively, when compared with the test data obtained from fatigue damage experiments. The agreement between the analytical model and the neural network model is one order of magnitude closer than that between the analytical model and the experimental data. Therefore, it is concluded that the neural network model developed in this paper has the potential of serving as an intelligent instrument for on-line monitoring of fatigue damage dynamics.

## 5. Conclusions and future work

The notion of damage-mitigation as perceived in this paper is to generate the information of fatigue damage for the critical plant components so that decisions for damage monitoring, diagnosis, prognosis and control can be made for structural durability and life extension of these components. A fatigue damage model (Ray and Wu, 1994a), formulated in the continuous-time setting, has been used for this purpose. However, traditional numerical techniques may not satisfy the stringent requirements of data processing time for simultaneous damage prediction at several critical points because the damage model involves complex non-linear calculations. Potentially, an artificial neural network can be used to circumvent the problems associated with data processing delays. Since errors surfaces in the back-propagation neural network may slow down convergence of the training procedure, a modified back-propagation algorithm has been developed to accelerate the convergence rate. This procedure is a combination of two methods, namely, momentum and adaptive learning rate. On this basis, a neural network has been synthesized and its prediction of fatigue damage is shown to be in close agreement with that of the non-linear differential equations that represent the fatigue damage dynamics. These results of simulation experiments suggest that this neural network algorithm is potentially an intelligent instrument for on-line monitoring of fatigue damage dynamics and also a tool for prediction of the remaining service life.

Future research is recommended for application of this neural network structure for on-line estimation of the cumulative damage at several critical points such as blades of gas turbines. In this approach, the cumulative damage is fed back after one time step delay to the input stage of the neural network to predict the damage increment during the next time step. Due to the property of highly parallel structure in the neural network, the cumulative damage of these critical points can be very quickly obtained as a part of the real-time control system. Therefore, for those critical points in which the range of operating load is similar, a single neural-network-based damage predictor can be multiplexed to obtain the necessary information in real time. This implementation may also serve as an instrument for on-line monitoring of fatigue damage dynamics.

The neural network approach presented in this paper can also incorporate alternative fatigue damage models, such as the small fatigue crack model of Newman (1992) which can be converted from the cycle base to the continuous-time base following the procedure outlined by Ray *et al* (1994a). Nevertheless, the proposed neural-network-based damage predictor should be updated with new models of fatigue and damage accumulation as they become available.

## Acknowledgments

The authors acknowledge benefits of discussion with Dr Xiaowen Dai and Mr Sekhar Tangirala during the course of this research. The work reported in this paper was supported in part by the National Science Foundation under research grant no. ECS-9216386 and the Electric Power Research Institute under contract no. EPRI RP8030-5.

## References

Bannantine J A, Comer J J and Handrock J L 1990 *Fundamentals of Metal Fatigue Analysis* (New Jersey: Prentice Hall)

Bolotin V V 1989 *Prediction of Service Life for Machines and Structures* ASME Press

Dowling N E 1983 Fatigue life prediction for complex load versus time histories *J. Eng. Mater. Technol, Trans. ASME* **105** 206–214

Haykin S 1994 *Neural Networks: A Comprehensive Foundation* (New York: Macmillan)

Hornik K, Stinchcombe M and White H 1989 Multi-layer feedforward networks are universal approximators *Neural Networks* **2** 359–366

Jacobs R A 1988 Increased rates of convergence through learning rate adaptation *Neural Network* **1** 295–307

Lippmann R P 1987 An introduction to computing with neural nets *IEEE ASSP Magazine* **4**(2) 4–22

Newman J C Jr 1992 Fracture mechanics parameters for small fatigue cracks *Small Crack Test Methods, ASTM STP 1149* eds J Larsen and J E Allison American Society of Testing and Materials, Philadelphia 6–33

Paris P C and Erdogan F 1963 A critical analysis of crack propagation laws *J. Basic Eng., Trans. ASME* **D85** 528–534

Ray A and Wu M-K 1994a Fatigue damage control of mechanical systems *Smart Mater. and Struct.* **3** 1 47–58

Ray A and Wu M-K 1994b Damage-mitigating control of space propulsion systems for high performance and extended life, NASA Lewis Research Center *NASA Contractor Report* 194470

Ray A, Wu M-K, Carpino M and Lorenzo C F 1994a Damage-mitigating control of mechanical systems: part I—conceptual development and model formulation *ASME J. Dyn. Syst., Meas. Control* **116** 437–447

Ray A, Wu M-K, Carpino M and Lorenzo C F 1994b Damage-mitigating control of mechanical systems: part II—formulation of an optimal policy and simulation *ASME J. Dyn. Syst., Meas. Control* **116** 448–455

Ray A, Dai X, Wu M-K, Carpino M and C F Lorenzo 1994c Damage-mitigating control of a reusable rocket engine *AIAA J. Propulsion and Power* **10** 225–234

Rumelhart D E and McClelland 1986 eds 1986 *Parallel Distributed Processing: Exploration in the microstructure of cognition* **1** Foundations 318–362 (Cambridge, MA: MIT Press)

Rychlik I 1993 Notes on cycle counts in irregular loads *Fatigue Fract. Eng. Mater. Struct.* **16** 377–390

Suresh S 1991 *Fatigue of Materials* (Cambridge: Cambridge University Press)

Swain M H, Everett R A, Newman J C Jr and Phillips E P 1990 The growth of short cracks in 4340 steel and aluminium-lithium 2090 *AGARD Report* 767 7.1–7.30

Troudet T and Merrill W 1990 A real time neural net estimator of fatigue life *Proc. ICJNN* (San Diego) 17–21

Vogl T P, Mangis J K, Rigler A K, Zink W T and Alkon D L 1988 Accelerating the convergence of the back-propagation method *Biological Cybernetics* **59** 257–263