A Particle Swarm Optimization Approach for Training Artificial Neural Networks with Uncertain Data

Steffen Freitag[1,2], Rafi L. Muhanna[1] and Wolfgang Graf [2]
[1]*School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA, steffen.freitag@gtsav.gatech.edu, rafi.muhanna@gtsav.gatech.edu*
[2]*Institute for Structural Analysis, Technische Universität Dresden, 01062 Dresden, Germany, steffen.freitag@tu-dresden.de, wolfgang.graf@tu-dresden.de*

Abstract. Artificial neural networks are powerful tools to learn functional relationships between data. They are widely used in engineering applications. Recurrent neural networks for fuzzy data have been introduced to map uncertain structural processes with deterministic or uncertain network parameters. Based on swarm intelligence, a new training strategy for neural networks is presented in this paper. Accounting for uncertainty in measurements, particle swarm optimization (PSO) approaches using interval and fuzzy numbers are developed. Applications are focused on the description of time-dependent material behavior with recurrent neural networks for uncertain data within interval and fuzzy finite element analyses. Network training with PSO allows to create special network structures with dependent parameters in order to consider physical boundary conditions of investigated materials.

Keywords: particle swarm optimization; neural network; uncertainty; interval numbers; fuzzy numbers, constitutive material description, finite element method

1. Introduction

Reliability assessment of structures requires knowledge of its behavior under environmental influences. Information on the structural behavior may be obtained by structural monitoring. Existing structures can be investigated by in situ monitoring whereas material tests can be performed to investigate new materials. As a result of experimental investigations, data series for measured structural actions and responses are available. Measured results are more or less characterized by data uncertainty due to varying boundary conditions, inaccuracies in measurements, and / or incomplete sets of observations. Interval or fuzzy numbers can be used to represent imprecise parameters, see e.g. (Möller and Beer, 2008). Time-dependent structural parameters are quantified as interval or fuzzy processes.

Functional relationships between uncertain data are required to describe the observed physical phenomena. Commonly, constitutive models are used for stress-strain relationships. Their parameters must be identified by an inverse analysis. If no closed-form expression can be obtained, optimization approaches can be applied to determine the unknown parameters of a predefined model.

An alternative approach to get functional relationships between uncertain data is the application of artificial intelligence. Artificial neural networks are widely used in engineering. Fields of applications in civil engineering are presented, e.g. in (Adeli, 2001). Often, multilayer perceptrons with feed forward architecture are utilized to learn functional relationships in deterministic data. For this purpose, several training strategies

5th International Conference on Reliable Engineering Computing (REC 2012)
Edited by M. Vořechovský, V. Sadílek, S. Seitl, V. Veselý, R. L. Muhanna and R. L. Mullen

ISBN 978-80-214-4507-9. Published by Ing. Vladislav Pokorný − LITERA

are available (Haykin, 1999). In (Graf et al., 2011), neural network approaches for structural analysis with uncertain data are discussed. For time-dependent phenomena, recurrent neural networks can be applied. These advanced network architectures enable the consideration of the whole history for the computation of current states, see e.g. (Oeser and Freitag, 2009). Recurrent neural networks for fuzzy data (Freitag et al., 2011a) have been developed to identify deterministic dependencies (Graf et al., 2010) or uncertain dependencies (Freitag et al., 2011c) in fuzzy processes. In (Freitag et al., 2010a), a backpropagation training algorithm for recurrent neural networks with trainable fuzzy network parameters has been introduced. It is a gradient based approach using the derivatives of fuzzy activation functions. In general, interval arithmetic (Moore, 1979) or α-level optimization (Möller et al., 2000) can be used to compute the signals of recurrent neural networks for fuzzy data, see (Freitag, 2010). In this paper, a new training strategy for recurrent neural networks is introduced considering both ways of computation. It is based on swarm intelligence (Kennedy et al., 2001).

Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) is an optimization concept motivated by social behavior of group individuals. It is a random search strategy and requires multiple evaluations of an objective function. After random initialization, each individual (denoted as particle) share its information with other particles in the swarm in order to define its new position in the space of search variables. In (Eberhart and Shi, 2001), developments and applications of PSO are discussed. Applications in civil engineering are presented e.g. in (Perez and Behdinan, 2007) and (Li et al., 2007). The approaches in these works can be used for optimization tasks with constraints.

One of the first applications of PSO was the training of artificial neural networks, see e.g. (Kennedy and Eberhart, 1995). Algorithms for feed forward neural networks are presented e.g. in (Mendes et al., 2002) and (Kuok et al., 2010). A hybrid training strategy, combining backpropagation and PSO for training of feed forward neural networks, is shown in (Zhang et al., 2007). Accounting for uncertain training data, PSO approaches using interval and fuzzy numbers are developed in this paper. They can be applied to feed forward and recurrent neural networks. The advantage of PSO for recurrent neural networks is that all network parameters can be modified during training. Additionally, special network structures with dependent parameters can be created. This is helpful to consider physical boundary conditions, if neural networks are used as constitutive models.

Recurrent neural networks for interval or fuzzy data are used to describe uncertain stress-strain-time dependencies. A finite element formulation for neural network based material descriptions is shown. Neural networks can be applied as constitutive models within interval finite element methods (Muhanna et al., 2007), (Rao et al., 2011), fuzzy finite element methods (Möller et al., 2000), (Moens and Vandepitte, 2005) or fuzzy stochastic finite element methods (Graf et al., 2011), (Sickert et al., 2011). Examples are presented to show the applicability of the new approach.

2. Uncertain data

2.1. Intervals and fuzzy numbers

Uncertain data can be represented as intervals or fuzzy numbers. An interval

$$\bar{x} = [{}_l x,\ {}_r x] \tag{1}$$

is defined by its left ${}_l x$ and right ${}_r x$ bounds. A bar $\bar{}$ is used to indicate intervals. It is also common to define an interval by its midpoint

$$ {}_m x = \frac{{}_l x + {}_r x}{2} \tag{2} $$

and its width

$$ {}_w x = {}_r x - {}_l x \ . \tag{3} $$

If midpoints and widths are used, the left and right interval bounds are obtained by

$$ {}_l x = {}_m x - \frac{{}_w x}{2} \tag{4} $$

and

$$ {}_r x = {}_m x + \frac{{}_w x}{2} \ , \tag{5} $$

respectively.

Fuzzy numbers $\tilde{x}$ are uncertain sets gradually assessed by membership functions $\mu(x)$. The tilde $\tilde{}$ is used to indicate fuzziness. The functional values of $\mu(x)$ are defined in $[0,\ 1]$. For each realisation x, its level of membership to the set $\tilde{x}$ is between 0 and 1. Considering convex fuzzy numbers, an interval

$$ {}_s\bar{x} = [{}_{sl} x,\ {}_{sr} x] \tag{6} $$

is obtained for each level s of membership $\alpha_s = \mu\left({}_{sl} x\right) = \mu\left({}_{sr} x\right)$. A set of $s = 1, \ldots, S$ cuts (α-cuts) can be used to approximate the membership function of a fuzzy number $\tilde{x}$ by piecewise linear functions, see Figure 1. The interval bounds of each α-cut are given by

$$ {}_{sl} x = \min \left[x \in \mathbb{R} \mid \mu(x) \geq \alpha_s \right] \tag{7} $$

and

$$ {}_{sr} x = \max \left[x \in \mathbb{R} \mid \mu(x) \geq \alpha_s \right] \ , \tag{8} $$

respectively. A fuzzy number can be represented by its α-cuts as a discrete set of the corresponding interval bounds. The fuzzy number

$$ \tilde{x} = \langle {}_{1l} x, \ldots, {}_{Sl} x, {}_{Sr} x, \ldots, {}_{1r} x \rangle \tag{9} $$

contains all left and right interval bounds as a sorted sequence. In general, the α-cut S (with $\mu(x) = 1$) can be an interval or a deterministic number. If it is a deterministic number, i.e. ${}_{Sl} x = {}_{Sr} x = {}_S x$, the number of elements in Eq. (9) is an odd number. At least three elements are required to define a fuzzy number. In this case, the fuzzy number $\tilde{x} = \langle {}_{1l} x,\ {}_2 x,\ {}_{1r} x \rangle$ has a membership function with triangular shape. With four elements $\tilde{x} = \langle {}_{1l} x,\ {}_{2l} x,\ {}_{2r} x,\ {}_{1r} x \rangle$, a membership function with trapezoidal shape is created. The α-cut representation of fuzzy numbers is common in engineering. It allows to handle fuzzy numbers similar to intervals in numerical simulations, i.e. interval operations can be performed for each α-cut.

2.2. Interval and fuzzy processes

Interval and fuzzy processes can be represented by series of interval or fuzzy numbers. The interval process

$$ \bar{x}(\tau) = \left\{ {}^{[1]}\bar{x}, \ldots, {}^{[n]}\bar{x}, \ldots, {}^{[N]}\bar{x} \right\} \tag{10} $$

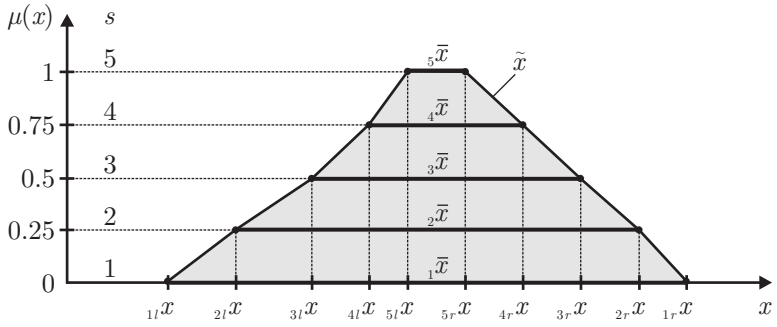

Figure 1. Fuzzy number represented by its α-cuts.

has discrete functional values (intervals ${}^{[n]}\bar{x}$) for each time point ${}^{[n]}\tau$. The time steps are equidistant, i.e. $\Delta\tau = {}^{[n]}\tau - {}^{[n-1]}\tau \quad \forall \; n = 2, \ldots, N$. Eq. (10) can also be formulated for fuzzy processes

$$\tilde{x}(\tau) = \left\{ {}^{[1]}\tilde{x}, \ldots, {}^{[n]}\tilde{x}, \ldots, {}^{[N]}\tilde{x} \right\} \; . \tag{11}$$

2.3. Functional relationships between interval or fuzzy processes

Mappings can be created to describe functional relationships between interval or fuzzy processes. Here, three types of mapping are regarded (exemplified for fuzzy processes – a formulation for interval processes is straightforward):

– **Type 1 mapping**

$$\tilde{\mathbf{x}}(\tau) \; \mapsto \; \tilde{\mathbf{z}}(\tau) \tag{12}$$

The vector of fuzzy processes $\tilde{\mathbf{x}}(\tau)$ is mapped onto the vector of fuzzy processes $\tilde{\mathbf{z}}(\tau)$ with deterministic mapping parameters.

– **Type 2 mapping**

$$\mathbf{x}(\tau) \; \tilde{\mapsto} \; \tilde{\mathbf{z}}(\tau) \tag{13}$$

The vector of deterministic processes $\mathbf{x}(\tau)$ is mapped onto the vector of fuzzy processes $\tilde{\mathbf{z}}(\tau)$ with fuzzy mapping parameters.

– **Type 3 mapping**

$$\tilde{\mathbf{x}}(\tau) \; \tilde{\mapsto} \; \tilde{\mathbf{z}}(\tau) \tag{14}$$

The vector of fuzzy processes $\tilde{\mathbf{x}}(\tau)$ is mapped onto the vector of fuzzy processes $\tilde{\mathbf{z}}(\tau)$ with fuzzy mapping parameters.

The Type 3 mapping is the general case. Type 1 and Type 2 mappings can be treated as special cases of the Type 3 mapping. The vector $\tilde{\mathbf{x}}(\tau)$ contain $j = 1, \ldots, J$ fuzzy components $\tilde{\mathbf{x}}_j(\tau)$, which are related to the $k = 1, \ldots, K$ fuzzy components $\tilde{\mathbf{z}}_k(\tau)$ of vector $\tilde{\mathbf{z}}(\tau)$. With respect to the representation of fuzzy processes in Eq. (11), the fuzzy number ${}^{[n]}\tilde{z}_k$ of time step $[n]$ can depend on all $j = 1, \ldots, J$ fuzzy numbers ${}^{[r]}\tilde{x}_j$ of prior and current time steps $[r] = [1], \ldots, [n]$.

Mathematical formulations are required to describe the mappings. These formulations contain unknown parameters which have to be identified by an inverse analysis. For Type 2 and Type 3 mappings, these parameters are fuzzy numbers or intervals. An optimization task can be formulated to identify deterministic, interval or fuzzy parameters.

2.4. Optimization task

The objective of an inverse analysis is to identify unknown deterministic, interval or fuzzy parameters. Forward analyses with deterministic processes $\mathbf{x}(\tau)$, interval processes $\bar{\mathbf{x}}(\tau)$ or fuzzy processes $\tilde{\mathbf{x}}(\tau)$ and predefined sets of parameters lead to interval processes $\bar{\mathbf{z}}^*(\tau)$ or fuzzy processes $\tilde{\mathbf{z}}^*(\tau)$. An optimization task can be performed to minimize the difference between computed results ($\bar{\mathbf{z}}^*(\tau)$ or $\tilde{\mathbf{z}}^*(\tau)$) and available data ($\bar{\mathbf{z}}(\tau)$ or $\tilde{\mathbf{z}}(\tau)$). The difference between computed and collected interval data is obtained by

$$E^h = \frac{1}{N}\frac{1}{K}\sum_{n=1}^{N}\left[\sum_{k=1}^{K}\left\{\left({}^{[n]}_{l}z_k - {}^{[n]}_{l}z^*_k\right)^2 + \left({}^{[n]}_{r}z_k - {}^{[n]}_{r}z^*_k\right)^2\right\}\right] , \tag{15}$$

whereas

$$E^h = \frac{1}{N}\frac{1}{K}\frac{1}{S}\sum_{n=1}^{N}\left[\sum_{k=1}^{K}\left\{\sum_{s=1}^{S}\left[\left({}^{[n]}_{sl}z_k - {}^{[n]}_{sl}z^*_k\right)^2 + \left({}^{[n]}_{sr}z_k - {}^{[n]}_{sr}z^*_k\right)^2\right]\right\}\right] \tag{16}$$

is used to evaluate the distance between computed and collected fuzzy data. If different patterns $h = 1, \ldots, H$ are available for parameter identification, the averaged error can be computed by

$$E^{av} = \frac{1}{H}\sum_{h=1}^{H}\left[E^h\right] . \tag{17}$$

The scaling with the number of patterns H, the number of time steps N, the number of components K, and the number of α-cuts S in the above Eqs. (15) to (17) is done due to practical reasons. It is easier to compare and evaluate errors with different selected and available numbers of H, N, K, and S.

Deterministic, interval or fuzzy parameters can be identified using Eq. (17) as objective function to be minimized. The optimization task can be solved by application of swarm intelligence.

3. Particle swarm optimization

Particle swarm optimization is a random search strategy motivated by social behavior of group individuals. Individuals of the group (swarm) are denoted as particles. Each particle is represented by a vector including all unknown parameters of the objective function – the space of search variables. First, the parameters of all particles $i = 1, \ldots, I$ of the swarm are randomly initialized. Then, the objective function is evaluated for each particle. The new position of each particle, i.e. a new set of parameters, is defined by its own search history, information of other particles, and random influences. Different ways of sharing information between particles in the swarm can be chosen, see e.g. (Fontan et al., 2011). Here, a fully connected topology is selected, i.e. each particle shares its information with all other particles in the swarm. This procedure is

applied to multiple runs $(r) = (1), \ldots, (R)$ until a predefined number of runs R is reached or the functional value of the objective function (Eq. (17)) is less than a predefined error value.

3.1. DETERMINISTIC PARAMETERS

Particle swarm optimization with deterministic particles can be used for parameter identification in case of Type 1 mapping (Eq. (12)). Each particle i is represented by a vector $\mathbf{a}^i$, which has $q = 1, \ldots, Q$ components a_q^i. The number of components Q is equal to the number of search variables, i.e. the dimension of the search space.

In each run (r), the objective function (Eq. (17)) is evaluated for each particle. The position of the best particle in the swarm, i.e. the set of parameters with the least value of the objective function in all runs, is stored as vector $\mathbf{g}$ (global best). Additionally, the best positions of each particle i are stored as vectors $\mathbf{p}^i$ (individual best).

Each component q of particle i is updated by

$$ {}^{(r+1)}a_q^i = {}^{(r)}a_q^i + {}^{(r)}\Delta a_q^i \ , \tag{18} $$

with

$$ {}^{(r)}\Delta a_q^i = c_3 \cdot {}^{(r-1)}\Delta a_q^i + c_1 \cdot d \cdot \left(p_q^i - {}^{(r)}a_q^i \right) + c_2 \cdot e \cdot \left(g_q - {}^{(r)}a_q^i \right) \tag{19} $$

for the next run $(r+1)$. In Eq. (19), d and e are realizations of independent uniformly distributed random variables in $[0,\ 1]$. For each particle i and each component q, different samples of d and e are chosen. The constants c_1, c_2, and c_3 are introduced to control the search behavior of the swarm. They are used to allow the selection of different weights for historical, individual best, and global best influences. It is common to restrict ${}^{(r)}\Delta a_q^i$ to

$$ \Delta_{\min} a_q \leq {}^{(r)}\Delta a_q^i \leq \Delta_{\max} a_q \ , \tag{20} $$

where $\Delta_{\min} a_q$ and $\Delta_{\max} a_q$ can be defined with respect to the assumed width of the q-th component, i.e. the q-th dimension of the search space, see e.g. (Eberhart and Shi, 2001).

The following conditions are defined for the first run $(r) = (1)$:

- random initialization of all particles ${}^{(1)}\mathbf{a}^i$ in the search space
- initial position is equal to individual best
- after evaluation of the objective function for all particles, best initial position is equal to global best
- prior incremental update is zero (${}^{(0)}\Delta a_q^i = 0$)

3.2. INTERVAL PARAMETERS

If interval parameters are required to map deterministic processes $\mathbf{x}(\tau)$ or interval processes $\bar{\mathbf{x}}(\tau)$ onto interval processes $\bar{\mathbf{z}}(\tau)$ (Type 2 and Type 3 mappings), an extension of the presented well known PSO algorithm is necessary. Particles $\bar{\mathbf{a}}^i$, global best $\bar{\mathbf{g}}$, and individual best $\bar{\mathbf{p}}^i$ of each particle are defined as interval numbers according to Section 2.

The update

$$ {}^{(r+1)}\bar{a}_q^i = {}^{(r)}\bar{a}_q^i + {}^{(r)}\Delta\bar{a}_q^i \tag{21} $$

is done by interval arithmetic, see e.g. (Moore, 1979). For the left bound

$$ {}^{(r)}_{l}\Delta a_q^i = {}^{(r)}_{m}\Delta a_q^i - \frac{{}^{(r)}_{w}\Delta a_q^i}{2} \tag{22} $$

and the right bound

$$ {}^{(r)}_{r}\Delta a_q^i = {}^{(r)}_{m}\Delta a_q^i + \frac{{}^{(r)}_{w}\Delta a_q^i}{2} \tag{23} $$

of ${}^{(r)}\Delta\bar{a}_q^i$, the midpoint and width representation of intervals is used, compare Eqs. (2) to (5). The incremental update of the midpoint is computed by

$$ {}^{(r)}_{m}\Delta a_q^i = c_3 \cdot {}^{(r-1)}_{m}\Delta a_q^i + c_1 \cdot d \cdot \left({}_{m}p_q^i - {}^{(r)}_{m}a_q^i\right) + c_2 \cdot e \cdot \left({}_{m}g_q - {}^{(r)}_{m}a_q^i\right) \ . \tag{24} $$

The width of interval ${}^{(r)}\Delta\bar{a}_q^i$ is obtained by

$$ {}^{(r)}_{w}\Delta a_q^i = \begin{cases} {}^{(r)}_{w}\Delta\hat{a}_q^i, & \text{if } {}^{(r)}_{w}\Delta\hat{a}_q^i \geq 0 \\ 0, & \text{if } {}^{(r)}_{w}\Delta\hat{a}_q^i < 0 \ , \end{cases} \tag{25} $$

with

$$ {}^{(r)}_{w}\Delta\hat{a}_q^i = c_3 \cdot {}^{(r-1)}_{w}\Delta a_q^i + c_1 \cdot d \cdot \left({}_{w}p_q^i - {}^{(r)}_{w}a_q^i\right) + c_2 \cdot e \cdot \left({}_{w}g_q - {}^{(r)}_{w}a_q^i\right) \ . \tag{26} $$

It should be noted, that different realizations (d and e) of independent uniformly distributed random variables are used in Eqs. (24) and (26).

3.3. Fuzzy parameters

For Type 2 and Type 3 mappings of deterministic processes $\mathbf{x}(\tau)$ or fuzzy processes $\tilde{\mathbf{x}}(\tau)$ onto fuzzy processes $\tilde{\mathbf{z}}(\tau)$, fuzzy parameters are required. PSO can be extended to fuzzy particles, i.e. particles $\tilde{\mathbf{a}}^i$, global best $\tilde{\mathbf{g}}$, and individual best $\tilde{\mathbf{p}}^i$ of each particle are fuzzy numbers, see Section 2.

Fuzzy arithmetic operations are performed (interval arithmetic for each α-cut) to get the updated particle position

$$ {}^{(r+1)}\tilde{a}_q^i = {}^{(r)}\tilde{a}_q^i + {}^{(r)}\Delta\tilde{a}_q^i \ . \tag{27} $$

For ${}^{(r)}\Delta\tilde{a}_q^i$, the left and right bounds of each α-cut s are given by

$$ {}^{(r)}_{sl}\Delta a_q^i = {}^{(r)}_{sm}\Delta a_q^i - \frac{{}^{(r)}_{sw}\Delta a_q^i}{2} \tag{28} $$

and

$$ {}^{(r)}_{sr}\Delta a_q^i = {}^{(r)}_{sm}\Delta a_q^i + \frac{{}^{(r)}_{sw}\Delta a_q^i}{2} \ , \tag{29} $$

respectively. For α-cut $s = 1$, the incremental update of the midpoint

$$ {}^{(r)}_{1m}\Delta a^i_q = c_3 \cdot {}^{(r-1)}_{1m}\Delta a^i_q + c_1 \cdot d \cdot \left({}_{1m}p^i_q - {}^{(r)}_{1m}a^i_q \right) + c_2 \cdot e \cdot \left({}_{m}g_q - {}^{(r)}_{1m}a^i_q \right) \tag{30} $$

is computed similar to the interval approach in Section 3.2, compare Eq. (24). The same holds for the width of the interval ${}^{(r)}_{1}\Delta \bar{a}^i_q$, i.e.

$$ {}^{(r)}_{1w}\Delta a^i_q = \begin{cases} {}^{(r)}_{1w}\Delta \hat{a}^i_q, & \text{if } {}^{(r)}_{1w}\Delta \hat{a}^i_q \geq 0 \\ 0, & \text{if } {}^{(r)}_{1w}\Delta \hat{a}^i_q < 0 \,, \end{cases} \tag{31} $$

with

$$ {}^{(r)}_{1w}\Delta \hat{a}^i_q = c_3 \cdot {}^{(r-1)}_{1w}\Delta a^i_q + c_1 \cdot d \cdot \left({}_{1w}p^i_q - {}^{(r)}_{1w}a^i_q \right) + c_2 \cdot e \cdot \left({}_{1w}g_q - {}^{(r)}_{1w}a^i_q \right) \,. \tag{32} $$

For all other α-cuts ($s > 1$), three cases are distinguished for the incremental update of the midpoint

$$ {}^{(r)}_{sm}\Delta a^i_q = \begin{cases} {}^{(r)}_{sm}\Delta \hat{a}^i_q, & \text{if } {}^{(r)}_{s-1l}\Delta a^i_q \leq {}^{(r)}_{sm}\Delta \hat{a}^i_q \leq {}^{(r)}_{s-1r}\Delta a^i_q \\ {}^{(r)}_{s-1l}\Delta a^i_q, & \text{if } {}^{(r)}_{s-1l}\Delta a^i_q > {}^{(r)}_{sm}\Delta \hat{a}^i_q \\ {}^{(r)}_{s-1r}\Delta a^i_q, & \text{if } {}^{(r)}_{s-1r}\Delta a^i_q < {}^{(r)}_{sm}\Delta \hat{a}^i_q \,, \end{cases} \tag{33} $$

with

$$ {}^{(r)}_{sm}\Delta \hat{a}^i_q = c_3 \cdot {}^{(r-1)}_{sm}\Delta a^i_q + c_1 \cdot d \cdot \left({}_{sm}p^i_q - {}^{(r)}_{sm}a^i_q \right) + c_2 \cdot e \cdot \left({}_{sm}g_q - {}^{(r)}_{sm}a^i_q \right) \tag{34} $$

and for the incremental update of the width

$$ {}^{(r)}_{sw}\Delta a^i_q = \begin{cases} {}^{(r)}_{sw}\Delta \hat{a}^i_q, & \text{if } 0 \leq {}^{(r)}_{sw}\Delta \hat{a}^i_q \leq {}^{(r)}_{sw}\Delta_{\max} \hat{a}^i_q \\ {}^{(r)}_{sw}\Delta_{\max} \hat{a}^i_q, & \text{if } {}^{(r)}_{sw}\Delta \hat{a}^i_q > {}^{(r)}_{sw}\Delta_{\max} \hat{a}^i_q \\ 0, & \text{if } {}^{(r)}_{sw}\Delta \hat{a}^i_q < 0 \,, \end{cases} \tag{35} $$

with

$$ {}^{(r)}_{sw}\Delta \hat{a}^i_q = c_3 \cdot {}^{(r-1)}_{sw}\Delta a^i_q + c_1 \cdot d \cdot \left({}_{sw}p^i_q - {}^{(r)}_{sw}a^i_q \right) + c_2 \cdot e \cdot \left({}_{sw}g_q - {}^{(r)}_{sw}a^i_q \right) \tag{36} $$

and

$$ {}^{(r)}_{sw}\Delta_{\max} \hat{a}^i_q = 2 \cdot \min \left[\left({}^{(r)}_{sm}\Delta a^i_q - {}^{(r)}_{s-1l}\Delta a^i_q \right), \left({}^{(r)}_{s-1r}\Delta a^i_q - {}^{(r)}_{sm}\Delta a^i_q \right) \right] \,. \tag{37} $$

Different realizations d and e of independent uniformly distributed random variables are used for each α-cut. If α-cut $s = S$ is restricted to give deterministic numbers, only midpoints are updated for α-cut S.

4. Artificial neural networks for interval and fuzzy data

Artificial neural network concepts can be applied to map deterministic processes $\mathbf{x}(\tau)$, interval processes $\bar{\mathbf{x}}(\tau)$ or fuzzy processes $\tilde{\mathbf{x}}(\tau)$ onto interval processes $\bar{\mathbf{z}}(\tau)$ or fuzzy processes $\tilde{\mathbf{z}}(\tau)$. Two ways of computation are possible to process interval or fuzzy data with neural networks:

1. interval arithmetic (for each α-cut)

2. optimization (α-level optimization)

Interval arithmetic approaches for deterministic network parameters are presented in (Graf et al., 2010). Extensions for a priori defined and trainable interval or fuzzy network parameters are published in (Freitag et al., 2011c) and (Freitag et al., 2011a), respectively. Algorithms for signal computation with α-level optimization can be found in (Freitag, 2010) and (Freitag et al., 2011b). In the following, the neural network approaches are formulated for fuzzy data and Type 3 mapping, see Eq. (14). However, they can also be applied to interval data or Type 1 and Type 2 mappings, which is straightforward.

4.1. FEED FORWARD NEURAL NETWORKS

If the fuzzy number ${}^{[n]}\tilde{z}_k$ of time step $[n]$ depends on the $j = 1, \ldots, J$ current fuzzy numbers ${}^{[n]}\tilde{x}_j$ only, feed forward networks can be used as mathematical formulation of the mappings introduced in Section 2.3. Neural networks with feed forward architecture consist of (M) layers, i.e. an input layer, $(M-2)$ hidden layers and an output layer. The number of input and output neurons is given by the number of components J and K, respectively. The number of hidden layers and neurons has to be defined with respect to the complexity of the formulation. In general, fully connected networks are considered, i.e. each neuron in layer (m) has synaptic connections to all neurons in the following layer $(m+1)$, see Figure 2. For specific applications, special network structures may be created, see Section 5.

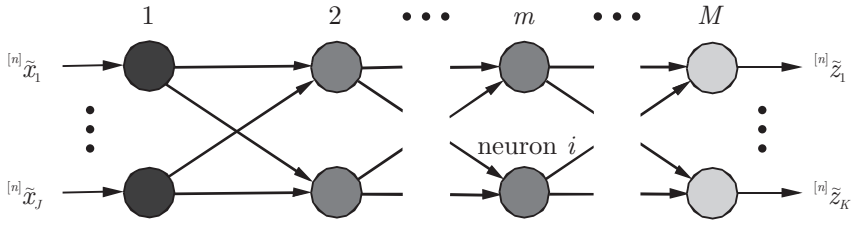

Figure 2. Feed forward neural network.

In each time step $[n]$, the fuzzy components ${}^{[n]}\tilde{x}_j$ (e.g. structural actions) may be transformed to dimensionless fuzzy network input signals, e.g.

$$ {}^{[n]}\tilde{x}_j^{(1)} = \frac{{}^{[n]}\tilde{x}_j}{x_j^{sc}} \, . \tag{38} $$

The dimensionless network output signals ${}^{[n]}\tilde{x}_k^{(M)}$ may be scaled to fuzzy components (e.g. structural responses)

$$ {}^{[n]}\tilde{z}_k = {}^{[n]}\tilde{x}_k^{(M)} \cdot z_k^{sc} \, . \tag{39} $$

For each component j and k, the scaling parameters x_j^{sc} and z_k^{sc} can be defined as the maximum absolute value of its possible positive and / or negative values.

The signals of feed forward neural networks are computed layer by layer. In the hidden and output neurons, fuzzy output signals

$$ {}^{[n]}\tilde{x}_i^{(m)} = \tilde{\varphi}_i^{(m)} \left(\sum_{h=1}^{H} \left[{}^{[n]}\tilde{x}_h^{(m-1)} \cdot \tilde{w}_{ih}^{(m)} \right] + \tilde{b}_i^{(m)} \right) \tag{40} $$

are computed by means of a fuzzy activation function $\tilde{\varphi}_i^{(m)}(.)$. These fuzzy output signals are transfered by synaptic connections to the neurons of the next layer. The argument of the fuzzy activation function of neuron i in layer (m) contains all fuzzy output signals ${}^{[n]}\tilde{x}_h^{(m-1)}$ of the previous layer $(m-1)$ multiplied by the fuzzy weights $\tilde{w}_{ih}^{(m)}$ and a fuzzy bias value $\tilde{b}_i^{(m)}$. Various types of monotonic and differentiable fuzzy activation functions can be used, see e.g. (Freitag, 2010).

The fuzzy weights, fuzzy bias values and perhaps parameters of the fuzzy activation function are unknown fuzzy network parameters. The PSO approaches presented in Section 3 can be used for parameter identification. It is proposed to initialize the particle components representing fuzzy weights and fuzzy bias values randomly, e.g. in $[-1,\ 1]$. In general, the values of these fuzzy parameters are not restricted in the search space. The search space can be restricted with respect to selected particle components representing the fuzzy factors of the fuzzy activation functions.

4.2. Recurrent neural networks

More general is the assumption, that all $j = 1, \ldots, J$ fuzzy numbers ${}^{[r]}\tilde{x}_j$ of prior and current time steps $[r] = [1], \ldots, [n]$ have influences to the current fuzzy number ${}^{[n]}\tilde{z}_k$ of time step $[n]$. In this case, recurrent neural networks are suitable to formulate the mappings according to Section 2.3.

In addition to feed forward networks, context neurons are used to consider the whole history for the computation of the current fuzzy number ${}^{[n]}\tilde{z}_k$ of time step $[n]$. All hidden and output neurons are connected to their context neurons, see Figure 3.

In each context neuron, the fuzzy output signal is transfered to the fuzzy context signal

$$ {}^{[n]}\tilde{y}_i^{(m)} = {}^{[n]}\tilde{x}_i^{(m)} + {}^{[n-1]}\tilde{y}_i^{(m)} \cdot \tilde{\lambda}_i^{(m)} \ . \tag{41} $$

The influence of the previous fuzzy context signal ${}^{[n-1]}\tilde{y}_i^{(m)}$ is considered by the fuzzy feedback factor $\tilde{\lambda}_i^{(m)}$. Fuzzy feedback factors $\tilde{\lambda}_i^{(m)}$ are additional fuzzy network parameters defined in the interval $[0,\ 1]$.

Each context neuron sends weighted fuzzy signals with a time delay of one time step to all hidden or output neurons in its layer. Hence, Eq. (40) must be extended to

$$ {}^{[n]}\tilde{x}_i^{(m)} = \tilde{\varphi}_i^{(m)} \left(\sum_{h=1}^{H} \left[{}^{[n]}\tilde{x}_h^{(m-1)} \cdot \tilde{w}_{ih}^{(m)} \right] + \sum_{q=1}^{I} \left[{}^{[n-1]}\tilde{y}_q^{(m)} \cdot \tilde{c}_{iq}^{(m)} \right] + \tilde{b}_i^{(m)} \right) \tag{42} $$

in order to consider the fuzzy context signals ${}^{[n-1]}\tilde{y}_q^{(m)}$ multiplied by the fuzzy context weights $\tilde{c}_{iq}^{(m)}$. It should be noted, that a feed forward neural network is obtained as a special case of the discussed recurrent neural network, if all fuzzy context weights are set to zero.

The fuzzy context weights $\tilde{c}_{iq}^{(m)}$ and fuzzy feedback factors $\tilde{\lambda}_i^{(m)}$ are additional unknown fuzzy network parameters, which can be identified by the introduced PSO approaches, see Section 3. The search space

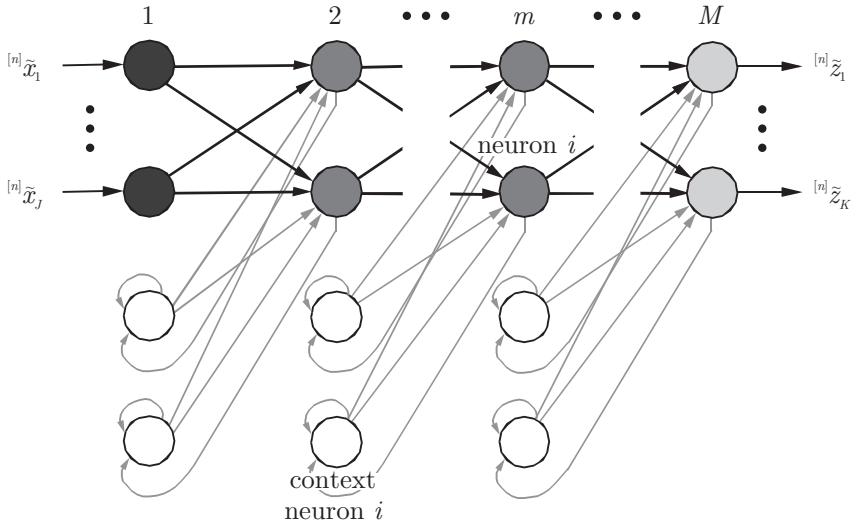

Figure 3. Recurrent neural network.

must be restricted to $[0,\ 1]$ for the particle components, which represent fuzzy feedback factors $\tilde{\lambda}_i^{(m)}$. If an updated interval bound is less than zero or greater than one, it is set to zero or to one, respectively. The search space is not restricted for fuzzy context weights $\tilde{c}_{iq}^{(m)}$. They can be initialized randomly in $[-1,\ 1]$.

5. Application for time-dependent material behavior

The presented neural network approaches can be applied to describe uncertain material behavior. Uncertain nonlinear stress-strain dependencies can be identified with feed forward neural networks for elastic material behavior. For nonlinear stress-strain-time dependencies (viscous material behavior), recurrent neural networks can be utilized. An α-level optimization is applied to compute the network outputs.

Fuzzy strain processes can be mapped onto fuzzy stress processes or vice versa. Here, an approach for strain to stress mapping is presented. In this case, the fuzzy processes $\tilde{\mathbf{x}}(\tau)$ represent fuzzy strain processes $\tilde{\varepsilon}(\tau)$ and the fuzzy processes $\tilde{\mathbf{z}}(\tau)$ correspond to fuzzy stress processes $\tilde{\sigma}(\tau)$. The strain and stress vectors include all components, which are required for strain and stress tensors ($J = K = 6$ for 3D, $J = K = 3$ for 2D and $J = K = 1$ for 1D material models). The fuzzy network parameters can be identified by results of experimental investigations.

5.1. Tangential stiffness

Applications of neural network based constitutive models within the finite element method require the tangential stiffness matrix of the material description ${}^{[n]}\tilde{\mathbf{C}}$ in order to get the tangential system stiffness matrix. The components of the uncertain tangential stiffness matrix

$$ {}^{[n]}\tilde{C}_{kj} = \frac{\partial {}^{[n]}\Delta\tilde{\sigma}_k}{\partial {}^{[n]}\Delta\tilde{\varepsilon}_j} \tag{43} $$

are determined in linearized form by the partial derivatives of the incremental fuzzy stress components

$$ {}^{[n]}\Delta\tilde{\sigma}_k = {}^{[n]}\tilde{\sigma}_k - {}^{[n-1]}\tilde{\sigma}_k \tag{44} $$

with respect to the incremental fuzzy strain components

$$ {}^{[n]}\Delta\tilde{\varepsilon}_j = {}^{[n]}\tilde{\varepsilon}_j - {}^{[n-1]}\tilde{\varepsilon}_j \ . \tag{45} $$

The incremental fuzzy stresses

$$ {}^{[n]}\Delta\tilde{\sigma}_k = \left({}^{[n]}\tilde{x}_k^{(M)} - {}^{[n-1]}\tilde{x}_k^{(M)} \right) \cdot z_k^{sc} \tag{46} $$

contain the fuzzy output signals of the neural network ${}^{[n]}\tilde{x}_k^{(M)}$ (time step $[n]$) and ${}^{[n-1]}\tilde{x}_k^{(M)}$ (time step $[n-1]$). The chain rule is applied two times in Eq. (43), which leads to

$$ \frac{\partial^{[n]}\Delta\tilde{\sigma}_k}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{\partial\left({}^{[n]}\tilde{\sigma}_k - {}^{[n-1]}\tilde{\sigma}_k \right)}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{\partial^{[n]}\tilde{\sigma}_k}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{\partial^{[n]}\tilde{\sigma}_k}{\partial^{[n]}\tilde{x}_k^{(M)}} \cdot \frac{\partial^{[n]}\tilde{x}_k^{(M)}}{\partial^{[n]}\tilde{x}_j^{(1)}} \cdot \frac{\partial^{[n]}\tilde{x}_j^{(1)}}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} \ . \tag{47} $$

The partial derivatives of the fuzzy stress components with respect to the fuzzy output signals in Eq. (47) are obtained by

$$ \frac{\partial^{[n]}\tilde{\sigma}_k}{\partial^{[n]}\tilde{x}_k^{(M)}} = \frac{\partial\left({}^{[n]}\tilde{x}_k^{(M)} \cdot z_k^{sc} \right)}{\partial^{[n]}\tilde{x}_k^{(M)}} = z_k^{sc} \ . \tag{48} $$

The partial derivatives of the fuzzy input signals with respect to the incremental fuzzy strain components

$$ \frac{\partial^{[n]}\tilde{x}_j^{(1)}}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{\partial\left(\frac{{}^{[n]}\tilde{\varepsilon}_j}{x_j^{sc}} \right)}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{1}{x_j^{sc}} \cdot \frac{\partial\left({}^{[n]}\Delta\tilde{\varepsilon}_j + {}^{[n-1]}\tilde{\varepsilon}_j \right)}{\partial^{[n]}\Delta\tilde{\varepsilon}_j} = \frac{1}{x_j^{sc}} \tag{49} $$

in Eq. (47) are evaluated using Eq. (38) (with ${}^{[n]}\tilde{x}_j = {}^{[n]}\tilde{\varepsilon}_j$) and (45). Eqs. (48) and (49) are substituted in Eq. (47) and hence, the components of the tangential stiffness matrix are obtained by

$$ {}^{[n]}\tilde{C}_{kj} = \frac{z_k^{sc}}{x_j^{sc}} \cdot \frac{\partial^{[n]}\tilde{x}_k^{(M)}}{\partial^{[n]}\tilde{x}_j^{(1)}} \ . \tag{50} $$

The partial derivatives of the network output signals ${}^{[n]}\tilde{x}_k^{(M)}$ with respect to the network input signals ${}^{[n]}\tilde{x}_j^{(1)}$ are evaluated using multiple applications of the chain rule. An efficient algorithm to compute these partial derivatives is presented in (Freitag et al., 2011b).

5.2. SPECIAL NETWORK STRUCTURES

Physical boundary conditions of investigated materials can be considered by creating special network structures. The tangential stiffness matrix should be symmetric for materials with isotropic properties. This condition (${}^{[n]}\tilde{C}_{kj} = {}^{[n]}\tilde{C}_{jk}$) can be fulfilled by

$$\frac{\partial {}^{[n]}\tilde{x}_k^{(M)}}{\partial {}^{[n]}\tilde{x}_j^{(1)}} = \frac{\partial {}^{[n]}\tilde{x}_j^{(M)}}{\partial {}^{[n]}\tilde{x}_k^{(1)}} \ , \tag{51}$$

if $x_j^{sc} = x_k^{sc}$ and $z_k^{sc} = z_j^{sc}$, see Eq. (50). Symmetric partial derivatives of the network output signals ${}^{[n]}\tilde{x}_k^{(M)}$ with respect to the network input signals ${}^{[n]}\tilde{x}_j^{(1)}$ can be guaranteed for networks with three layers and linear activation functions (with the same slope parameter) in the output layer. In Figure 4, the symmetry of the synaptic connections is exemplified. The symmetry condition for deterministic, interval or fuzzy weights

$$\tilde{w}_{ij}^{(2)} = \tilde{w}_{ji}^{(3)} \tag{52}$$

is also valid for recurrent neural networks and arbitrary numbers of neurons in the three layers.

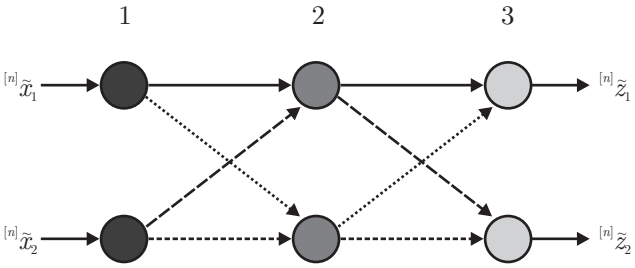

Figure 4. Neural network with symmetric derivatives.

Some stress and strain components are decoupled for isotropic or orthotropic behavior, i.e. the stress component ${}^{[n]}\tilde{\sigma}_k$ only depends on its corresponding strain component ${}^{[n]}\tilde{\varepsilon}_k$. This can be achieved by partially connected neural networks, see Figure 5. Selected weights and context weights are set to zero, which is equivalent to cut synaptic connections in a fully connected feed forward or recurrent neural network.

A simple feed forward network with two layers (no hidden layers) and linear activation functions (identity function) is equivalent to Hooke's law for linear elastic material. In Figure 6, the neural network representation of linear elastic material is demonstrated for the 3D case. The deterministic, interval or fuzzy weights are $\tilde{w}_{11} = \tilde{w}_{22} = \tilde{w}_{33} = \tilde{c}_1$, $\tilde{w}_{44} = \tilde{w}_{55} = \tilde{w}_{66} = \tilde{c}_2$, $\tilde{w}_{12} = \tilde{w}_{13} = \tilde{w}_{23} = \tilde{w}_{21} = \tilde{w}_{31} = \tilde{w}_{32} = \tilde{c}_3$ for isotropic material behavior, see Eq. (53).

$$\tilde{\mathbf{C}} = \begin{bmatrix} \tilde{c}_1 & \tilde{c}_3 & \tilde{c}_3 & 0 & 0 & 0 \\ \tilde{c}_3 & \tilde{c}_1 & \tilde{c}_3 & 0 & 0 & 0 \\ \tilde{c}_3 & \tilde{c}_3 & \tilde{c}_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \tilde{c}_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \tilde{c}_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \tilde{c}_2 \end{bmatrix} \tag{53}$$

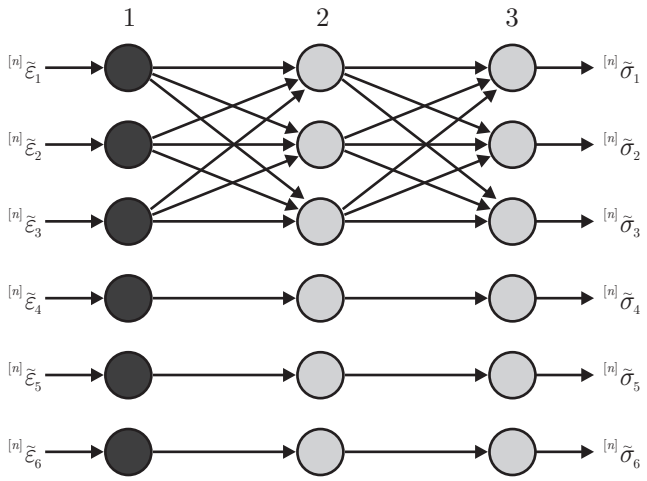

Figure 5. Partially connected neural network with symmetric derivatives.

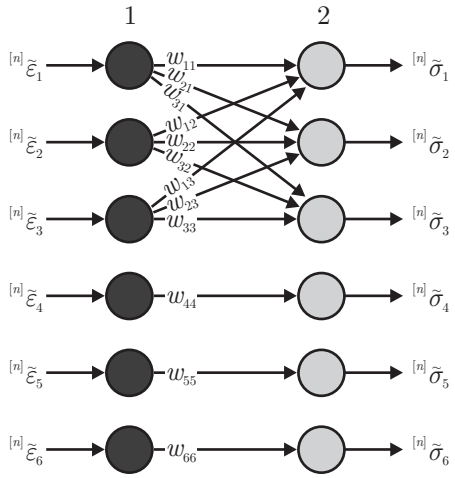

Figure 6. Feed forward neural network for linear elastic material behavior.

6. Examples

6.1. Verification with 1D fractional rheological model

The presented recurrent neural network approach is applied to identify and to predict uncertain stress-strain-time dependencies of the fuzzy fractional Newton element. The differential equation

$$\tilde{\sigma}(\tau) = \tilde{p}\,\frac{\mathrm{d}^{\tilde{r}}}{\mathrm{d}\tau^{\tilde{r}}}\,\tilde{\varepsilon}(\tau) \tag{54}$$

of this rheological element, see e.g. (Oeser and Freitag, 2009), contains a fractional derivative of strain $\tilde{\varepsilon}(\tau)$ with respect to time τ. In this example, $\tilde{p}$ is defined as deterministic parameter $\tilde{p} = p = 101\,000\ \ (\mathrm{MPa\,s}^r)$.

The operator $\tilde{r}$ represents the order of the derivative. It is a fuzzy number between zero (linear elastic spring) and one (dashpot). Here, it is defined as a fuzzy number with triangular shape $\tilde{r} = \langle 0.13,\ 0.14,\ 0.15 \rangle$).

The fractional differential equation (54) is solved by the Laplace transform. The strain boundary condition $\tilde{\varepsilon}(\tau) = \tilde{\varepsilon}^*$ is used to obtain the relaxation function of the fuzzy fractional Newton element. Convolution of the relaxation function and time step discretization of the fuzzy strain process (equidistant time steps $\Delta\tau$) lead to

$$ {}^{[n]}\tilde{\sigma} = \sum_{i=1}^{n} \left\{ \frac{p \cdot {}^{[i]}\Delta\tilde{\varepsilon}}{\Gamma(2-\tilde{r}) \cdot \Delta\tau^{\tilde{r}}} \left[(n+1-i)^{(1-\tilde{r})} - (n-i)^{(1-\tilde{r})} \right] \right\} . \tag{55} $$

It can be seen, that the stress in time step $[n]$ dependents on the current strain and the whole strain history. Eq. (55) is utilized to verify the presented recurrent neural network approach. Training and validation patterns are computed by solving Eq. (55) within a fuzzy analysis (FA) (α-level optimization (Möller et al., 2000)). The time step length $\Delta\tau = 100$ s is chosen. Three α-cuts ($\alpha_1 = 0$, $\alpha_2 = 0.5$ and $\alpha_3 = 1$) are evaluated. The same fuzzy stress and fuzzy strain processes as presented in (Freitag et al., 2010b) are utilized, see Figures 7 to 10.

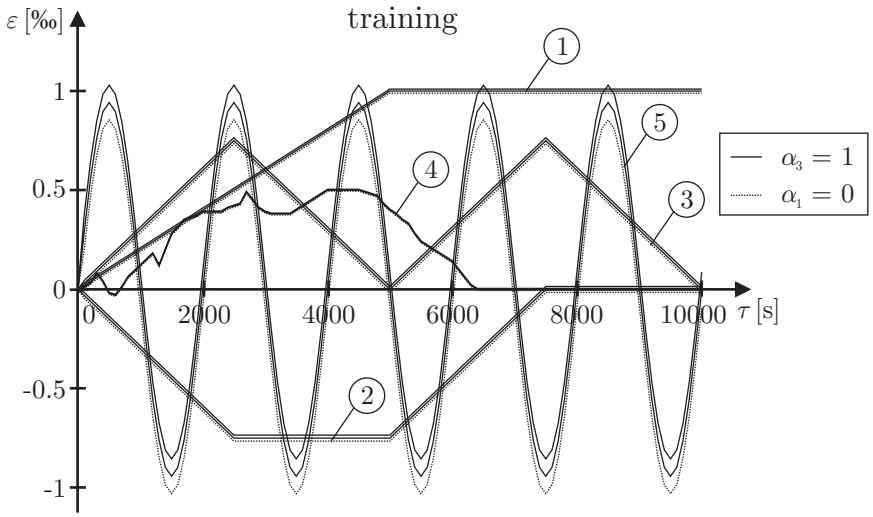

Figure 7. Fuzzy strain processes for network training.

The five fuzzy strain processes and the corresponding five fuzzy stress processes plotted in Figures 7 and 8 are used to train a recurrent neural network for Type 3 mapping ($\tilde{\varepsilon}(\tau) \tilde{\mapsto} \tilde{\sigma}(\tau)$). Nonlinear activation functions in the form of the area hyperbolic sine (arsinh) are used in the hidden neurons and a linear activation function is used in the output neuron. The signals of the recurrent neural network are computed by interval arithmetic operations for each α-cut. The developed PSO approach for fuzzy numbers is applied to identify the fuzzy network parameters. The number of particles is selected as $I = 20$. The control parameters ($c_1 = c_2 = 1.494$ and $c_3 = 0.729$) are defined according to (Eberhart and Shi, 2001). The training results of the recurrent neural network (RNN) are shown in Figure 8.

The five additional fuzzy strain processes in Figure 9 are used to validate the identified uncertain stress-strain-time dependency. The recurrent neural network predictions show a very good agreement with the desired responses obtained by a fuzzy analysis using Eq. (55), see Figure 10.

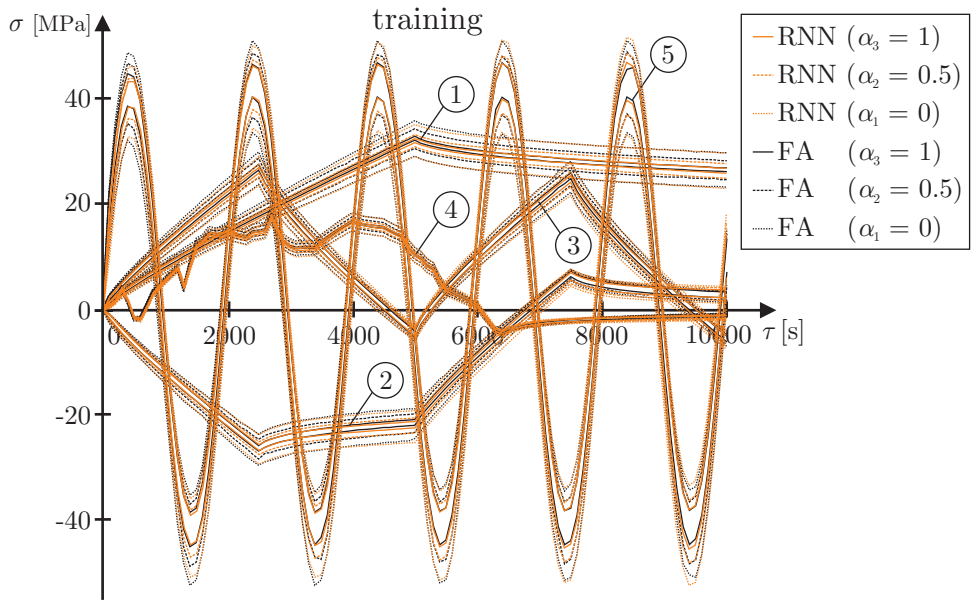

Figure 8. Fuzzy stress processes for network training.

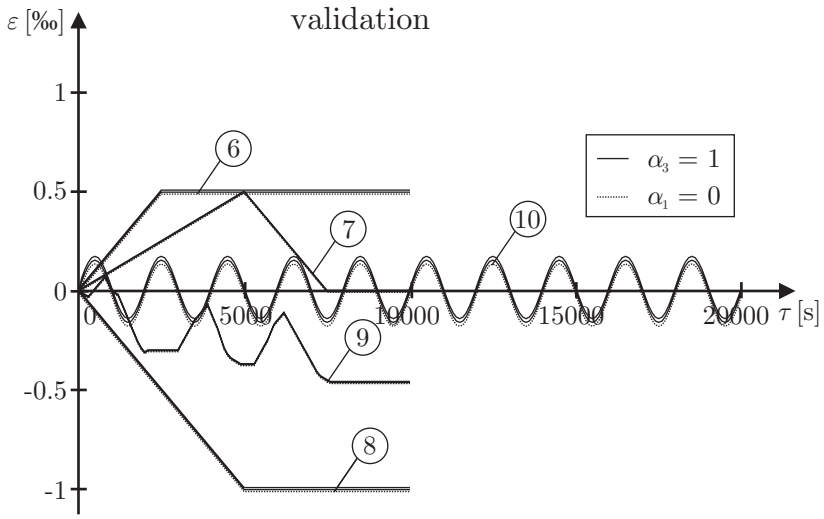

Figure 9. Fuzzy strain processes for network validation.

The same quality is achieved in comparison with the results in (Freitag et al., 2010b), where a backpropagation training algorithm has been applied. But here, a recurrent neural network with three hidden neurons and four context neurons ($1-3-1$ architecture) was sufficient for PSO training, whereas three hidden layers with 13 hidden and 14 context neurons in total ($1-5-5-3-1$ architecture) were required for backpropagation training.

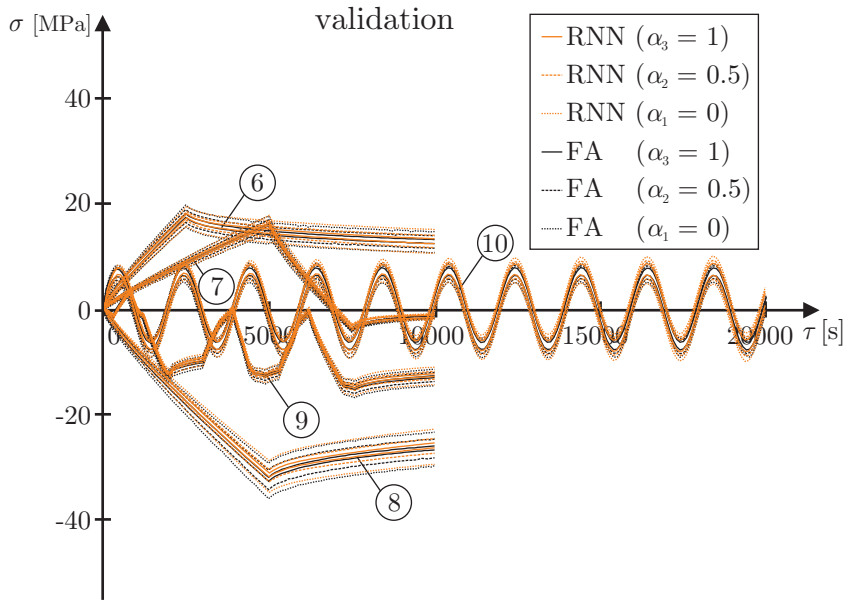

Figure 10. Fuzzy stress processes for network validation.

6.2. VERIFICATION WITH 3D MATERIAL MODEL

The proposed strategy for symmetric and decoupled stiffness is verified by a 3D linear elastic material model. Here, results for the mapping of deterministic strain processes onto deterministic stress processes are presented (special case of Type 1 mapping). The modulus of elasticity $E = 210\,000\,\mathrm{MPa}$ and Poisson's ratio $\nu = 0.2$ lead to the deterministic tangential stiffness matrix

$$\mathbf{C} = \begin{bmatrix} 233333 & 58333 & 58333 & 0 & 0 & 0 \\ 58333 & 233333 & 58333 & 0 & 0 & 0 \\ 58333 & 58333 & 233333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 87500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 87500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 87500 \end{bmatrix} \mathrm{MPa}\ . \tag{56}$$

Deterministic stress and strain processes (two patterns with $N = 1000$ time steps each) are used to train and validate a recurrent neural network with one hidden layer comprising 6 neurons ($6 - 6 - 6$ architecture). Linear activation functions are used in the output neurons and nonlinear activation functions in the form of the area hyperbolic sine are used in the hidden neurons. The network has 12 context neurons to consider possible history dependencies in the data series. However, the time-independent mapping of the strain vector ${}^{[n]}\varepsilon$ onto the stress vector ${}^{[n]}\sigma$ should be learned by the recurrent neural network.

The discussed PSO approach is applied to identify the deterministic network parameters. The same number of particles ($I = 20$) and values for the constants $c_1 = c_2 = 1.494$ and $c_3 = 0.729$ are used as in the previous example. The symmetry condition of Eq. (52) is used to get a symmetric tangential stiffness matrix, which is not possible by applying backpropagation training algorithms, see (Freitag et al., 2011b).

The training pattern has been presented 10^5 times to the network to identify the linearity and the time-independence between the strain and stress processes. As a result, the tangential stiffness matrix of the training pattern (Tr)

$$\mathbf{C}^{Tr} = \begin{bmatrix} 232826 & 58225 & 58191 & 0 & 0 & 0 \\ 58225 & 232769 & 58211 & 0 & 0 & 0 \\ 58191 & 58211 & 232743 & 0 & 0 & 0 \\ 0 & 0 & 0 & 87373 & 0 & 0 \\ 0 & 0 & 0 & 0 & 87372 & 0 \\ 0 & 0 & 0 & 0 & 0 & 87369 \end{bmatrix} \text{MPa} \tag{57}$$

is obtained, which contains the mean values of the partial derivatives of the stress components with respect to the strain components considering all 1000 time steps.

The network prediction has been verified with a second pattern comprising $N = 1000$ time steps, too. The mean values of the partial derivatives of the stress components with respect to the strain components of the validation pattern (V) are summarized as

$$\mathbf{C}^{V} = \begin{bmatrix} 232911 & 58247 & 58210 & 0 & 0 & 0 \\ 58247 & 232856 & 58232 & 0 & 0 & 0 \\ 58210 & 58232 & 232824 & 0 & 0 & 0 \\ 0 & 0 & 0 & 87385 & 0 & 0 \\ 0 & 0 & 0 & 0 & 87381 & 0 \\ 0 & 0 & 0 & 0 & 0 & 87364 \end{bmatrix} \text{MPa} \, . \tag{58}$$

The error is less than 0.25% for all components of the tangential stiffness matrix computed with the training and the validation patterns. In future works, symmetric network structures will also be applied to describe uncertain stress-strain-time dependencies.

7. Conclusion

In this paper, a new training strategy for artificial neural networks is presented. It is based on swarm intelligence. PSO approaches for interval and fuzzy numbers are developed accounting for uncertainty in measurements. These approaches have the flexibility of modifying all parameters during training of recurrent neural networks. Additionally, special network structures can be created, which is important for using neural networks as constitutive models. An application for time-dependent material behavior is presented. Results of verifications with a fuzzy fractional Newton element and a 3D linear elastic material model show high approximation quality of the developed neural network approaches. The new approaches can be applied to measured interval and fuzzy data. Recurrent neural networks for uncertain data can be utilized as constitutive models within interval, fuzzy, and fuzzy stochastic finite element analyses.

References

Adeli, H. Neural Networks in Civil Engineering: 1989-2000. *Computer-Aided Civil and Infrastructure Engineering*, 16:126–142, 2001.

Eberhart, R. C. and Y. Shi. Particle Swarm Optimization: Developments, Applications and Resources. In *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, pp. 81–86, Seoul, 2001.

Fontan, M., A. Ndiaye, D. Breysse, F. Bos, and C. Fernandez. Soil–structure interaction: Parameters identification using particle swarm optimization. *Computers and Structures*, 89(17-18):1602-1614, 2011.

Freitag, S. *Modellfreie numerische Prognosemethoden zur Tragwerksanalyse*. Veröffentlichungen – Institut für Statik und Dynamik der Tragwerke, Heft 19, Technische Universität Dresden, 2010.

Freitag, S., W. Graf, and M. Kaliske. Identification and prediction of time-dependent structural behavior with recurrent neural networks for uncertain data. In M. Beer, R. L. Muhanna, and R. L. Mullen, editors, *Proceedings of the 4th International Workshop on Reliable Engineering Computing*, pp. 577–596, Singapore, 2010. Research Publishing Services, Singapore.

Freitag, S., W. Graf, and M. Kaliske. Prediction of Time-Dependent Structural Responses with Recurrent Neural Networks. *Proceedings in Applied Mathematics and Mechanics*, 10:155–156, 2010.

Freitag, S., W. Graf, and M. Kaliske. Recurrent Neural Networks for Fuzzy Data. *Integrated Computer-Aided Engineering*, 18(3):265–280, 2011.

Freitag, S., W. Graf, and M. Kaliske. Recurrent Neural Networks for Fuzzy Data as a Material Description within the Finite Element Method. In Y. Tsompanakis and B. H. V. Topping, editors, *Proceedings of the Second International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*, paper 28, Chania, 2011. Civil-Comp Press, Stirlingshire.

Freitag, S., W. Graf, M. Kaliske, and J.-U. Sickert. Prediction of time-dependent structural behaviour with recurrent neural networks for fuzzy data. *Computers and Structures*, 89(21-22):1971–1981, 2011.

Graf, W., S. Freitag, M. Kaliske, and J.-U. Sickert. Recurrent neural networks for uncertain time-dependent structural behavior. *Computer-Aided Civil and Infrastructure Engineering*, 25(5):322–333, 2010.

Graf, W., J.-U. Sickert, S. Freitag, S. Pannier, and M. Kaliske. Neural Network Approaches in Structural Analysis under Consideration of Imprecision and Variability. In Y. Tsompanakis and B. H. V. Topping, editors, *Soft Computing Methods for Civil and Structural Engineering*, Chapter 4, pp. 59–85. Saxe-Coburg Publications, Stirlingshire, 2011.

Haykin, S. *Neural Networks – A Comprehensive Foundation*. Prentice-Hall, Upper Saddle River, 1999.

Kennedy, J. and R. C. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks IV*, pp. 1942–1948. IEEE, Piscataway, 1995.

Kennedy, J., R. C. Eberhart, and S. Yuhui. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, 2001.

Kuok, K. K., S. Harun, and S. M. Shamsuddin. Particle swarm optimization feedforward neural network for modeling runoff. *International Journal of Environmental Science and Technology*, 7(1):67–78, 2010.

Li, L. J., Z. B. Huang, F. Liu, and Q. H. Wu. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures*, 85:340–349, 2007.

Mendes, R., P. Cortez, M. Rocha, and J. Neves. Particle Swarms for Feedforward Neural Network Training. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 1895–1899. IEEE, Honolulu, 2002.

Möller, B. and M. Beer. Engineering computation under uncertainty – Capabilities of non-traditional models. *Computers and Structures*, 86(10):1024–1041, 2008.

Möller, B., W. Graf, and M. Beer. Fuzzy structural analysis using α-level optimization. *Computational Mechanics*, 26(6):547–565, 2000.

Moens, D. and D. Vandepitte. A survey of non-probabilistic uncertainty treatment in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 194:1527–1555, 2005.

Moore, R. E. *Methods and Applications of Interval Analysis*. SIAM, Studies in Applied Mathematics, 2, Philadelphia, 1979.

Muhanna, R. L., H. Zhang, and R. L. Mullen. Interval Finite Elements as a Basis for Generalized Models of Uncertainty in Engineering. *Reliable Computing*, 13(2):173–194, 2007.

Oeser, M. and S. Freitag. Modeling of materials with fading memory using neural networks. *International Journal for Numerical Methods in Engineering*, 78(7):843–862, 2009.

Perez, R. E. and K. Behdinan. Particle swarm approach for structural design optimization. *Computers and Structures*, 85:1579–1588, 2007.

Rao, M. V. R., R. L. Mullen, and R. L. Muhanna. A new interval finite element formulation with the same accuracy in primary and derived variables. *International Journal for Reliability and Safety*, 5(3/4):336–357, 2011.

Sickert, J.-U., S. Freitag, and W. Graf. Prediction of uncertain structural behaviour and robust design. *International Journal for Reliability and Safety*, 5(3/4):358–377, 2011.

Zhang, J.-R., J. Zhang, T.-M. Lok, and M. R. Lyu. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied Mathematics and Computation*, 185:1026–1037, 2007.