WARSAW UNIVERSITY OF TECHNOLOGY Index 351733
FACULTY OF CIVIL ENGINEERING
COMMITTEE FOR CIVIL AND WATER ENGINEERING
POLISH ACADEMY OF SCIENCES ISSN 1230-2945
DOI: 10.24425/ace.2022.143033
ARCHIVES OF CIVIL ENGINEERING
Vol. LXVIII ISSUE 4 2022
pp. 179 –197

Research paper

Multi-scale modelling of brick masonry using a numerical homogenisation technique and an artificial neural network

Aleksander Urbański[1], Szymon Ligęza[2], Marcin Drabczyk[3]

Abstract: A new method of creating constitutive model of masonry is reported in this work. The model is not an explicit orthotropic elastic-plastic one, but with an artificial neural network (ANN) giving an implicit constitutive function. It relates the new state of generalised stresses Σ^{n+1} with the old state Σ^{n} and with an increment of generalised strains ΔE (plane-stress conditions are assumed). The first step is to run a strain- controlled homogenisation, repeatedly, on a three-dimensional finite element model of a periodic cell, with elastic-plastic models (Drucker–Prager) of the components; thus a set of paths is created in $(\Sigma, \Delta E)$ space. From these paths, a set of patterns is formed to train the ANN. A description of how to prepare these data and a discussion on ANN training issues are presented. Finally, the procedure based on trained ANN is put into a finite-element code as a constitutive function. This enables the analysis of arbitrarily large masonry systems. The approach is verified by comparing the results of the developed model basing on ANN with a direct (single-scale) one, which showed acceptable accuracy.

Keywords: artificial neural network, finite element method, homogenisation, masonry

[1]Dsc., Phd., Eng., Cracow University of Technology, Faculty of Civil Engineering, ul. Warszawska 24, 31-155, Kraków, Poland, e-mail: aurbansk123@gmail.com, ORCID: 0000-0002-5544-9134
[2]Msc., Eng., AGH University of Science and Technology, Faculty of Drilling, Oil and Gas (doctoral student), al. Mickiewicza 30, 30-059 Kraków, Poland, e-mail: sligeza@yandex.com, ORCID: 0000-0001-6702-9260
[3]Msc., Eng., Idealogic Ltd., ul. Kapelanka 26, 30-347 Kraków, Poland, e-mail: marcindrb@gmail.com, ORCID: 0000-0001-9245-1025

1. Introduction

In this paper, the report of the successful attempt to model masonry structures using finite-element method (FEM), with an artificial neural network (ANN) driver acting as a constitutive procedure, is given. Brick masonry is an example of a medium possessing a periodic microstructure, while its components (brick and mortar) have highly nonlinear material properties. The direct finite element modelling of such media is limited to only small structural details. Larger structural systems require the introduction of an equivalent, homogenised material model.

In the last 30 years, different researchers have made many attempts to create such a model, see [1, 2], where state-of-the art reports may be found. However, owing to the complexity of the problem, there is no commonly accepted one. Thus, the main concern of the authors was to recognize practical possibility of building constitutive models of wider class of masonry-like media by ANN, but not a detailed investigation of specific masonry behaviour.

In this study, all modelling stages were performed using the Hybrid FEM-ANN (HFE-MANN), to run finite element computations at micro- and macro-levels. The AppANN was used to train and optimise topology of the artificial neural network. Both are original software products developed by Idealogic Ltd.

2. Theory / calculation

2.1. General concept of multi-scale masonry modelling using artificial neural networks

In this work the masonry structure modelling is limited to the plane-stress conditions. This is the simplest choice for masonry, as indicated by many previous studies [3–5]. The desired model is in the form given by Eq. (2.1), consistent with an incremental solution algorithm for any problem with a strain-driven nonlinear constitutive relation:

$$\boldsymbol{\Sigma}^{i+1} = F\left(\boldsymbol{\Sigma}^{i},\, \Delta E\right) \tag{2.1}$$

where $\boldsymbol{\Sigma}^{i} = \{\, N_{XX},\ \ N_{YY},\ \ N_{XY} \,\}^{T}$, $\boldsymbol{\Sigma}^{i+1}$ are membrane forces (macro-stresses) in the subsequent instants $i, i+1$, related to an $\{X, Y\}$ coordinate system in which the X direction coincides with the mortar layer $\Delta\mathbf{E} = \{\, \Delta E_{XX},\ \Delta E_{YY},\ \Delta\Gamma_{XY} \,\}^{T}$ are the corresponding average incremental strains.

Most of the in-plane macro-constitutive masonry models to be used in finite-element codes have the formal structure of an elastic-plastic model, with orthotropic elastic and plastic parts. The elastic part of the model can be described by approximate, simplified relations [6, 7] or by exact formulae obtained by homogenisation [8]. In plastic masonry behaviour, the plasticity condition, as well as the flow-rule potential, is not a function of the stress invariants, but of all macro-stress components, [1, 9]. Other model enhancements, e.g. a description of softening or damage behaviour can also be introduced, considering the

orthotropy of the media [10]. All of these factors are sources of difficulty when creating a strictly empirical data set for the given model, as a full identification would require multiple runs up to and including damaging the sample.

Another method of creating such a model, without assuming its particular form, which will be examined later in this paper, is to build an artificial neural network, trained on a large numerically generated set of paths in the macro-stress-strain space. Obviously, this idea is a general one, and has been used by other researchers with similar problems [11–18]. Nowadays, ANN are successfully used in other area of civil enginering, like shown in [19]. The theoretical background for this is provided by the mathematical theory of artificial neural networks, with the universal approximation theorem [20,21]. It has been proven that a multilayered, unidirectional perceptron neural network is capable of approximating an arbitrary multi-dimensional nonlinear function e.g. $\mathbf{\Phi}\colon \mathbf{X} \in R^n \rightarrow \mathbf{Y} \in R^m$. The artificial neural network used in our attempts at masonry modelling is shown schematically in Figure 2.

Obviously, the optimal artificial neural network parameters for any masonry model must be established, e.g. the number of layers and neurons in each, the neuron activation function and the weight and bias coefficient values. This will be briefly described in Section 2.3.

The complete approach to numerically analyse masonry structures using an artificial neural network as the constitutive driver is shown in Figure 1. To verify the approach, the obtained results are compared with direct (single-scale) model.

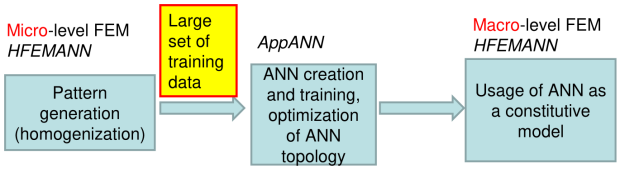

Fig. 1. Application schema for using an artificial neural network in a FE analysis

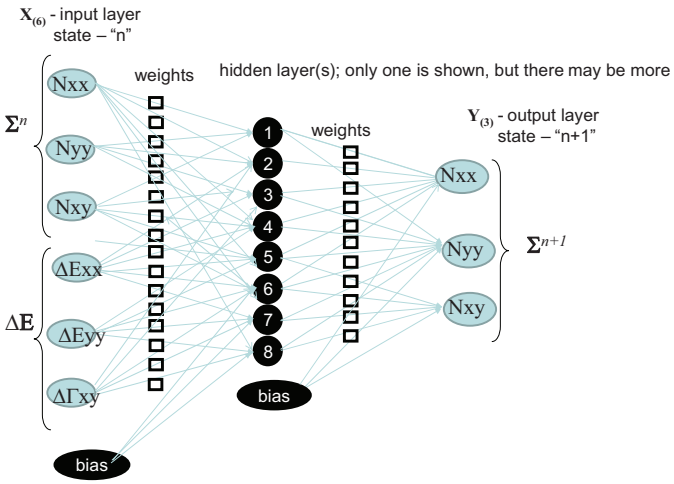

Fig. 2. Artificial neural network of multilayer perceptron type as a constitutive function for masonry

2.2. Micro-level masonry analysis

To gather data needed by an identification procedure of the masonry constitutive macro-model, i.e. the set of relations between the membrane forces and in-plane strains, a numerical homogenisation analysis is performed on a periodic, repetitive, three-dimensional (3D) masonry cell. It is worth to note that in similar circumstances Drosopoulos et al. [16] perform homogenization on 2D plane stress masonry cell. The starting data are the cell morphology and the constitutive properties of the components. The technique used is a generalised homogenisation with applied strain-control. Then, the total strain fields are decomposed into the control strains $\mathbf{E}(\lambda)$, which are given as constant within the periodic cell, with λ – the pseudo-time parameter. The periodic fluctuation of the strain field $\boldsymbol{\varepsilon}^p$ are the result of the microstructure:

$$\boldsymbol{\varepsilon}(\mathbf{x}) = L\mathbf{E}(\lambda) + \boldsymbol{\varepsilon}^p(\mathbf{x}) \tag{2.2}$$

The fully 3D fluctuation of the strain field $\boldsymbol{\varepsilon}^p$ is related to the periodic displacement field $\boldsymbol{\varepsilon}^p(\mathbf{x}) = B\mathbf{u}^p(\mathbf{x})$ where B is the differential operator of Cauchy's compatibility equations (in matrix form). The displacement field $\mathbf{u}^p(\mathbf{x})$ is a primary unknown, which can be determined numerically from the equilibrium equations:

$$B^T\boldsymbol{\sigma}\left(L\mathbf{E}(\lambda) + B\mathbf{u}^p(\mathbf{x})\right) = \mathbf{0} \tag{2.3}$$

The details of the incremental finite element formulation of the homogenisation with general material nonlinearity can be found in [8]. The output of the one run of the homogenisation procedure is a path in the space of generalised stresses and strains:

$$\lambda \rightarrow \{\mathbf{E}(\lambda), \boldsymbol{\Sigma}(\lambda)\}, \qquad \boldsymbol{\Sigma}(\lambda) = \frac{1}{S_C}\int_{V_C} L^T\boldsymbol{\sigma}\,\mathrm{d}V \tag{2.4}$$

In Eqs. (2.2)–(2.4), $L^T = [I_{3\times3}, 0_{3\times3}]$ is a 3×6 matrix selecting in-plane components, V_C is the periodic cell volume, and $S_C = a\cdot b$ is its in-plane area. Figure 3 shows a finite element model of a periodic masonry cell. To minimise the computational effort, only half of a schema is built. This is sufficient because symmetry of the cell and the imposed strains versus the XY plane. Obviously, in this case, the resulting membrane forces should be multiplied by a factor of two. It has a periodic boundary condition imposed on the opposite planes $x = \pm a/2$ and $y = \pm b/2$. The easiest implementation method is the following: given a list of nodal pairs, it is sufficient to modify the routine attaching the equation numbers to the nodes, such that each node j, which is coupled to node i by the periodicity relation, $j > i$, inherits the degrees of freedom (DOF) numbers already set for node i.

For both masonry components, i.e. brick and mortar, a Drucker–Prager (DP) elastic-plastic model is assumed. The DP model data, i.e. cohesion $c_{(.)}$ and friction angle $\phi_{(.)}$, are derived from the compressive $f_{c(.)}$ and tensile $f_{t(.)}$ strengths, being standard material data

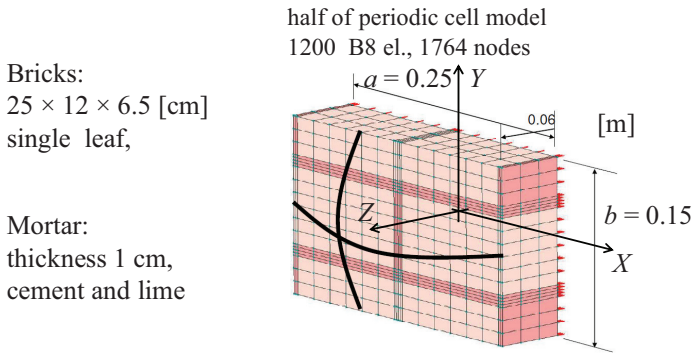

Fig. 3. FE model of a periodic cell

for ceramics. The formulae given in Eq. (2.5) are used:

$$\sin\phi_{(.)} = \frac{1 - f_{(.)}}{1 + f_{(.)}} \quad \Rightarrow \quad \phi_{(.)} = \arcsin\left(\frac{1 - f_{(.)}}{1 + f_{(.)}}\right), \tag{2.5}$$
$$c_{(.)} = f_{c(.)}\frac{1 - \sin\phi_{(.)}}{\cos\phi_{(.)}}$$

In the considered example, the data roughly correspond to masonry built of class B10 bricks and lime-cement mortar. The specific values are as follows:

- Brick: $E_b = 9000$ MPa, $\nu_b = 0.167$, $f_{cb} = 7.5$ MPa, $f_{tb} = 0.75$ MPa ($f_b = f_{tb}/f_{cb} = 0.10$),
- Mortar: $E_m = 900$ MPa, $\nu_m = 0.167$, $f_{cm} = 0.75$ MPa, $f_{tm} = 0.075$ MPa ($f_m = f_{tm}/f_{cm} = 0.10$).

The exemplary results of the micro-level homogenisation analysis are shown in Figure 4. The stress level (SL) is understood as the relative distance, in the radial (deviatoric) direction, of a stress point from the yield surface.

Note, that from the viewpoint of the effectiveness of the entire three-step procedure (in Figure 1), important is a way of creating large data sets to run homogenisation analysis repeatedly, with different imposed strains. As the first step a range of each component of the strains $\mathbf{E}$ should be established, as physically justified. In a considered case, the cube in the strain space, with $E_0 = 0.01$, is set: $\{-E_0 \le E_{XX} \le E_0, -E_0 \le E_{YY} \le E_0, 0 \le \Gamma_{XY} \le E_0\}$. Assuming a division in each direction at M intervals, starting from 0 to point $\{E_{XXi}, E_{YYj}, \Gamma_{XYk}\}$, $NPATH = 4 \cdot 20 \cdot 11 = 880$ different paths are created:

$$\begin{aligned} E_{XXi} &= E_0 \cdot f_i(\lambda), \quad i = 1, \ldots, M = 20 \\ E_{YYj} &= E_0 \cdot f_j(\lambda), \quad j = 1, \ldots, M = 20 \\ \Gamma_{XYk} &= E_0 \cdot f_k(\lambda), \quad k = 1, \ldots, M = 10 \end{aligned} \tag{2.6}$$

Two types of control function $f(\lambda)$ are applied:

1. Smooth, linear: $f_i(\lambda) = A_i\lambda$, $A_i = 2m/M$, $m = \{0, 1, \ldots, M/2\} \cup \{-1, \ldots, -M/2\}$ for E_{XX}, E_{YY}, for Γ_{XY} : $A_i = m/M$, $m = \{0, 1, \ldots, M\}$; and $\lambda = (0, 1)$.

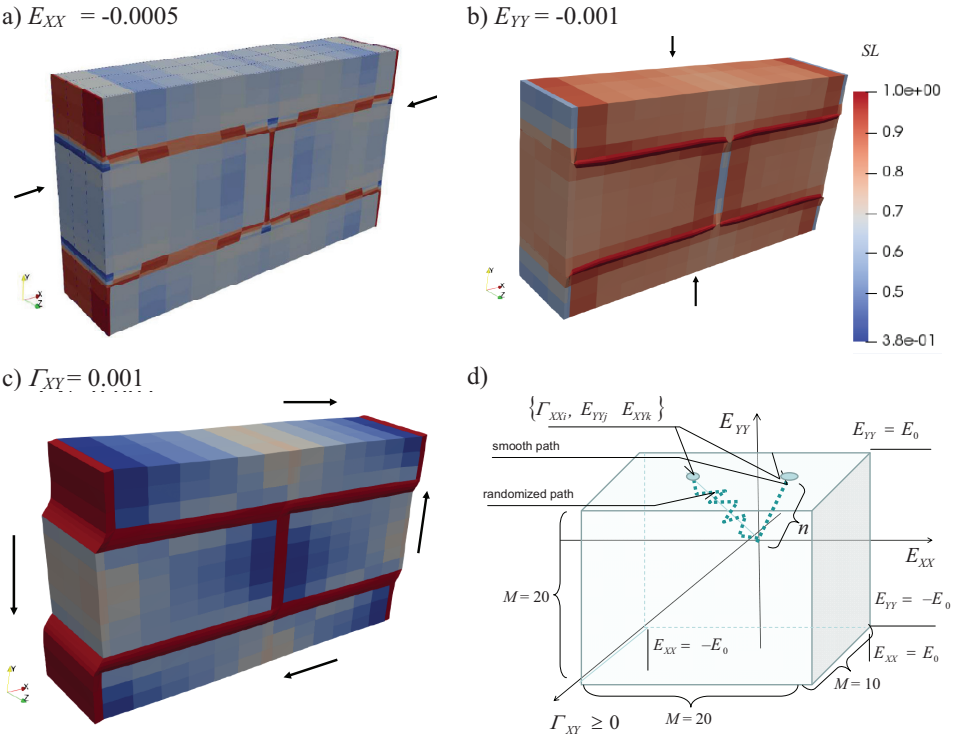

Fig. 4. Stress level and deformation for imposed strains (a)–(c). Visualisation of imposed strain space (d)

2. Noisy, randomised (Figure 5): $f_i^{RAND}(\lambda) = \overline{f}_i(\lambda) \cdot RAND(\alpha, \beta)$, with $\overline{f}_i(\lambda) = f_i(\lambda)$ as in 1), $\alpha = 0.5$ and $\beta = 1.5$.

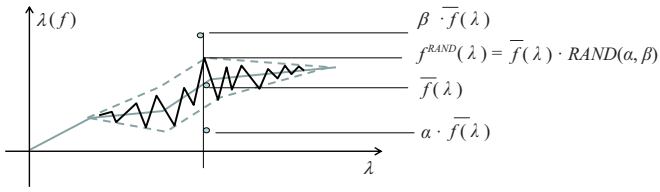

Fig. 5. Randomised control function

For each of the path divisions, $NSTEP = 20$ equal increments of the imposed strains were performed. Note that to minimise the computational effort, the results obtained for positive values $\Gamma_{XY} > 0$ are taken also for imposed shear strains $\Gamma_{XY} < 0$, with $N_{XY}^{\text{neg}} = -N_{XY}^{\text{pos}}$; however the normal forces are preserved: $N_{XX}^{\text{neg}} = N_{XX}^{\text{pos}}$, $N_{YY}^{\text{neg}} = N_{YY}^{\text{pos}}$. This is because the normal forces accompanying the shear emerge due to the dilatancy effect and are independent of the shear direction. Each converged step generates one record,

including a pattern for training the ANN which consists of input $\mathbf{X} = \{\boldsymbol{\Sigma}^{n-1}, \Delta\mathbf{E}^n\}$ and output $\mathbf{Y} = \{\boldsymbol{\Sigma}^n\}$ vectors. Using $2 \cdot 2 \cdot 880 = 3520$ paths, 65536 records were obtained; this was because not for all paths a converged solution in the full range of λ was reached. The entire pattern generation lasted about 8 h on a PC with an i7 processor.

In the procedure described above, the number of **X**, **Y** patterns obtained from one path is equal to the number N of macro stress-strain points in it, $\{\mathbf{E}(\lambda_i), \boldsymbol{\Sigma}(\lambda_i)\}$, $i = 0, N$, leading to:

$$\mathbf{X} = \left\{\boldsymbol{\Sigma}^{i-1}, \Delta\mathbf{E}^i = \mathbf{E}(\lambda_i) - \mathbf{E}(\lambda_{i-1})\right\}, \qquad \mathbf{Y} = \left\{\boldsymbol{\Sigma}^i\right\}, \quad i = 1, N. \tag{2.7}$$

The number of patterns might substantially increase, up to $Nmax = N(N-1)/2$, when for any given stress point, a larger set of strains increments and the corresponding stresses are included:

$$\mathbf{X} = \left\{\boldsymbol{\Sigma}^{i-1},\ \Delta\mathbf{E}^i_j = \mathbf{E}(\lambda_j) - \mathbf{E}(\lambda_{i-1})\right\}, \qquad \mathbf{Y} = \left\{\boldsymbol{\Sigma}^j\right\}, \quad i = 1, N, \quad j = i+1, N \tag{2.8}$$

This procedure, called "diluting", is shown in Figure 6. In addition to increasing the number of patterns, it widens the range of the strain

increments. It may only be performed for smooth paths and with controlled intensity. This is essential when using a trained ANN as the constitutive function in a boundary value problem; however, it does not require any additional computational effort. In the presented example procedure of diluting was performed. The number of records finally used to train the artificial neural network was about 250,000.

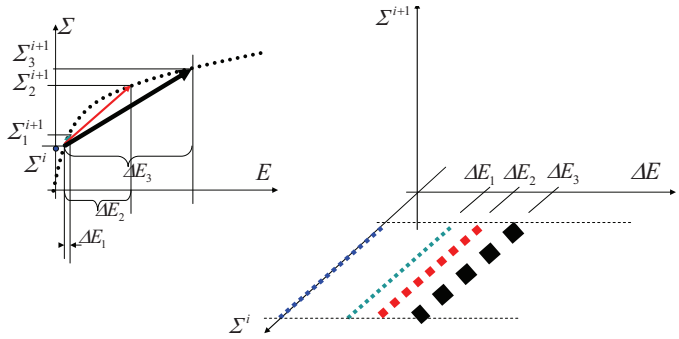

Fig. 6. Increasing the number of patterns and widening the range of strain increments by diluting procedure

Figure 7 shows comparison of the exemplary paths obtained from the finite element homogenisation, with their approximation done by the artificial neural network trained with the procedures described in Section 2.3 It considers both smooth (left) and randomised (right) path cases.

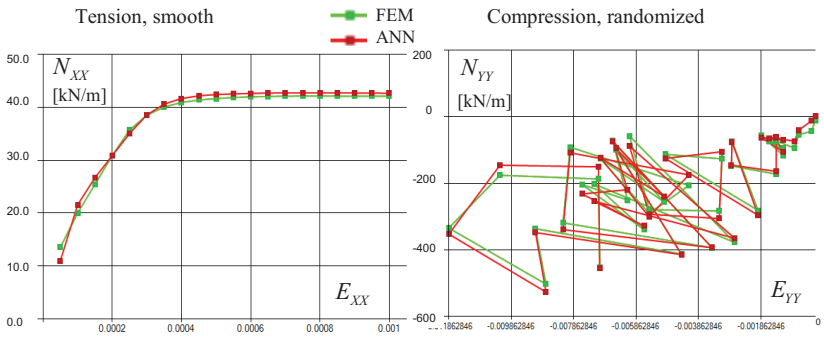

Fig. 7. Paths obtained by the finite element homogenisation and their prediction by the artificial neural network

2.3. Identification of a macro-level constitutive model using an artificial neural network

In the following section, the functionality of a developed application, AppANN, is briefly described. This app allows a user with only basic knowledge of artificial neural networks technology, efficiently create an artificial neural networks to be implemented in an finite element code. The multi-layered, unidirectional perceptron neural net (shown schematically in Figure 3) is a nonlinear function $\mathbf{\Phi}\colon \mathbf{X} \in R^N \rightarrow \mathbf{Y} \in R^M$, defined recursively as:

$$
\begin{aligned}
&\mathbf{\Phi}\colon R^N \ni \mathbf{X} = \{x_i\} = \left\{y_i^{(0)}\right\}, \quad i = 1, n^{(0)}, \quad n^{(0)} = N \\
&y_i^{(k)} = \alpha^{(k)}\left(\sum_{j=1,n^{(k-1)}} w_j^{(k-1)} y_j^{(k-1)} + b^{(k-1)}\right), \quad k = 1, L. \\
&\left\{y_i^L\right\} = \mathbf{Y} \in R^M, \quad i = 1, n^L, \quad n^L = M
\end{aligned}
\tag{2.9}
$$

In Eq. (2.9), $L > 1$ is the number of layers, and $n^{(k)}$ is the number of neurons in the k-th layer. These parameters define the topology of the artificial neural net. $w_j^{(k)}$ are the coefficients of the linear combination named as the weights, and $b^{(k)}$ is the bias, which is a term that shifts values of linear combinations. $\alpha^{(k)}$ is the activation function, which is generally nonlinear. The activation functions available in our implementation, are presented in Table 1.

Table 1. Activation functions available in AppANN

Name	Linear	Sigmoid	Tanh	Softplus	Softsign	Relu
$\alpha(x) =$	x	$\dfrac{1}{1+e^{-x}}$	$\tanh(x)$	$\ln(1+e^x)$	$\dfrac{x}{1+\lvert x\rvert}$	$\begin{matrix}0, & x<0\\ x, & x\geq 0\end{matrix}$

For an assumed set of topological parameters L and $n^{(k)}$, the values of $w_j^{(k)}$ and $b^{(k)}$ are the results of using an objective loss function to minimise error. The available loss functions (with their customary names and related acronyms) are listed below the **Loss function** item. The *back propagation* algorithm is used for those optimisation procedures. It is described in numerous references, e.g. [21]. Using the AppANN application developed in this project, one can control the creation of an artificial neural network by simply using a dialog window. When in **Manual** mode, for the one run of training an ANN, the following options (hyper-parameters), are settable, see Figure 8:

Technology selection. The options are *TensorFlow* or *Keras-TensorFlow*.

TensorFlow – is an open-source low-level machine-learning programming library, written by the Google Brain Team for machine-learning developments (C++) [22].

Keras – is the front-end overlay that provides a high-level interface to *TensorFlow*. It enables the rapid prototyping, configuration, and implementation of neural networks [22]. It works very well with the optimisation library used in the *Hyperopt* project (Python).

Learning rate. It is a step size of decline used by gradient descent algorithm in optimization of the neural network weights. The *Learning rate* is an important parameter, and its proper selection is crucial for the optimizer to find minima in multidimensional spaces. A too small step causes a very slow neural networks training, while too-large step might jump over local minima, thereby decreasing the prediction quality.

The operation of the gradient-optimiser algorithm can be compared to going downhill the shortest possible path to reach the local or global minimum of the cost function. After

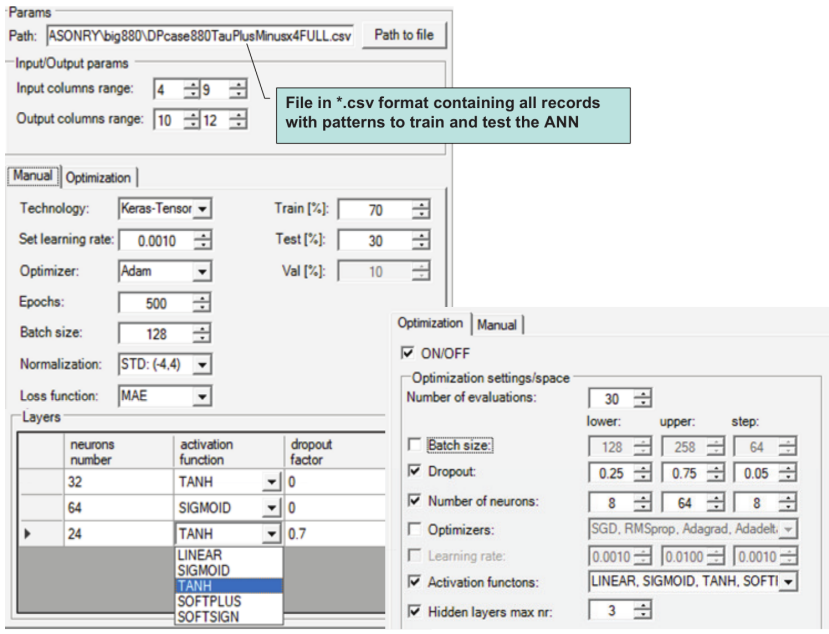

Fig. 8. Dialogs enabling the user to set hyper-parameters in manual and optimization mode

each epoch, the error value between the real and predicted data is calculated by means of *back propagation*. Then, the values of the weights and biases in the neural units are rectified. The magnitude of the "*Learning rate*" depends on the selected optimiser.

Optimiser. The user can choose between the following optimisers: Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop), Adaptive Gradient (Adagrad), and Adaptive Delta (Adadelta). For their descriptions see [24].

Number of Epochs. With each subsequent epoch, the prediction result is compared with the actual result. The magnitude of the error between these values is used to correct the weights in the neural network, which are updated in the next epoch. Hence, the prediction accuracy increases with the number of epochs. After exceeding a certain quantity (unknown in advance), the prediction quality ceases to improve, and the risk of over-fitting increases. Then, the so called "early stopping" method can be used. It interrupts the network-training process when, after a predetermined number of successive epochs, the loss-function or success metrics improvement is less than the assumed minimum increase. The model learning time directly depends on this factor.

Batch Size. This specifies the number of samples from the training data that will pass through the network simultaneously (in parallel). The algorithm divides the amount of data by the batch size, thereby obtaining the number of iterations per epoch. Each iteration uses a different portion of the previously divided data until the entire training set has passed through the network. This marks the end of one epoch.

The size of one data portion should be dependent on the computing power. The larger the latter is, the more able the network is to process a larger portion of data in parallel. Thus it speeds up the calculation process at the expense of a decrease in quality.

Normalisation. Optimisation algorithms require the normalisation or standardisation of data to eliminate the "gradient explosion" effect. This is because each column represents a dimension with a scale that is generally different from the scale of the other dimensions. Hence, all data columns should first be transformed to have the same scale. The user selects between the following transformation types.

Normalisation: Knowing the minimum and maximum values in the i-th column, taken from all of the training-data records, we can replace them with values from the selected range; e.g. 0 to 1. To do this, we simply scale them linearly according to the following formula: $z_i = (x_i - \min(x_i))/(\max(x_i) - \min(x_i))$. A characteristic feature of min-max normalisation is the preservation of the original data distribution. Therefore, this method is not suitable for limiting the impact of outliers [23].

Standardization: The purpose of the method is to change the distribution of data to obtain a distribution with an average value of 0 and a standard deviation of 1. After standardisation, values below the average will be negative, and values above the average will be positive. As a rule, standardised values range from –4 to 4. The standardised value is calculated from the following formula $z_i = (x_i - \overline{x}_i)/\sigma_i$, where $\overline{x}_i$ is the arithmetic mean and σ_i is the standard deviation of the variable [23]. Note, that the vectors of the values related to *normalisation* or *standardisation*, i.e. $(\max(x_i), \min(x_i), \overline{x}_i, \sigma_i, i = 1, N)$, established while creating the artificial neural network, must be stored. Each time when

the ANN is invoked (as the representing constitutive model), they will be used to perform normalise the input **X** and re-normalise the output **Y**.

Loss function. The available functions are means squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and coefficient of determination (R^2). Exact formulae can be found in numerous references, e.g. [20, 22].

Train/Test data splitting. The user must to decide the percentage of testing and training data to be used in the learning process. The default value is 70% for training and 30% for testing.

Configuration of hidden layers. First, the user must select the number of hidden layers, i.e. $L-1$. Then, for each individual layer, the following settings must be configured: number of neurons $n^{(k)}$, type of activation function $\alpha^{(k)}$, and dropout on individual layers. The number of layers and neurons depends on the number of interrelationships between the dimensions in the data and the complexity of these correlations.

Dropout factor. This factor takes a value from 0 to 1 and is one of the methods to combat model over-fitting. In practice, it means the given percentage of neurons is deactivated for a given epoch. The dropout's task is to stimulate other neurons, previously inactivated, which can improve the quality of the model. In the **Optimisation** mode an optimal setting of the hyper-parameters is sought. The most important is to optimize the artificial neural network topology. The main-program algorithm uses the *Hyperopt* library for this task. Its optimisation technique (*tree structured parzen estimator*) is described briefly in [24]. Figure 10 shows the dialog that controls the of hyper-parameter space.

After completing the optimization, the resulting data describing the artificial neural network are stored in a binary file, for further usage in the finite element program. Moreover, the user may trace the result by visualising of the minimisation process for different loss functions, see Figure 9, or by comparing the paths obtained as FE results and their prediction by the ANN, given in Figure 7.

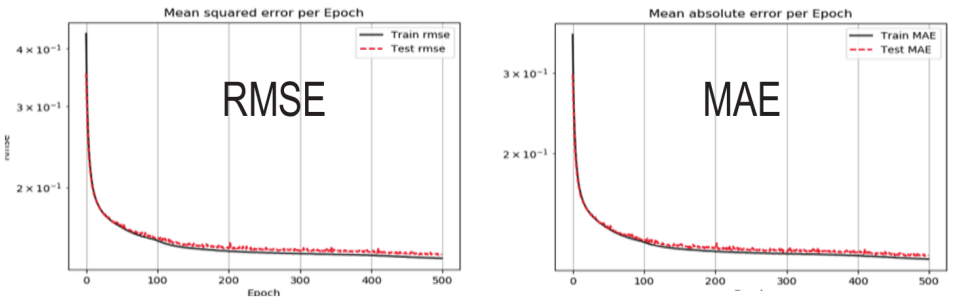

Fig. 9. Selected learning-process accuracy measures as a function of the number of epochs

Activated items give the user the opportunity to define the space of the optimised parameters, by setting the lower range limit, the upper range limit, the step (for numerical parameters), and by the specification for other items. It should be remembered that setting

too many parameters and value ranges will result in a huge multi-dimensional space, which the optimiser will have to search in to find a local minimum. This will increase the number of trial evaluations and thus the time it takes to find the optimal solution.

In the considered masonry example, with the hyper-parameter setup given in Figure 8 (left), one ANN training run lasted approximately 100 s. However searching for the optimal topology and the other items in the range specified in Figure 8 (right), overriding the *configuration of hidden layers* data, lasted about 8 h on an i7PC. The artificial neural network used successfully in the finite element analysis of a masonry was built with one hidden layer $L = 1$, and $n^{(1)} = 16$ neuron, with the RELU activation function.

2.4. Macro-level finite element analysis of masonry structures

In the macro-level finite element analysis of masonry structures, a few simplifying assumptions are made, in this work. They are as follows:

1. The masonry walls are modelled with planar shell elements.
2. At the level of a shell-element cross-section the membrane state is treated as uncoupled with the bending and transversal shear states. The membrane state is analysed using the ANN, as described in previous sections. For the latter, purely linear elastic cross-sectional models are assumed in the following form; i.e. for components with bending moments and curvature tensors:

$$\mathbf{M} = \mathbf{D}_B \cdot \boldsymbol{\kappa}, \qquad \boldsymbol{\kappa} = \begin{bmatrix} \kappa_{XX} \\ \kappa_{YY} \\ 2\kappa_{XY} \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} M_{XX} \\ M_{YY} \\ M_{XY} \end{bmatrix}$$

(2.10)
$$\mathbf{D}_B = D \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix}, \qquad D = \frac{Eh^3}{12\left(1-\nu^2\right)}$$

and for the shear forces and mean shear angles:

(2.11)
$$\mathbf{Q} = \mathbf{D}_S \cdot \boldsymbol{\beta}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_X \\ \beta_Y \end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix} Q_X \\ Q_Y \end{bmatrix}, \qquad \mathbf{D}_S = \frac{5}{6} Gh \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The above simplification is allowable in the analysis of masonry walls, being the part of a 3D structural system in a case where no significant transversal loads are applied. For the membrane state, the artificial neural network provides only a part of the constitutive model, i.e. the function giving the generalised stress response. The other needed part, the material stiffness matrix $\mathbf{D}_M$, is taken as initial, orthotropic, and linearly elastic. A module of the elastic orthotropic material can be easily obtained from a micro-level analysis of the periodic cell model, with linearly elastic components models. It is sufficient to perform three homogenisation runs with the imposed strains: $\mathbf{E}_{XX} = \{1,0,0\}^T$, $\mathbf{E}_{YY} = \{0,1,0\}^T$, $\boldsymbol{\Gamma}_{XY} = \{0,0,1\}^T$. The corresponding vectors of membrane forces $\boldsymbol{\Sigma}_{XX}$, $\boldsymbol{\Sigma}_{YY}$, and $\boldsymbol{\Sigma}_{XY}$ create the initial constitutive matrix of an orthotropic model, which for the considered

example, takes the following values:

$$\mathbf{D}_M = \begin{bmatrix} \boldsymbol{\Sigma}_{XX}^T \\ \boldsymbol{\Sigma}_{YY}^T \\ \boldsymbol{\Sigma}_{XY}^T \end{bmatrix} = \begin{bmatrix} 777.0 & 72.4 & 0 \\ 72.4 & 486.0 & 0 \\ 0 & 0 & 182.4 \end{bmatrix} \quad [\mathrm{MN/m}] \tag{2.12}$$

Although the described method of proceeding with the constitutive matrix of the membrane model has a limited choice of nonlinear problem-solution algorithms to the initial stiffness algorithm only, it is more convenient than the alternative. Attempts have been made to introduce a constitutive matrix approximately obtained by a finite differentiation of the artificial neural network, i.e.:

$$\mathbf{D}_{M\_ANN} = \frac{\Delta\boldsymbol{\Sigma}}{\Delta\mathbf{E}} = \left[\frac{N_i\left(\mathbf{E} + \{\Delta E_j\}^T\right) - N_i(\mathbf{E})}{\Delta E_j} \right], \quad i, j = XX, YY, XY \tag{2.13}$$

with small increments of $\Delta E_j = \varepsilon \cdot \|\Delta\mathbf{E}\|$, where $\varepsilon = 0.01$. However, it turns out to be inefficient because of the numerical instabilities frequently encountered in Newton-type solution algorithms. Moreover, the ANN function must be invoked three additional times for each instance.

If ANN are used as a base of constitutive model, it is hard to set a strict limit between linear (elastic) and nonlinear range. The displacement-load curve observed in incremental procedure indicates appearance of nonlinearity of material response. Also iterative procedure at each incremental step requires an increasing number of iteration to fulfil convergence criteria in terms of norms of residuum and displacement sub-increment vectors. This indicates grows of the distance between linear and actual, ANN generated response.

3. Results

3.1. Model validation

To verify the correctness of the approach, we analysed the same masonry object using both of its models, with the same loads and boundary conditions. The first model was built from shell elements with an artificial neural networks driver as the constitutive macro-model, see Section 2.4. The second was a direct 3D, single-scale model, with its micro-structure represented with the same accuracy and constitutive modelling as the micro-level model used to create the large set of patterns, as described in Section 2.2. Both models, representing a repeatable fragment of the wall, are shown in detail in Figure 10. Obviously, the main difference between the two models lies in the size of the model and subsequent computational effort. It is also evident, that a direct model representing the entire wall structure, together with micro-level phenomena, would lead to multi-million DOFs (about 22 million, in this case), which is hardly acceptable as a practical situation. The comparison is straightforward for the displacement field, see Figure 11, where qualitative and quantitative agreements are visible. For the results of the membrane forces (in MN/m)

and stresses (in MPa) (Figure 12), only a qualitative agreement may be seen. It is difficult to do a quantitative assessment because of the 3D fluctuation of the stress field in the direct model, owing to the presence of the masonry microstructure.

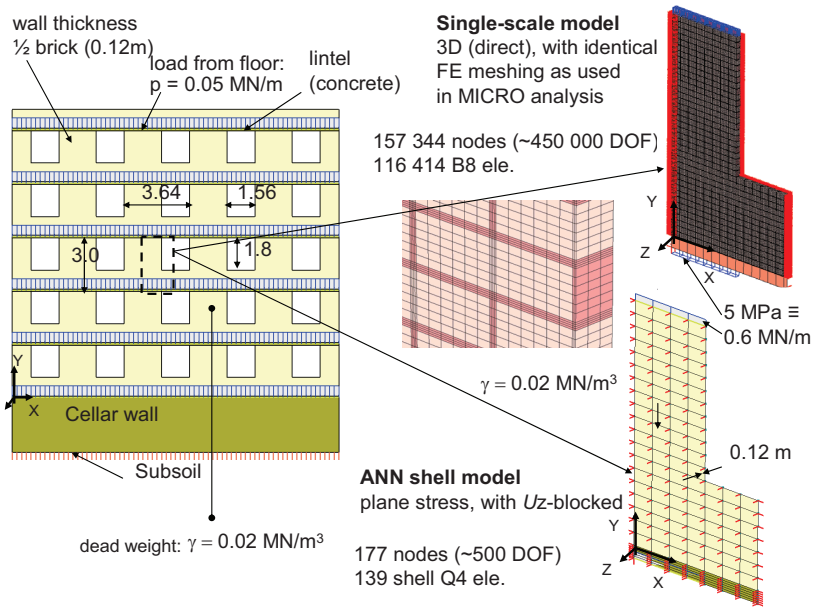

Fig. 10. FE models representing a fragment of the complete masonry wall

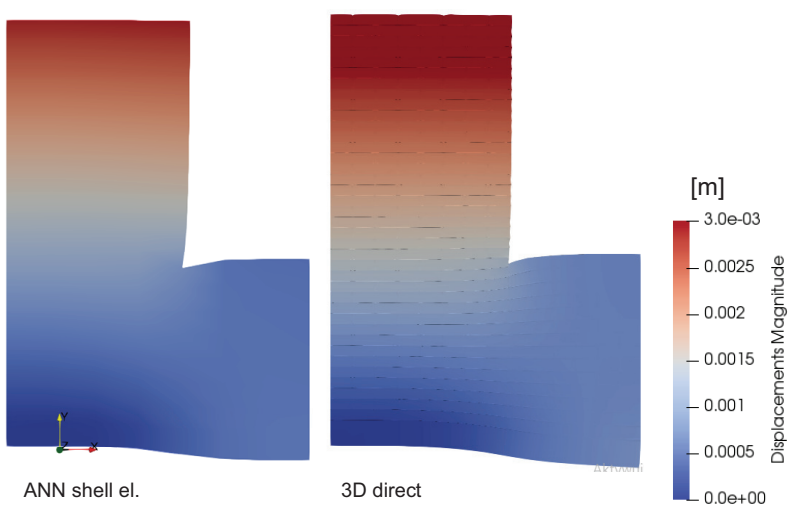

Fig. 11. Repeatable wall fragment. Displacement magnitude $\|U\|$

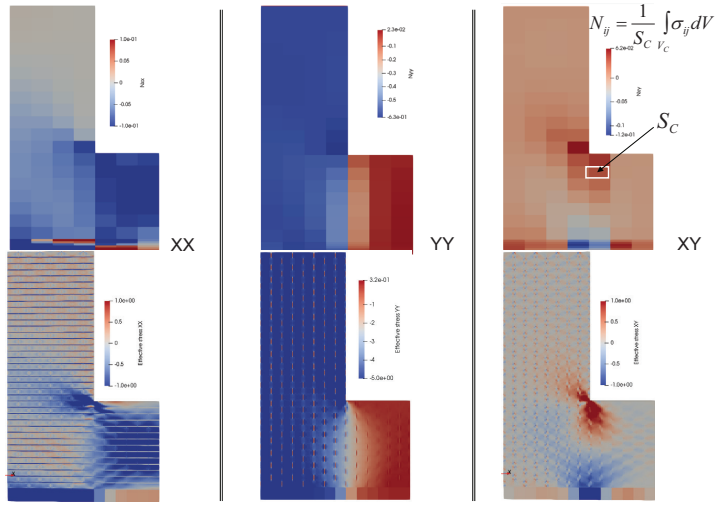

Fig. 12. Membrane forces N_{ij} (ANN shell) / Stresses σ_{ij} at the outer surface (3D direct)

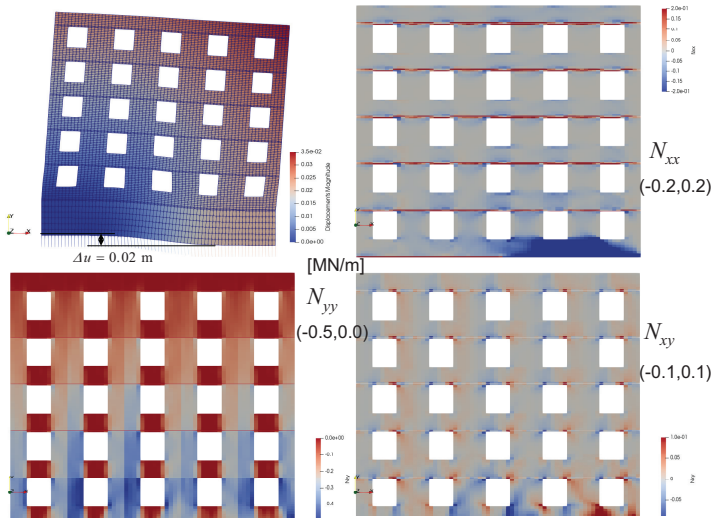

Fig. 13. Complete wall – ANN shell (demonstrative example).
Deformation and membrane forces N_{ij}

3.2. Demonstrative example

To present the possibilities and performance of the ANN method, the entire masonry wall, shown in Figure 10, is subjected to displacements applied to its base (cellar) wall. This might be seen as a simplified modelling of a typical situation when one must to assess the

influence of some type of subterranean activity, e.g. mining or tunnelling, on the structure. The model has 7287 nodes. The size of one Q4 shell element representing the masonry is equal to the size of a periodic cell (0.25 m $\times$ 0.15 m). The execution time was 240 s for a 10-steps analysis performed on a PC with an i7 processor.

4. Discussion and conclusions

During the research on the applicability of an artificial neural network based constitutive modelling within a finite element analysis, particularly in the context of masonry structures, a few issues should be considered. Firstly, large patterns set must be created, on which the artificial neural networks will be trained. The range of these patterns should cover the range element analysis. When this postulate is not fulfilled, one would hardly expect correct results. of stress points and strain increments expected to be encountered during the intended finite

The "diluting" procedure, described in Section 2.2, may be seen as a remedy, at least when monotonic loads are considered, as it was in the examples assessed. However, applying a randomised path may be helpful to make the ANN model usable, in cases when some loading-unloading action takes place. Because of the developed AppANN application, described in Section 2.3, the user does not need sophisticated knowledge about artificial neural networks algorithms and their optimisation.

The modelling of masonry walls subjected not only to in-plane action but also to bending moments is possible; however, it might require substantially greater computational effort, related to creating bigger patterns sets during homogenisation. Note that in this case, the size of the input ($\mathbf{X}$) and output ($\mathbf{Y}$) vectors will be doubled, which may have some influence on the effectiveness of the artificial neural network training procedures.

In this study only elastic-perfectly plastic models of masonry constituents were taken into account, thus phenomena like appearance of cracks was disregarded. If so, applicability of our approach is limited to situation not reaching ultimate states of masonry structures. Enhancements of ANN based modelling seems possible, but would require proper description of damage or softening behaviour for the mortar and brick during micro-level which should (as expected) be automatically covered by ANN. This may require enlargement of $\mathbf{X}$ (input) vector on total macro strains $\mathbf{E}$, or some macro-damage measures derived from micro analysis, as basing only on increment $\Delta\mathbf{E}$ would be insufficient. This, together with mesh-sensitivity for macro analysis, is an interesting problem which may be undertaken in future research. Another modelling option for farther investigation is ANN based description of loading-unloading processes, as only monotonic loading were examined in this paper. This may be needed for FE analysis of masonry structures subjected to seismic excitation.

Dealing with other types of masonry, e.g. made of hollow bricks, cavity or cinder blocks, appears to be possible within the proposed macro-constitutive modelling method.

Our final conclusion is that the artificial neural network-based constitutive models can be used in masonry modelling; however, more research is needed to make them a "black box tool" for general use during the finite element analysis.

Acknowledgements

The work was supported by Polish research grant NCBiR no. POIR 01.01.01-00-0276/16-00, entitled "Modern application of artificial neural networks in software applying the finite-element method, dedicated to solving complex engineering problems in construction", which is gratefully acknowledged.

References

[1] S. Jemioło, L. Małyszko, *FEM and constitutive modelling in analysis of masonry structures. Theoretical basis*, vol. 1. Olsztyn: University of Warmia and Mazury, 2013 (in Polish).

[2] P.B. Lourenço, "Computational strategies for masonry structures", PhD thesis, Delft University of Technology, Netherlands, 1996.

[3] P. Pegon, A. Anthoine, "Numerical strategies for solving continuum damage problems involving softening: Application to the homogenization of masonry", in *Advances in Non-Linear Finite Element Methods*, B. Topping, et. al., Eds. Edinburgh, Scotland: CIVIL-COMP Ltd.,1994, pp. 143–157.

[4] A. Cormeau, N.G. Shrive, "A 2D model for the prediction of failure modes in masonry subject to in-plane loads", in *Proceedings of 3th Computer Methods in Structural Masonry*, Book&Journals International Ltd., 1995.

[5] P. Bilko, L. Małyszko, "An Orthotropic Elastic-Plastic Constitutive Model for Masonry Walls", *Materials*, 2020, vol. 13, art. ID 4064; DOI: 10.3390/ma13184064.

[6] A. Zucchini, P.B. Lourenço, "A micro-mechanical model for the homogenisation of masonry", *International Journal of Solids and Structures*, 2002, vol. 39, pp. 3233–3255; DOI: 10.1016/S0020-7683(02)00230-5.

[7] S. Pietruszczak, X. Niu, "A mathematical description of macroscopic behaviour of brick masonry", *International Journal Solids Structures*, 1992, vol. 29, pp. 531–546; DOI: 10.1016/0020-7683(92)90052-U.

[8] A. Urbański, *The unified, finite element formulation of homogenization of structural members with a periodic microstructure*, Cracow University of Technology, 2005.

[9] M. Kawa, "Failure criterion for brick masonry: A micro-mechanics approach", *Studia Geotechnica et Mechanica*, 2014, vol. 36, no. 3, pp. 37–48; DOI: 10.2478/sgem-2014-0025.

[10] A. Urbański, F. Pachla, K. Wartak-Dobosz, "An orthotropic elasto-plastic-damage model of masonry. Implementation as ZSoil user model", in *Numerics in geotechnics and structures 2015*, T. Zimmermann, et al., Eds. Rossolis Editions & Zace Services Ltd, 2016, pp. 25–42.

[11] Z. Waszczyszyn, "Neural Networks in the Analysis and Design of Structures", *CISM Courses and Lectures*, no. 404. Wien –New York: Springer, 1999.

[12] M. Lefik, "Artificial neural network for modelling an effective behavior of composite materials", in *Proceedings of the Fifth World Congress on Computational Mechanics (WCCM V)*, *Vienna, Austria, July 7-12*. Vienna University of Technology, 2002.

[13] J. Ghaboussi, D.A. Pecknold, M. Zhang, R. HajAli, "Autoprogressive training of neural network constitutive models", *International Journal for Numerical Methods in Engineering*, 1998, vol. 42, no. 1, pp. 105–126; DOI: 10.1002/(SICI)1097-0207(19980515)42:1<105::AID-NME356>3.0.CO;2-V.

[14] Y.M.A. Hashash, S. Jung, J. Ghaboussi, "Numerical implementation of a neural network based material model in finite element analysis", *International Journal for Numerical Methods in Engineering*, 2004, vol. 59, pp. 989–1005; DOI: 10.1002/nme.905.

[15] P.A. Lucon, R.P. Donovan, "An artificial neural network approach to multiphase continua constitutive modeling", *Composites*, 2007, vol. 38, no. 7-8, pp. 817–823; DOI: 10.1016/j.compositesb.2006.12.008.

[16] G.A. Drosopoulos, G.E. Stavroulakis, "Data-driven computational homogenization using Neural Networks FE2-NN application on damaged masonry", *Journal on Computing and Cultural Heritage*, 2021, vol. 14, no. 1, pp. 1–19; DOI: 10.1145/3423154.

[17] I.B. Rocha, P. Kerfriden, F.P. van der Meer, "Micromechanics-based surrogate models for the response of composites: A critical comparison between a classical mesoscale constitutive model, hyper-reduction and neural networks", *European Journal of Mechanics and Solids*, 2020, vol. 82; DOI: 10.1016/j.euromechsol.2020.103995.

[18] B.A. Le, J. Yvonnet, Q.-C. He, "Computational homogenization of nonlinear elastic materials using neural networks", *International Journal for Numerical Methods in Engineerin*g, 2015, vol. 104, no. 12, pp. 1061–1084; DOI: 10.1002/nme.4953.

[19] H. Baluch, I. Nowosinska, "Application of Artificial Neural Networks in Planning Track Superstructure Repairs", *Archives of Civil Engineering*, 2020, vol. 66, no. 4, pp. 45–060; DOI: 10.24425/ace.2020.135208.

[20] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the theory of neural computation*. Redwood City, Calif: Addison-Wesley Pub, 1991.

[21] C.M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[22] S. Raschka, *Python Machine Learning*. Gliwice, Poland: Helion, 2017 (in Polish).

[23] M. Szeliga, *Data Science and Machine Learning*. Warsaw, Poland: PWN, 2017 (in Polish).

[24] J. Bergstra, R. Bardenet, Y. Bengio, B. Kegl, "Algorithms for Hyper-Parameter Optimization", in *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 2546–2554.

Wielo-skalowe modelowanie muru z cegieł przy użyciu metody homogenizacji i sztucznych sieci neuronowych

Słowa kluczowe: homogenizacja, konstrukcje murowe, metoda elementów skończonych, sztuczne sieci neuronowe

Streszczenie:

W pracy przedstawiono sposób tworzenia makro-modelu konstytutywnego muru ceglanego. Przyjmuje się założenia płaskiego stanu naprężenia. Tworzony model nie jest modelem ortotropowym sprężysto-plastycznym, ale jest zbudowany jako sztuczna sieć neuronowa (SSN) dająca niejawną funkcję konstytutywną. Wiąże ona nowy stan naprężeń uogólnionych (sił membranowych) Σ^{n+1} ze poprzednim stanem Σ^n oraz przyrostem odkształceń uogólnionych ΔE. Forma tak utworzonego makro-modelu konstytutywnego jest zgodna z analiza przyrostową problemu statyki w przypadku nieliniowości materiałowych.

Składniki muru (cegła i zaprawa) są opisane modelami sprężysto-plastycznymi Druckera-Pragera. Parametry materiałowe składników muru oraz geometria komórki powtarzalnej stanowią dane wejściowe, służące budowie makro-modelu muru.

Pierwszym krokiem w tworzeniu modelu jest wielokrotne przeprowadzenie procedury uogólnionej homogenizacji sterowanej odkształceniami, opisanej w [8], na trójwymiarowym modelu skończenie elementowym komórki powtarzalnej muru. W ten sposób tworzony jest duży zbiór wzorców w przestrzeni (X, Y), $X = (\Sigma^n, DE) \rightarrow Y = \Sigma^{n+1}$, które służą do trenowania SSN. Opisano sposób przygotowania tych danych uwzględniający oprócz standartowego monotonicznego sterowania wymuszeniami także i sterowanie przy użyciu losowych funkcji sterujących, oraz specjalną procedurę nazwaną „rozrzedzaniem" (ang. „diluting") znacznie zwiększającą zakres i liczność zbioru wzorców, bez konieczności zwiększania liczby przebiegów procedury homogenizacji.

Krótko omówiono zagadnienie trenowania SSN, oraz opcjonalne możliwości autorskiego oprogramowania kontrolującego proces nauczania SSN i optymalizacji jej parametrów takich jak ilość

warstw i neuronów oraz dobór funkcji aktywacji. Aplikacja ta, o nazwie AppANN, pozwala na tworzenie efektywnych SSN użytkownikowi posiadającemu jedynie podstawową wiedzę o tym narzędziu.

Jedną z opcji modelowania konstytutywnego, w którą wyposażono autorski program do analizy statycznej zadań fizycznie nieliniowych (HFEMANN), jest procedura która wartościom wektora wejściowego $X = (\Sigma^n, DE)$ przypisuje wektor $Y = \Sigma^{n+1}$, przy dowolnych parametrach SSN. Na tym etapie, danymi wprowadzanymi do programu, dla makro-modelu konstytutywnego rozważanego materiału są parametry wytrenowanej SNN.

Podejście weryfikowane jest poprzez porównanie wyników opracowanego modelu opartego na SSN z modelem bezpośrednim (jedno-skalowym) fragmentu konstrukcji murowej, które wykazało akceptowalną dokładność. W rozważanym problemie makro-modelowania muru podejście to umożliwia analizę dowolnie dużych systemów ścian murowych, co pokazano na analizie przypadku ściany 5-pietrowej budynku poddanej wymuszeniom górniczym. Wykazano efektywność w porównaniu z analizą MES modelu jedno-skalowego. Wyniki silnie zależą od pokrywania się zbiorów uczących $(\Sigma^n, \Delta E)$ z występującymi podczas analizy konstrukcji z makro-modelem opartym o SSN.

Ostateczny wniosek który podaje się w zakończeniu jest taki, że makro-modele konstytutywne oparte na sztucznych sieciach neuronowych mogą być wykorzystywane w modelowaniu murów. Potrzebne są jednak dalsze badania, aby uczynić je narzędziem do ogólnego użytku podczas analizy.

Praca powstała w ramach grantu NCBiR nr: POIR 01.01.01-00-0276/16-00, pod tytułem „Nowatorskie zastosowanie sztucznych sieci neuronowych w oprogramowaniu wykorzystującym metodę elementów skończonych, przeznaczonym do rozwiązywania złożonych problemów inżynierskich w budownictwie”.

Received: 2022-02-23, Revised: 2022-05-31