Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 365 (2020) 113008
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

# A machine learning based plasticity model using proper orthogonal decomposition

Dengpeng Huang*, Jan Niklas Fuhg, Christian Weißenfels, Peter Wriggers

*Institute of Continuum Mechanics, Leibniz Universität Hannover, Appelstr. 11, 30167 Hannover, Germany*

Received 17 December 2019; received in revised form 16 March 2020; accepted 17 March 2020
Available online 2 April 2020

**Abstract**

Data-driven material models have many advantages over classical numerical approaches, such as the direct utilization of experimental data and the possibility to improve performance of predictions when additional data is available. One approach to develop a data-driven material model is to use machine learning tools. These can be trained offline to fit an observed material behaviour and then be applied in online applications. However, learning and predicting history dependent material models, such as plasticity, is still challenging. In this work, a machine learning based material modelling framework is proposed for both elasticity and plasticity. The machine learning based hyperelasticity model is developed with the Feed forward Neural Network (FNN) directly whereas the machine learning based plasticity model is developed by using of a novel method called Proper Orthogonal Decomposition Feed forward Neural Network (PODFNN). In order to account for the loading history, the accumulated absolute strain is proposed to be the history variable of the plasticity model. Additionally, the strain–stress sequence data for plasticity is collected from different loading–unloading paths based on the concept of sequence for plasticity. By means of the POD, the multi-dimensional stress sequence is decoupled leading to independent one dimensional coefficient sequences. In this case, the neural network with multiple output is replaced by multiple independent neural networks each possessing a one-dimensional output, which leads to less training time and better training performance. To apply the machine learning based material model in finite element analysis, the tangent matrix is derived by the automatic symbolic differentiation tool AceGen. The effectiveness and generalization of the presented models are investigated by a series of numerical examples using both 2D and 3D finite element analysis.


*Keywords:* Machine learning; Artificial neural network; Plasticity; Proper orthogonal decomposition; Finite element method

## 1. Introduction

With the development of data mining technology, machine learning algorithms, high performance computing and robust numerical methods, data-driven computational modelling play an important role in not only accurate but also fast predictions of complex industrial processes. In particular, accurate material models are key parts in structure analysis. In the past years, tremendous effort has been made in developing material models, see e.g. the review of models by Cao [1] for metal forming processes. However, the proposed models show limitations in generalization or accuracy in some cases when the model is applied to engineering problems.

* Corresponding author.
*E-mail address:* huang@ikm.uni-hannover.de (D. Huang).

https://doi.org/10.1016/j.cma.2020.113008
0045-7825/

As a data-driven approach, the Machine Learning (ML) based material modelling provides an alternative tool to narrow the gap between experimental data and material models. By use of the ML technology, such as artificial neural networks, see e.g. [2], or Gaussian Processes, see e.g. [3], constitutive equations can be approximated by using experimental data without postulation of a specific constitutive model. An advantage of machine learning based material models is that they can iteratively be improved if more experimental data are available, which yields more flexible and sustainable material descriptions. For a review of machine learning in computational mechanics, see [4] and references therein.

In order to replace the classical constitutive model in computational mechanics by data-driven modelling, multiple approaches have been proposed in the literature. The model-free data-driven computing paradigm proposed by Kirchdoerfer and Ortiz [5,6],Eggersmann et al. [7] and Stainier et al. [8], conducts the computing directly from experimental material data under the constraints of conservation laws, which bypasses the empirical material modelling step. This approach works without constitutive model and seeks to find the closest possible state from a prespecified material data set. A manifold learning approach is proposed by Ibañez et al. [9,10] and Ibañez et al. [11], where the so-called constitutive manifold is constructed from collected data. A self-consistent clustering approach has been developed to predict the behaviour of heterogeneous materials under inelastic deformation, see [12] and [13]. Tang et al. [14] proposed a mapping approach, where one-dimensional data are mapped into three-dimensions for nonlinear elastic material modelling without the construction of an analytic mathematical function for the material equation. Since the performance of the data-driven computing is highly determined by quality and completeness of the available data, data completion and data uncertainties have been investigated, see [15] and [16]. Additionally, the data-driven constitutive model has been developed within a thermodynamic framework based on the so called GENERIC structure, which guarantees the energy conservation and positive entropy production, see [17] and [18].

In addition to the data-driven approaches mentioned above, the artificial neural network as a machine learning approach has been applied to approximate the constitutive model based on data as well, see [19,20], and [21]. In order to fit a constitutive material equation, the neural network is trained offline using experimental data collected from different loading paths. Afterwards, the network based model is applied online for testing and applications. A nested adaptive neural network has been applied in [19,22] for modelling the constitutive behaviour of geomaterials. In [20], a feed forward neural network based constitutive model is implemented in finite element analysis to capture the nonlinear material behaviour, where the consistent material tangent matrix is derived and evaluated. Artificial neural networks are also applied as incremental non-linear constitutive models in [21] for finite element applications. Furthermore, this approach has been applied to predict the stress–strain curves and texture evolution of polycrystalline metals by Ali et al. [23]. Instead of the offline training, the neural network based constitutive model can be trained online by auto-progressive algorithms as well, see [24] and [22]. Lastly, artificial neural networks have been applied to the heterogeneous material modelling, such as [25–28] and [29].

The data-driven model free approach conducts calculations directly from the data, which bypasses the model on one hand but highly relies on the quality and completeness of the data on the other hand. The machine learning approaches mentioned above apply the previous strain and stress as history variables, which introduces extra errors for the elastic stage of inelastic deformation, and thus affects the capabilities to capture the load history in real applications. Additionally, the derivation of the tangent matrix for the neural network based model is complex when changing the network architecture. Thus, there are many issues present in machine learning based material modelling approaches, such as the data collection strategy, the selection of history variables and the applications in finite element analysis.

The objective of this work is to develop a machine learning based hyperelastic and plasticity models for finite element applications as well as a corresponding data collection strategy. To simplify the data collection process from experiments, only strain components act as input data and only stress components represent output data. Instead of using the previous strain and stress as history variables in plasticity, in this work, the accumulated absolute total strain is applied as history variable to distinguish different loading paths. This variable can be computed from preexisting input data without additional effort as e.g. different experiments. Due to its history dependence, the training data for plasticity will be sequential data sets obtained under different loading–unloading paths. Since the isotropic plasticity can be formulated in the principle space, the training sequence data is collected only from tension and compression tests, which simplifies the data collection. A novel method called Proper Orthogonal Decomposition Feedforward Neural Network (PODFNN) is proposed in combination with the introduced history

variable for predicting the stress sequences in case of plasticity. By means of the Proper Orthogonal Decomposition (POD), the stress sequence is transformed into multiple independent coefficient sequences, where the stress at any time step can be recovered by a linear combination of the coefficients and the basis.

The presented approach decomposes the strain–stress relationship into multiple independent neural networks with only one output, which significantly decreases the complexity of the model. In order to apply the machine learning based model in finite element analysis, the tangent matrix has to be computed. It is derived by the symbolic differentiation tool AceGen, see [30]. The effectiveness and generalization of the machine learning based plasticity model is validated in 2D and 3D using several applications.

This paper is structured as follows: In Section 2, the machine learning based material modelling framework is presented. Then the Feed forward Neural network (FNN) is applied to learn the hyper-elastic material law in Section 3. In Section 4, the data collection strategy for plasticity is proposed. Based on the training data, the machine learning based plasticity model is developed and validated in finite element analysis in Section 5, which is followed by the conclusions in Section 6.

## 2. Data-driven material modelling framework

To develop a data-driven material model by means of machine learning technology, three steps are necessary: data collection, machine learning and validation, see Fig. 1. As a fundamental ingredient for data-driven models, the data, representing the material behaviour, have to be collected firstly. According to the specific problem, the training data can be collected from experiments and simulations.

In this work, strain–stress data are employed as the input and output of the data-driven model. For the plasticity model, the strain–stress data will be collected for specific loading paths and stored as sequences. Depending on the problem, the training data usually have to be preprocessed utilizing data scaling, data decomposition and data arrangement.

The second step is to fit the constitutive equation related to the data by means of the ML technology. The Artificial Neural Network (ANN) as a machine learning technology will be employed in this work. The hyperparameters of the neural network based model have to be selected according to the data and the accompanying accuracy requirements. Once the model is trained, the describing parameters will be used and stored for the material description of the developed model.

The final step is to validate the accurate reproduction of the ML based material model. To do so, the ML based material model is compared with a standard material model within several finite element applications. By deriving the tangent matrix and residual vector, the ML based model can be incorporated into a FEM code. The performance of the developed model will be evaluated by benchmark tests. If the accuracy of the material model cannot meet the necessary requirements, the model hyperparameters will be optimized or supplemental data will be collected. Therefore, the machine learning based framework is an open system and the accuracy of the developed model can be improved iteratively during its application.

## 3. Machine learning (ML) based hyperelasticity

Before the plasticity model is developed in detail, a ML based hyper-elasticity model is presuited by utilizing feed forward neural networks (FNNs) in this section.

### *3.1. Feed forward neural network*

Feed forward neural network is a fundamental machine learning technology. A deep FNN is composed of several connected layers of artificial neurons and biases, where the data is fed into an input layer and then flows through some hidden layers. The output is finally predicted at an output layer, as shown in Fig. 2. The neurons from different layers are fully connected through the weights $w$. In the prediction phase, the data flows in one way from the input layer to the output layer. In the training phase, the global error defined by the mean-squared differences between the target value and the FNN output will be back-propagated through the hidden layers. This step is performed in order to update the weights, where the objective is to minimize the global error.

At each neuron, an activation function is attached, see Fig. 3. The output of each neuron is computed by multiplying the outputs from the previous layer with the corresponding weights. For the neuron $j$ in the layer

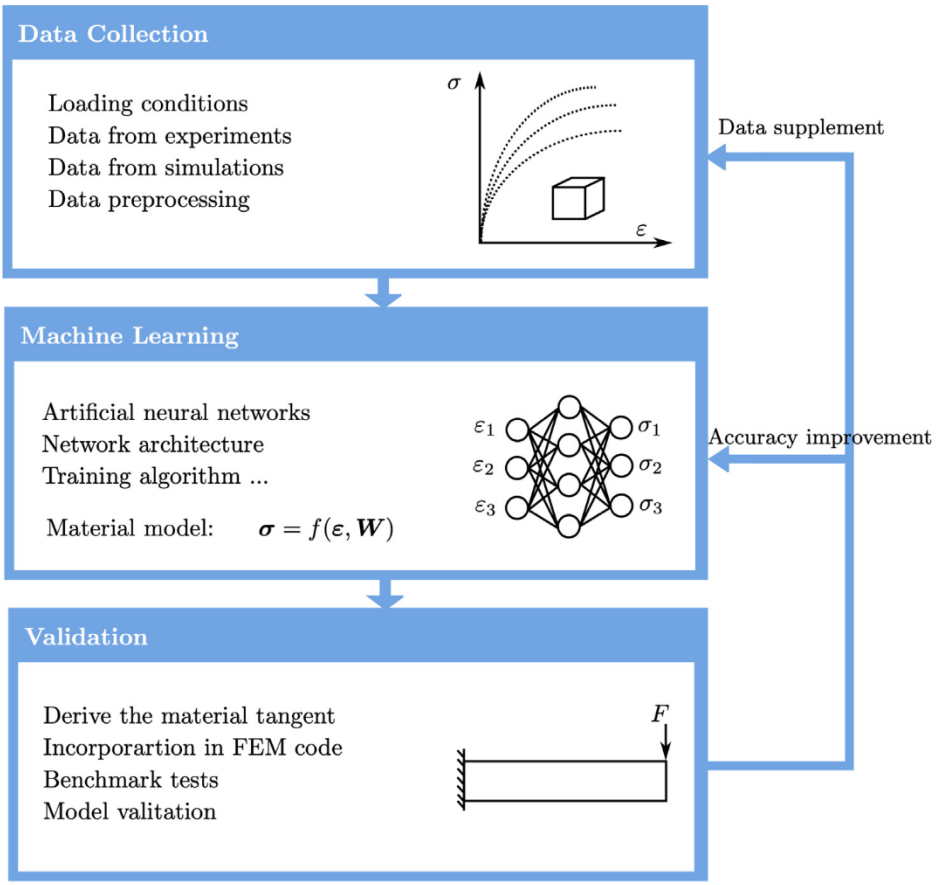



**Fig. 1.** The data-driven material modelling framework.

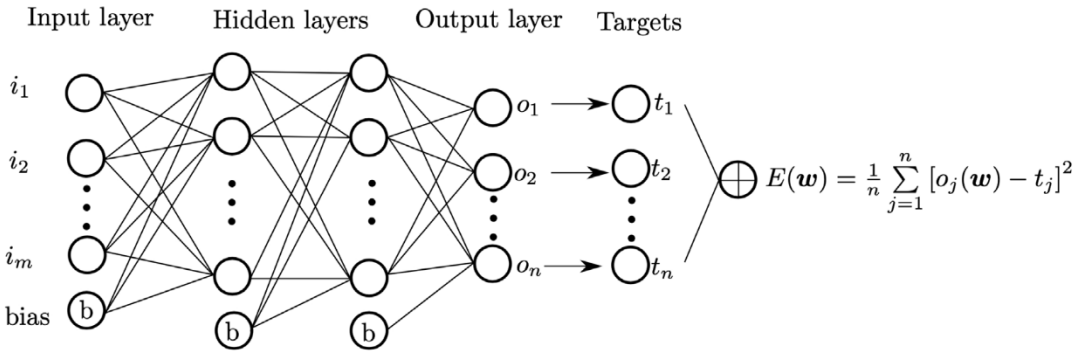



**Fig. 2.** The feed forward neural network.

$k$, the data of the previous layer $k-1$ is summed up and then altered by an activation function. The output of the neuron $j$ in layer $k$ is computed as

$$o_j^k = f_s\left(\sum_{i=1}^{N} w_{ij} o_i^{k-1} + b_i^{k-1}\right), \tag{1}$$

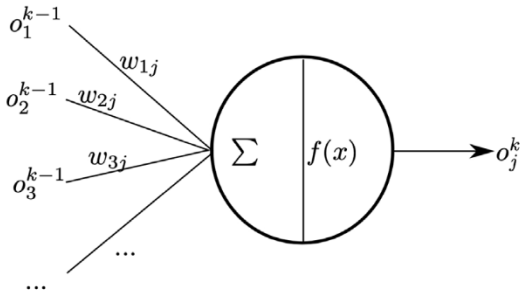



**Fig. 3.** The artificial neuron.

where $N$ is the number of neurons in the previous layer $k-1$, $w_{ij}$ is the weight connecting neurons $i$ and $j$, $o_i^{k-1}$ is the output of the neuron $i$ in layer $k-1$ whereas $b_i^{k-1}$ is its bias. A common choice for the activation function is the sigmoid function

$$f_s(x) = \frac{2}{1+e^{-2x}} - 1. \tag{2}$$

The specific architecture of the FNN, such as the number of layers and the number of neurons in each layer, has to be determined according to the complexity of the data set.

### 3.2. Neural network training

In the training phase, the weights of neural network will be initialized firstly, see [31], which is followed by the weights updating using a training algorithm such that the global error is minimized. The global error, also named as loss function or network performance, is defined according to the difference between the network prediction and the target data as shown in Fig. 2. The mean squared error is used to measure the loss

$$E(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}[o_i(\boldsymbol{w}) - t_i]^2 = \frac{1}{N}\sum_{i=1}^{N} e_i, \tag{3}$$

where $N$ is the number of outputs, $o_i$ is the $i$th output, $\boldsymbol{w}$ is the vector that contains the weights of neural network, and $t_i$ is the $i$th target value. Training a feed forward neural network is an optimization problem, where the global error is treated as the objective function. To minimize the global error, the Levenberg–Marquardt algorithm is applied to update the weighs, see [32],

$$\boldsymbol{w}^{n+1} = \boldsymbol{w}^n - (\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I})^{-1}\boldsymbol{J}^T \cdot \boldsymbol{e}, \tag{4}$$

in which $\boldsymbol{w}^{n+1}$ is the weight vector in iteration $n+1$, $\mu$ is a parameter to adaptively control the speed of convergence, and $\boldsymbol{J}$ is the Jacobian matrix that contains the derivatives of network errors with respect to the network weights

$$J_{ij} = \frac{\partial e_i}{\partial w_j^n}. \tag{5}$$

In the training process, many iterations are required to update the weights until the stopping criteria is fulfilled, where one iteration is also known as one epoch.

### 3.3. Data collection for the ML based hyperelasticity

To approximate hyperelastic behaviour by the FNN for finite element applications, the first task is to determine the input and output variables for the neural network. Since the loading and the unloading curve coincide for the elastic deformation, as shown in Fig. 4, the relationship between the strain space and stress space can be seen as a one-to-one mapping. Hence, the strain–stress mapping can be approximated by the FNN without considering the loading history.

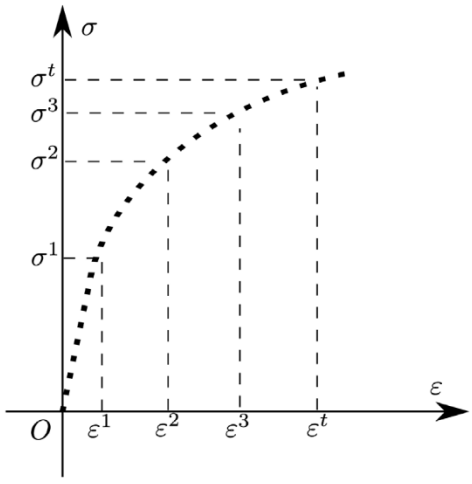



**Fig. 4.** The loading and unloading curve for hyperelasticity.

As a ML based material modelling approach, it has to be compatible not only with the unknown material law but also with the well-established material law. Thus, the well-known material model can be applied to validate the developed ML based model by comparing the simulation results. Instead of using experimental data, the training data for hyperelasticity are collected here from the non-linear neo-Hookean model, which is applied as the target model to learn

$$\boldsymbol{\sigma} = \frac{1}{2}\frac{\lambda}{J}(J^2 - 1)\boldsymbol{I} + \frac{\mu}{J}(\boldsymbol{b} - \boldsymbol{I}), \tag{6}$$

where the Cauchy stress $\boldsymbol{\sigma}$ and the left Cauchy Green tensor $\boldsymbol{b}$ are symmetric tensors. For the 2D problem, the inputs of the model can be chosen as the strain components $(J, b_{11}, b_{22}, b_{12})$, whereas the outputs are chosen as the stress components $(\sigma_{11}, \sigma_{22}, \sigma_{12})$. According to the number of input and the output, the architecture of the FNN can be determined as $4-n-3$ for instance, where one hidden layer with $n$ neurons is applied for this hyperelastic law. The input data is generated by taking equally spaced points within the given range of strain space. The stresses as output data can be computed from the neo-Hookean model in Eq. (6) accordingly.

Before training the FNN, the generated data is scaled to the range $(-1, 1)$ such that training is accelerated. Then the neural network is trained until the stopping criteria is reached. After training, the weights $\boldsymbol{w}$ and bias $\boldsymbol{b}_s$ will be saved as the model parameters. The ML based hyper-elasticity model is thus expressed as

$$\boldsymbol{\sigma}^{NN} = FNN(\boldsymbol{b}, J, \boldsymbol{w}, \boldsymbol{b}_s), \tag{7}$$

where $\boldsymbol{\sigma}^{NN}$ is the predicted Cauchy stress by the FNN.

### *3.4. The residual and tangent*

The ML based model can be used in the same way as the classical constitutive model in the finite element analysis. The residual for the static problem is given by

$$\boldsymbol{R}(\boldsymbol{u}) = \boldsymbol{f} - \int_{\Omega} \boldsymbol{B}^T \boldsymbol{\sigma}^{NN} d\Omega, \tag{8}$$

$$\boldsymbol{f} = \int_{\Omega} N\rho\hat{\boldsymbol{b}} dv - \int_{\partial\Omega} N\hat{\boldsymbol{t}} da, \tag{9}$$

in which $\boldsymbol{B}$ is the gradient of shape functions $N$, $\rho$ is the density, $\boldsymbol{\sigma}^{NN}$ is the stress computed from the machine learning based model, $\hat{\boldsymbol{b}}$ and $\hat{\boldsymbol{t}}$ are the body force and the surface traction respectively. Due to the non-linearity, the Newton Raphson iterative solution scheme is applied. The tangent matrix is computed by taking the derivative of

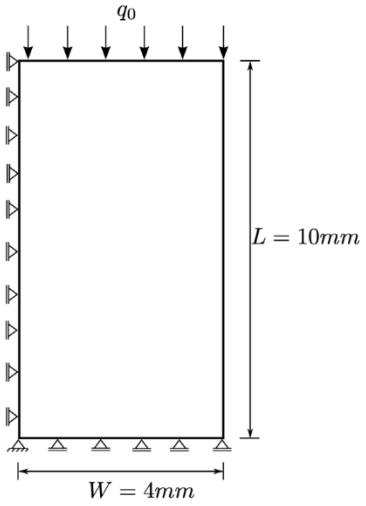



**Fig. 5.** Compression of the plate.

residual in terms of displacement

$$\boldsymbol{K}_T = \frac{\partial \boldsymbol{R}(\boldsymbol{u})}{\partial \boldsymbol{u}}. \tag{10}$$

The derivation of the tangent matrix for the neural network based model requires the computation of derivatives by the chain rule, which will be complex if the number of neuron is very large. In this work, the automatic differentiation tool AceGen, see [30], based on the symbolic computing in Mathematica is applied, by which the tangent matrix and residual vector can be derived automatically.

### *3.5. Testing the ML based hyperelasticity model in FEM*

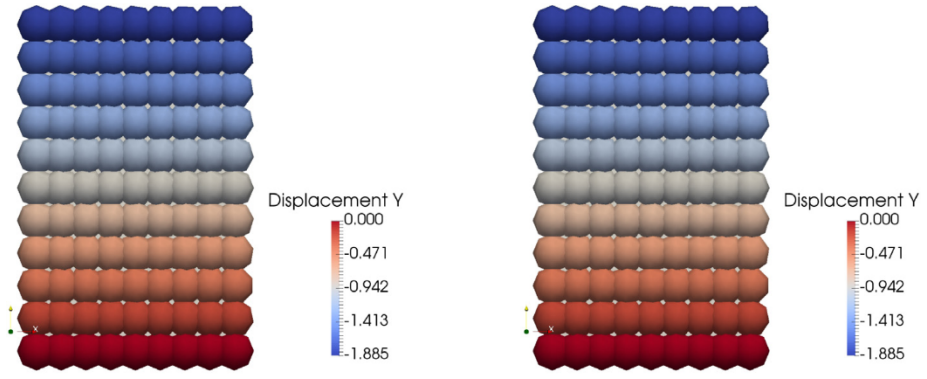
The material parameters for the neo-Hookean model used in the training data collection are set as $E = 700$ N/mm$^2$, $\nu = 0.499$. An FNN with architecture of 4-10-3 is applied, with 4 neurons in the input layer, 10 neurons in the hidden layer and 3 neurons in the output layer. The Levenberg–Marquardt algorithm [32] is applied as the training optimizer. After 14082 training iterations, the mean squared error decreased to 0.0326, which costs training time of 6 h 40 m 55 s.

The first example is the uniaxial compression test of a plate in 2D. As shown in Fig. 5, the pressure is imposed on the top surface of the plate, the bottom of the plate is fixed in vertical direction. The distributed load is given as $q_0 = -20$ MPa.

The final deformation of the plate computed with the ML based model is compared with the outcome of using the neo-Hookean model in Eq. (6). It can be see from Fig. 6 that the displacements in vertical direction are very close. The spheres displayed in the numerical examples in Fig. 6 and in the remaining figures of this work refer to the nodes of the finite element model. The computation time with analytical hyperelastic model is 8.14 s, whereas the computation time with the ML based model is 9.75 s on the same computer.

In order to further validate the generalization, a second test case, the Cook's membrane problem, is conducted. The tapered beam is clamped at the left end and loaded at the right end by a constant distributed vertical load $q_0 = 5$ Mpa, as depicted in Fig. 7. The geometric domain of the structure is discretized by 40 quadratic 9-node quadrilateral elements leading to 189 nodes.

With the same model as trained in the first test, the final deformation of the membrane is computed and compared. As shown in Fig. 8, the vertical displacement in both cases are very close to each other, which highlights the proficient generalization capabilities of the ML based elasticity model. The computation time with analytical hyperelastic model is 15.72 s, whereas the computation time with the ML based model is 19.88 s on the same computer.




(a) Neo-Hookean model (b) With ML based model

**Fig. 6.** Deformed state of the plate.

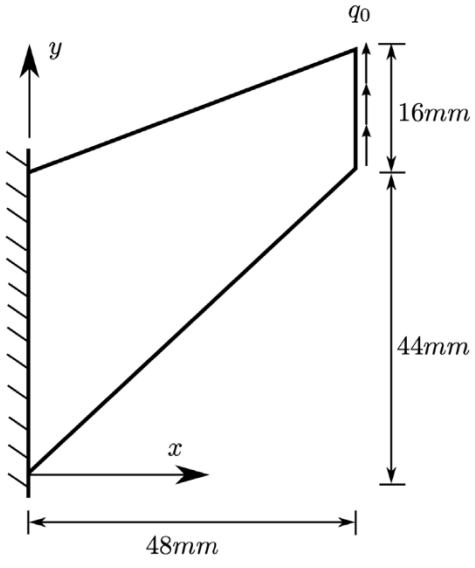
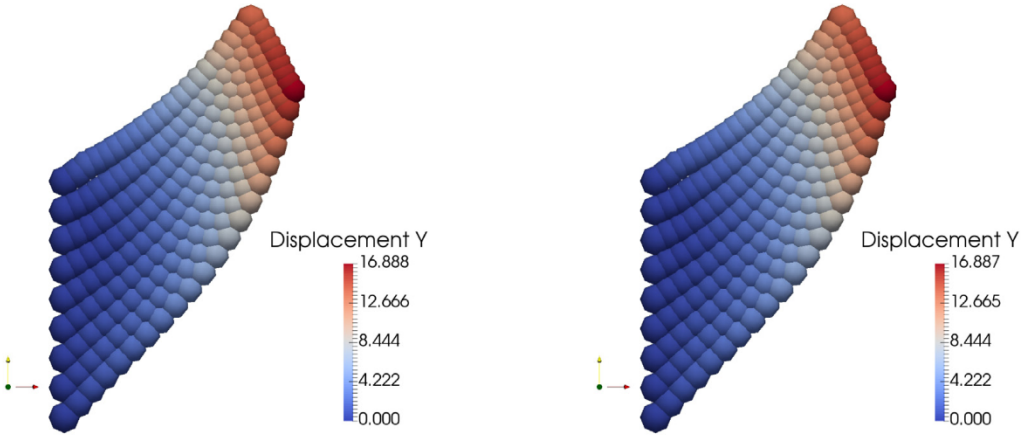



**Fig. 7.** Cook's membrane problem.

To this end, it proves that the FNN works well for approximating the nonlinear elastic behaviour as shown in the above results. Elastic deformation is a history independent process and the stress depends only on current kinematic variables. However, for plastic material behaviour, the response of the deformed material depends not only on the current deformation but also on its loading history. Since the FNN does not have any inherent ability to record loading history, the current approach needs to be improved. Furthermore, the collection process of the training data for plasticity needs to be different from elasticity.

## 4. Data collection strategy for plasticity

The aim of this part is to develop a data-driven material model which can be used to computationally reproduce the plastic material behaviour by means of machine learning tools. Collecting data from experiments is a key part for data-driven material model. In experiments, only the total strain and stress data of a specimen can be collected, which means the classical concept of elastic–plastic splitting to total strain cannot be applied in the data-driven model. This leads to the questions: how to build the data-driven model using the total strain and stress data available from experiment? and what kind of experiments have to be conducted to collect data?




(a) Neo-Hookean model (b) With ML based model

**Fig. 8.** Deformed state of the Cook's membrane.

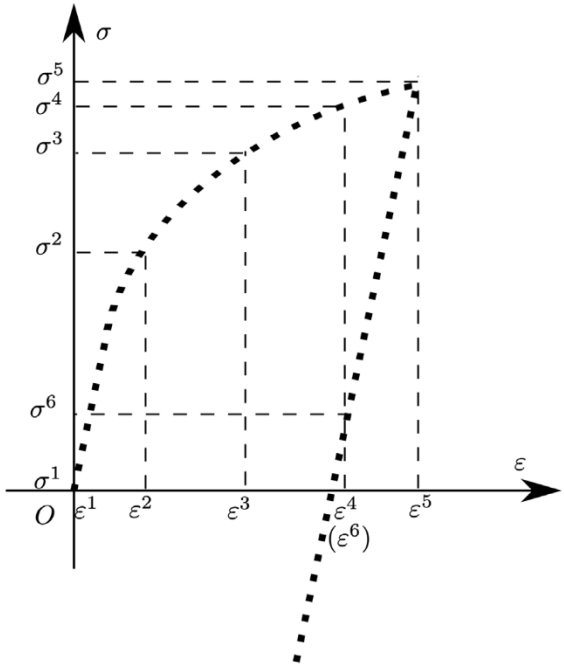



**Fig. 9.** The loading and unloading curve for plasticity.

### 4.1. Concept of sequence for plasticity

Since the plastic flow depends not only on the current stress state but also on the loading history, the plastic deformation is a history dependent process. For 1D plastic deformations, the loading and unloading curves do not coincide as shown in Fig. 9, where the strain and stress data are time series of data sets and can be seen as sequences.

Along a 1D loading–unloading path, the strain and stress data sets of one material point have a strict sequential order and can be written as two sets of corresponding sequences

$$\{\varepsilon^1, \varepsilon^2, \varepsilon^3, \ldots, \varepsilon^t, \ldots\} \Leftrightarrow \{\sigma^1, \sigma^2, \sigma^3, \ldots, \sigma^t, \ldots\}, \tag{11}$$

where $\varepsilon^t$ and $\sigma^t$ are the total strain and stress at time $t$ collected from the experiment. Thus, the basic data unit for a plasticity model is not a strain–stress pair but a strain–stress sequence pair. Each strain–stress sequence pair refers to one loading–unloading path and thus sequence data collected from different loading–unloading paths are required to train a data-driven plasticity model.

In machine learning, the classical equation of plasticity will be replaced with a ML based plasticity model driven by experimental data as

$$\boldsymbol{\sigma}^t = f^{ML}(\boldsymbol{\varepsilon}^t, \boldsymbol{h}^t), \tag{12}$$

where $\boldsymbol{h}^t$ is a history variable for distinguishing loading history and $\boldsymbol{\varepsilon}^t$ is the total strain. In this data-driven model, both the input and the output are sequence data. To build a ML based plasticity model, the history data as well as the strain–stress data have to be obtained from experiments.

The choice of history variable is crucial for a successful prediction of sequences. In the literature, the stress and strain in the last step are applied as the history information together with the current strain in the input, see [20]. However, the previous strain and previous stress are not enough to distinguish the loading history in real applications, since any error in the predicted previous stress by the ML tool will introduce an extra error into the system in an accumulate way. In this work, the accumulated absolute strain is applied in the input as history variable. The accumulated absolute strain component $\varepsilon^t_{acc,j}$ at time step $t$ can be computed for the $j$th strain component as

$$h^t_j := \varepsilon^t_{acc,j} = \begin{cases} \sum_{i=3}^{t} |\varepsilon_j^{i-1} - \varepsilon_j^{i-2}|, & t \geqslant 3, \\ 0, & t = 1, 2, \end{cases} \tag{13}$$

where $\varepsilon_j^{i-1}$ and $\varepsilon_j^{i-2}$ are total strain components at time step $i-1$ and $i-2$ respectively. This history variable has to be computed for each total strain component. $|\varepsilon_j^{i-1} - \varepsilon_j^{i-2}|$ is the absolute increment of strain component $j$ from time step $i-2$ to $i-1$, which is necessary to distinguish the loading history when tension and compression loadings are mixed, such as in loading–unloading path.

For the 1D case, depicted in Fig. 9, a monotonic loading (e.g. from $\sigma^1$ to $\sigma^4$) and a mixed loading–unloading (e.g. from $\sigma^1$ to $\sigma^6$) may lead to the same total strain (e.g. $\varepsilon^4 = \varepsilon^6$). To distinguish monotonic loading from mixed loading–unloading, the absolute value of strain increment $|\varepsilon^{i-1} - \varepsilon^{i-2}|$ is applied in Eq. (13), which leads to different accumulated absolute strain for different paths, e.g.

$$\varepsilon^6_{acc} = \sum_{i=3}^{6} |\varepsilon^{i-1} - \varepsilon^{i-2}| \tag{14}$$

$$= \sum_{i=3}^{4} |\varepsilon^{i-1} - \varepsilon^{i-2}| + \sum_{i=5}^{6} |\varepsilon^{i-1} - \varepsilon^{i-2}| \tag{15}$$

$$= \varepsilon^4_{acc} + \sum_{i=5}^{6} |\varepsilon^{i-1} - \varepsilon^{i-2}| > \varepsilon^4_{acc}. \tag{16}$$

Note that $|\varepsilon^{i-1} - \varepsilon^{i-2}|$ is applied instead of $|\varepsilon^i - \varepsilon^{i-1}|$ in Eq. (13), since $\sum_{i=3}^{t} |\varepsilon^i - \varepsilon^{i-1}|$ is equal to $\varepsilon^t$ for monotonic loading, and it is not a history variable but the current input. The advantage of applying the accumulated absolute strain as the history variable is that it can be obtained from the existing experimental data without the effort to collect them additionally.

### 4.2. Loading paths to collect data from experiments

To collect the strain–stress sequence data from experiments, the loading paths required in experiments have to be investigated. Since isotropic plasticity can be formulated in the principle strain–stress space, the ML based plasticity model can be formulated in the principle space as well, where the input and output of the model are principle stain and principle stress respectively. Therefore only the principle strain and principle stress are required to be collected from the experiments.

To collect the principle strain and principle stresses, the multi-axial loading tests can be conducted on special designed specimens, such as the biaxial experiments conducted by Mohr et al. [33] and the loading paths suggested

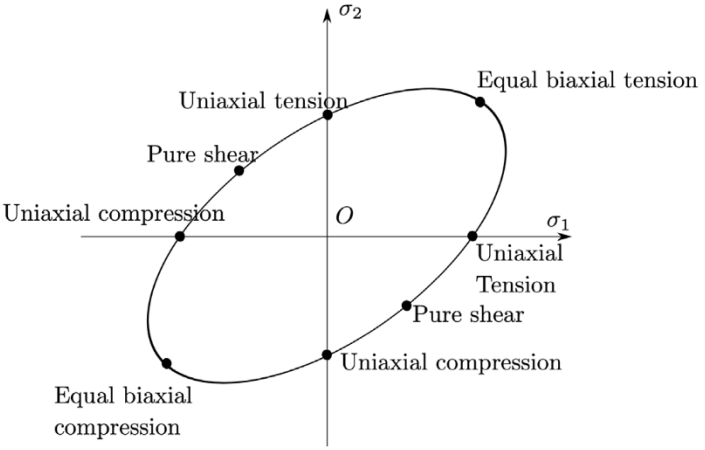



**Fig. 10.** Stress states in 2D.

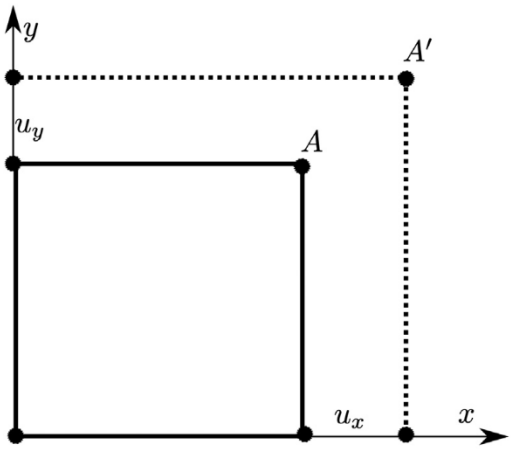



**Fig. 11.** Quadrilateral region within a specimen to collect data with biaxial loading.

by Goel et al. [34]. The von Mises yield surface covering different stress states is shown in Fig. 10 for the 2D case. In order to learn the plasticity behaviour by e.g. artificial neural networks, yielding as well as hardening effects have to be captured implicitly. To fully describe the stress states existed in the deformed structures, the data have to be collected from several tests under different loading–unloading paths. However, only biaxial tests for 2D and triaxial tests for 3D are required to collect data in principle space.

In experiments, the principle strain and stress data can be collected within a homogeneous region at one point within a specimen, which can be descried by a quadrilateral region for 2D case depicted in Fig. 11. The biaxial loading–unloading paths at this region can be characterized by the displacements of the edges connected to point $A$.

For each biaxial loading–unloading path, the point $A$ moves from its original position to $A'$ for loading and then goes back to original position for unloading, during which the displacements $(u_x, u_y)$ will increase linearly from zero to a specific value obeying $\sqrt{u_x^2 + u_y^2} = r_i, (i = 1, 2, 3, \ldots)$ and then decrease to zero. $max(r_i)$ has to be large enough to capture as much loading range of plastic deformation as possible. As shown in Fig. 12, the loading–unloading paths are just determined by setting a radius $r_i$ and different values of the angles $\phi$. Since the unloading can start from different positions, multiple circles with radius $r_i$ have to be applied in data collection.

For the 3D case, data can be collected from a cubic region, as shown in Fig. 13, where the triaxial loading–unloading paths at this point can be characterized by the displacement $(u_x, u_y, u_z)$ at the point $A$. The

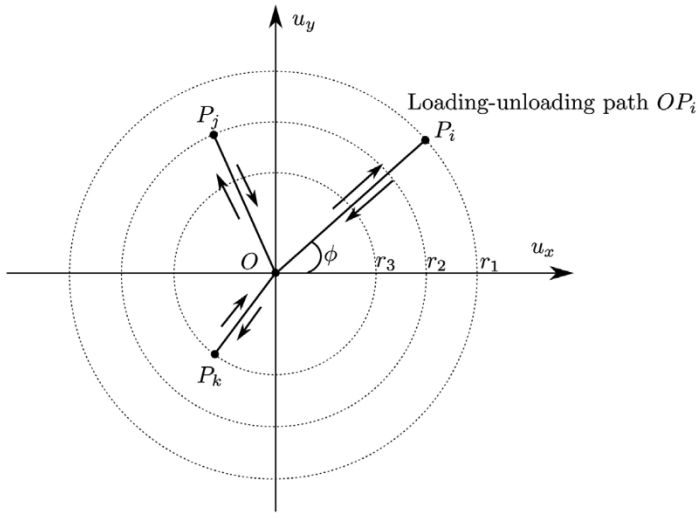



**Fig. 12.** Loading–unloading paths for data collection in 2D.

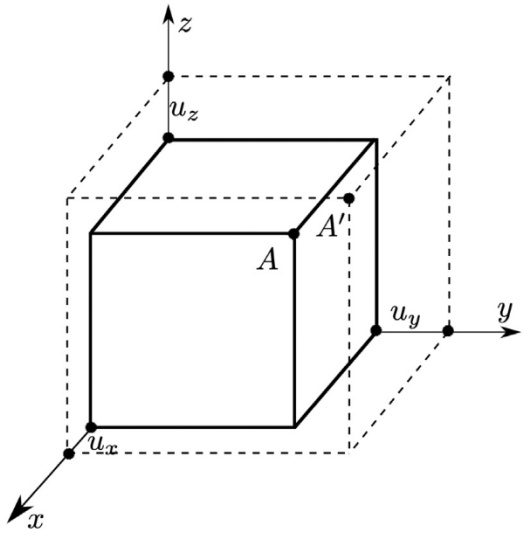



**Fig. 13.** Cubic region within a specimen to collect data with triaxial loading.

loading–unloading path for the 3D case can be generated in the spherical coordinate system as shown in Fig. 14, where the displacement of point $A$ obeys $\sqrt{u_x^2+u_y^2+u_z^2} = r_i, (i = 1, 2, 3, \ldots)$. The loading path $OP_i$ is distinguished by the angles $\phi$ and $\theta$ with radius $r_i$. By looping the loading path around the sphere, the whole range of stress states can be included.

For example, along the loading path $OP_i$ in Fig. 14, the strain–stress sequence data will be collected firstly as

$$\hat{\boldsymbol{\varepsilon}} = \begin{bmatrix} \varepsilon_1^1 & \varepsilon_1^2 & \ldots & \varepsilon_1^t & \ldots \\ \varepsilon_2^1 & \varepsilon_2^2 & \ldots & \varepsilon_2^t & \ldots \\ \varepsilon_3^1 & \varepsilon_3^2 & \ldots & \varepsilon_3^t & \ldots \end{bmatrix}_{3\times np}, \quad \boldsymbol{\sigma}_{opi} = \begin{bmatrix} \sigma_1^1 & \sigma_1^2 & \ldots & \sigma_1^t & \ldots \\ \sigma_2^1 & \sigma_2^2 & \ldots & \sigma_2^t & \ldots \\ \sigma_3^1 & \sigma_3^2 & \ldots & \sigma_3^t & \ldots \end{bmatrix}_{3\times np}, \tag{17}$$

where $np$ is the number of data point on the loading path $OP_i$, $\varepsilon_i^t$ $(i = 1, 2, 3)$ are the principle strains at time $t$ measured in the cubic region within the specimen, $\sigma_i^t$ $(i = 1, 2, 3)$ are the principle stresses at time $t$ in that region, $\hat{\boldsymbol{\varepsilon}}$ is the strain sequence and $\boldsymbol{\sigma}_{opi}$ is the stress sequence of path $OP_i$. Then the history data, accumulated absolute

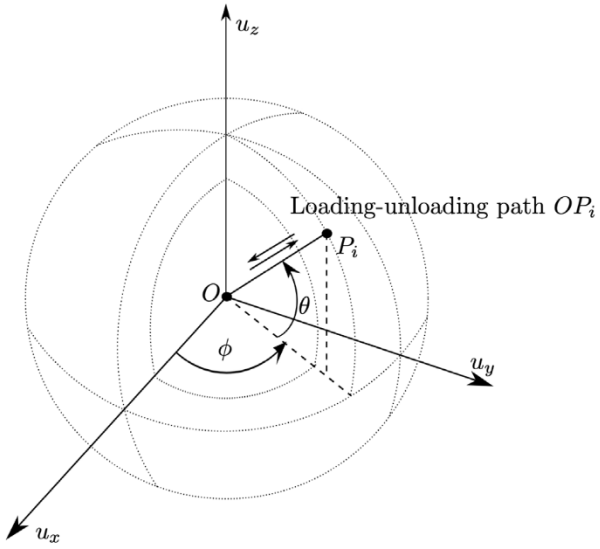



**Fig. 14.** Loading–unloading paths for data collection in 3D.

strain $\varepsilon^t_{acc}$, are computed from the strain sequence $\hat{\boldsymbol{\varepsilon}}$ using Eq. (13). The final strain sequence data of path $OP_i$ are obtained by combining the total strain sequence and the history variable sequence as

$$\boldsymbol{\varepsilon}_{opi} = \begin{bmatrix} \varepsilon^1_1 & \varepsilon^2_1 & \dots & \varepsilon^t_1 & \dots \\ \varepsilon^1_2 & \varepsilon^2_2 & \dots & \varepsilon^t_2 & \dots \\ \varepsilon^1_3 & \varepsilon^2_3 & \dots & \varepsilon^t_3 & \dots \\ \varepsilon^1_{acc,1} & \varepsilon^2_{acc,1} & \dots & \varepsilon^t_{acc,1} & \dots \\ \varepsilon^1_{acc,2} & \varepsilon^2_{acc,2} & \dots & \varepsilon^t_{acc,2} & \dots \\ \varepsilon^1_{acc,3} & \varepsilon^2_{acc,3} & \dots & \varepsilon^t_{acc,3} & \dots \end{bmatrix}_{6\times np}, \tag{18}$$

where $\varepsilon^t_{acc,1}$, $\varepsilon^t_{acc,2}$ and $\varepsilon^t_{acc,3}$ are computed from the strain components $\varepsilon^t_1$, $\varepsilon^t_2$ and $\varepsilon^t_3$ respectively according to Eq. (13). Finally, the input and output data are obtained by combining the sequences from all of loading paths $OP_i, (i = 1, 2, \dots, nl)$ as

$$\boldsymbol{I}_\varepsilon = \begin{bmatrix} \boldsymbol{\varepsilon}_{op1} & \boldsymbol{\varepsilon}_{op2} & \dots & \boldsymbol{\varepsilon}_{opi} & \dots \end{bmatrix}_{6\times M}, \quad \boldsymbol{O}_\sigma = \begin{bmatrix} \boldsymbol{\sigma}_{op1} & \boldsymbol{\sigma}_{op2} & \dots & \boldsymbol{\sigma}_{opi} & \dots \end{bmatrix}_{3\times M}, \tag{19}$$

where $nl$ is the number of loading path and $M = np \times nl$.

## 5. Machine learning based plasticity

After data collection, the feed forward neural network is used to learn the relationship within the data. The neural network will approximate a mapping between inputs and outputs. However, the accuracy of the approximation depends on the complexity of the relationship. As a widely used technique in model order reduction, the Proper Orthogonal Decomposition (POD) provides an approach to decouple the training data, which simplifies the training and increases accuracy.

### *5.1. Decoupling the stress data by POD*

The Proper Orthogonal Decomposition (POD) in combination with machine-learning tools, such as Gaussian Processes [35] and Long-Short-Term memory network [36], as a reduced order model has been used to surrogate model generation of fluid dynamic systems. Here we introduce a novel combination framework, where POD and FNNs are combined for preprocessing and prediction of sequence data. We call this approach Proper Orthogonal Decomposition Feed forward Neural Network (PODFNN). Using POD, the time series vector variables can be

represented with a reduced number of modes neglecting higher order modes if the error is acceptable. Thus, by use of the POD, the problem will be decoupled into a combination of several different modes.

As time series data, the stress sequence in training data can be rewritten as a snapshot matrix

$$\boldsymbol{O}_\sigma = \begin{bmatrix} \boldsymbol{\sigma}_{op1} & \boldsymbol{\sigma}_{op2} & \dots & \boldsymbol{\sigma}_{opi} & \dots \end{bmatrix} = \begin{bmatrix} \sigma_1^1 & \sigma_1^2 & \dots & \sigma_1^k & \dots \\ \sigma_2^1 & \sigma_2^2 & \dots & \sigma_2^k & \dots \\ \sigma_3^1 & \sigma_3^2 & \dots & \sigma_3^k & \dots \end{bmatrix}_{3\times M}, \tag{20}$$

where each column of the matrix is a snapshot and can be written as a vector $\boldsymbol{o}_\sigma^k = \begin{bmatrix} \sigma_1^k & \sigma_2^k & \sigma_3^k \end{bmatrix}^T$. Using POD, any snapshot $\boldsymbol{o}_\sigma^k$ can be represented by a linear combination of the basis

$$\boldsymbol{o}_\sigma^k = \bar{\boldsymbol{\sigma}} + \sum_{i=1}^{m} \alpha_i^k \boldsymbol{\varphi}_i, \tag{21}$$

where $\boldsymbol{\varphi}_i = \begin{bmatrix} \phi_1^k & \phi_2^k & \phi_3^k \end{bmatrix}^T$ is the $i$th basis vector, $\alpha_i^k$ is the coefficient, $m$ is the number of POD mode and $\bar{\boldsymbol{\sigma}} = \begin{bmatrix} \bar{\sigma}_1 & \bar{\sigma}_2 & \bar{\sigma}_3 \end{bmatrix}^T$ is the mean value vector of the snapshot matrix. Since $\bar{\boldsymbol{\sigma}}$, $\boldsymbol{\varphi}_i$ are constants, and the bases $\boldsymbol{\varphi}_i$ are independent with each other, the stress sequence can thus be decoupled into independent coefficient sequences.

The components of mean value vector $\bar{\boldsymbol{\sigma}}$ of the snapshot matrix are computed as

$$\bar{\sigma}_1 = \frac{1}{M}\sum_{i=1}^{M} \sigma_1^i, \quad \bar{\sigma}_2 = \frac{1}{M}\sum_{i=1}^{M} \sigma_2^i, \quad \bar{\sigma}_3 = \frac{1}{M}\sum_{i=1}^{M} \sigma_3^i. \tag{22}$$

To find the basis vectors and its coefficients, the deviation matrix is firstly computed as

$$\boldsymbol{O}_\sigma^{dev} = \begin{bmatrix} \sigma_1^1 - \bar{\sigma}_1 & \sigma_1^2 - \bar{\sigma}_1 & \dots & \sigma_1^k - \bar{\sigma}_1 & \dots \\ \sigma_2^1 - \bar{\sigma}_2 & \sigma_2^2 - \bar{\sigma}_2 & \dots & \sigma_2^k - \bar{\sigma}_2 & \dots \\ \sigma_3^1 - \bar{\sigma}_3 & \sigma_3^2 - \bar{\sigma}_3 & \dots & \sigma_3^k - \bar{\sigma}_3 & \dots \end{bmatrix}_{3\times M}. \tag{23}$$

Then, by applying Singular Value Decomposition (SVD) to the deviation matrix

$$\boldsymbol{O}_\sigma^{dev} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{24}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the unitary matrices, $\boldsymbol{S}$ is the diagonal matrix with non-negative real numbers on the diagonal, the basis vectors $\boldsymbol{\varphi}_i$ can be determined from the non-zero columns of matrix $\boldsymbol{U}$

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \boldsymbol{\varphi}_2 & \dots & \boldsymbol{\varphi}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_m \end{bmatrix}_{3\times m}, \tag{25}$$

where $\boldsymbol{u}_m$ is the $m$th non-zero column of matrix $\boldsymbol{U}$ and $m$ is equal to the rank of $\boldsymbol{O}_\sigma^{dev}$.

The coefficients $\boldsymbol{\alpha}^k = \begin{bmatrix} \alpha_1^k & \alpha_2^k & \dots & \alpha_m^k \end{bmatrix}^T$ can be obtained by projecting the snapshot $\boldsymbol{o}_\sigma^k$ on the basis matrix $\boldsymbol{\Phi}$

$$\boldsymbol{\alpha}^k = \boldsymbol{\Phi}^T \boldsymbol{o}_\sigma^k. \tag{26}$$

Since the stress components are independent, the deviation matrix $\boldsymbol{O}_\sigma^{dev}$ has the full rank of $m = 3$. In this work, all of the 3 modes are applied in the POD representation of stress sequences. The stress sequence in Eq. (19) can thus be represented by 3 independent coefficient sequences

$$\boldsymbol{O}_\sigma = \begin{bmatrix} \sigma_1^1 & \sigma_1^2 & \dots & \sigma_1^k & \dots \\ \sigma_2^1 & \sigma_2^2 & \dots & \sigma_2^k & \dots \\ \sigma_3^1 & \sigma_3^2 & \dots & \sigma_3^k & \dots \end{bmatrix}_{3\times M} \xrightarrow{POD} \begin{bmatrix} \alpha_1^1 & \alpha_1^2 & \dots & \alpha_1^k & \dots \end{bmatrix}_{1\times M} \\ \begin{bmatrix} \alpha_2^1 & \alpha_2^2 & \dots & \alpha_2^k & \dots \end{bmatrix}_{1\times M} \\ \begin{bmatrix} \alpha_3^1 & \alpha_3^2 & \dots & \alpha_3^k & \dots \end{bmatrix}_{1\times M}, \tag{27}$$

Therefore, the training data composed by the strain–stress sequences in Eq. (19) for plasticity model is transformed into training data composed by strain-coefficient sequences and can be written as

$$\begin{bmatrix} \boldsymbol{\varepsilon}_{op1} & \boldsymbol{\varepsilon}_{op2} & \dots & \boldsymbol{\varepsilon}_{opi} & \dots \end{bmatrix}_{6\times M} \Leftrightarrow \begin{bmatrix} \alpha_1^1 & \alpha_1^2 & \dots & \alpha_1^k & \dots \end{bmatrix}_{1\times M} \\ \begin{bmatrix} \alpha_2^1 & \alpha_2^2 & \dots & \alpha_2^k & \dots \end{bmatrix}_{1\times M} \\ \begin{bmatrix} \alpha_3^1 & \alpha_3^2 & \dots & \alpha_3^k & \dots \end{bmatrix}_{1\times M}, \tag{28}$$

where the three coefficient sequences are independent from each other.

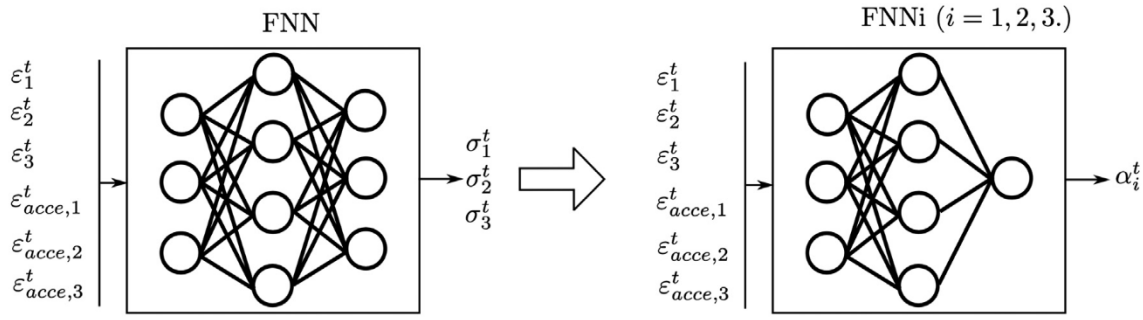



**Fig. 15.** Decoupling the strain–stress mapping (left) into independent strain-coefficient mappings (right) by POD.

### 5.2. Prediction of coefficients using FNN

Once the training data is prepared, FNNs are applied to learn the mapping between the strain sequence and the coefficient sequences in Eq. (28). Since the coefficients in the POD representation are independent, each coefficient can be predicted by one FNN, as shown in Fig. 15, where the original strain–stress mapping approximated by one complex neural network is decoupled into three independent stain-coefficient mappings approximated by simpler neural networks.

After training, the coefficient $\alpha_{NN,i}^t$ at time $t$ will be predicted by the feed forward neural network $FNN_i$ as

$$\alpha_{NN,i}^t = FNN_i(\boldsymbol{\varepsilon}^t, \boldsymbol{\varepsilon}_{acc}^t, \boldsymbol{w}, \boldsymbol{b}_s), \quad (i = 1, 2, 3), \tag{29}$$

where $i$ indicates the number of the coefficient, $\boldsymbol{\varepsilon}^t = \left[\varepsilon_1^t \quad \varepsilon_2^t \quad \varepsilon_3^t\right]^T$ is the current strain, $\boldsymbol{\varepsilon}_{acc}^t = \left[\varepsilon_{acc,1}^t \quad \varepsilon_{acc,2}^t \quad \varepsilon_{acc,3}^t\right]^T$ is the accumulated absolute strain, $\boldsymbol{w}$ is the weight matrix and $\boldsymbol{b}_s$ is the bias of neural network.

### 5.3. PODFNN Based plasticity model

Once the coefficient $\boldsymbol{\alpha}_{NN}^t$ is predicted by FNNs as described in Eq. (29), the principle stress can be recovered from the POD representation as

$$\tilde{\boldsymbol{\sigma}}^t{}_{PODFNN} = \bar{\boldsymbol{\sigma}} + \sum_{i=1}^{3} \alpha_{NN,i}^t \boldsymbol{\varphi}_i. \tag{30}$$

Finally, the Cauchy stress is obtained by transforming the principle stress into the general space

$$\boldsymbol{\sigma}^t{}_{PODFNN} = \boldsymbol{Q}\tilde{\boldsymbol{\sigma}}^t{}_{PODFNN}\boldsymbol{Q}^T. \tag{31}$$

The formulation of the ML based plasticity model defines a constitutive function in the following format

$$\boldsymbol{\sigma}^t{}_{PODFNN} = PODFNN(\boldsymbol{\varepsilon}_t, \boldsymbol{\varepsilon}_{acc}^t, \boldsymbol{w}, \boldsymbol{b}_s), \tag{32}$$

where $\boldsymbol{\varepsilon}_t$ is the current total strain, $\boldsymbol{\varepsilon}_{acc}^t$ is the history variable. To apply the ML based plasticity model in the finite element analysis, the residual vector and tangent matrix have to be derived. The automatic symbolic differentiation tool AceGen is applied to derive the tangent matrix again.

### 5.4. Testing the ML based plasticity model in FEM

In the following, the performance of the developed ML based plasticity model is evaluated using finite element applications.

*5.4.1. Data collection from analytical model*

Apart from collecting the training data from experiments, simulation data using the von Mises plasticity model can be collected as well to train the ML tool, in this way the performance of the ML based plasticity model can be verified by comparing to the analytical model. In this work, the strain–stress sequences are collected at a Gauss point of a finite element under specific loading paths described above. The strain is computed as the symmetric part of the displacement gradient for small deformations

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{H} + \boldsymbol{H}^T), \tag{33}$$

where $\boldsymbol{H}$ is the displacement gradient with $\boldsymbol{H} = \text{Grad}\boldsymbol{u}$. Using spectral decomposition, the strain can be formulated as

$$\boldsymbol{\varepsilon} = \boldsymbol{Q} \cdot \boldsymbol{\Lambda} \cdot \boldsymbol{Q}^T, \tag{34}$$

where $\boldsymbol{\Lambda}$ is the principle strain and $\boldsymbol{Q}$ is the rotation matrix obtained from the eigendirections. The principle strain $\boldsymbol{\Lambda}$ along different loading paths will serve as input data to train the ML based plasticity model.

For small strain plasticity, the additive decomposition of strain into elastic and plastic parts is assumed

$$\boldsymbol{\Lambda} = \boldsymbol{\Lambda}^e + \boldsymbol{\Lambda}^p, \tag{35}$$

where $\boldsymbol{\Lambda}^e$ and $\boldsymbol{\Lambda}^p$ are the elastic strain and plastic strain respectively. The plastically admissible stress is given by

$$\boldsymbol{\Sigma} = \rho \frac{\partial \psi}{\partial \boldsymbol{\Lambda}^e}, \tag{36}$$

where $\rho$ is the mass density and $\psi$ is called the specific free energy or alternatively the specific strain energy. The unit is energy per volume. In this work, the linear elasticity is assumed in the elastic part of deformation with the free energy density $\psi = \frac{\lambda \text{tr}^2(\boldsymbol{\Lambda}^e)}{2} + \mu \text{tr}(\boldsymbol{\Lambda}^{e,2})$, where $\lambda$ and $\mu$ are the Lamé constants. The principle stress $\boldsymbol{\Sigma}$ will serve as the output data, corresponding to the principle strain as input. By assuming the von Mises yield criteria, the yield function is written as

$$f = \sqrt{\frac{3}{2}} \| \boldsymbol{\Sigma}^{dev} \| - \sigma_y(\alpha), \tag{37}$$

where $\boldsymbol{\Sigma}^{dev}$ is the deviatoric stress with $\boldsymbol{\Sigma}^{dev} = \boldsymbol{\Sigma} - \frac{1}{3} tr\, \boldsymbol{\Sigma} \cdot \boldsymbol{1}$ and $\alpha$ is the isotropic hardening variable. By use of the associated plastic flow rule, the evolution equations for the principle plastic strain and the hardening variable are formulated as

$$\dot{\boldsymbol{\Lambda}}^p = \dot{\gamma} \frac{\partial f}{\partial \boldsymbol{\Sigma}^{dev}}, \quad \dot{\alpha} = \dot{\gamma} \frac{\partial f}{\partial A} \tag{38}$$

where $\gamma$ is the plastic multiplier and $A$ is the thermodynamic force conjugate with $\alpha$. The plastic flow has to full fill the Kuhn–Tucker conditions

$$f \leqslant 0, \quad \dot{\gamma} \geqslant 0, \quad \dot{\gamma} f = 0. \tag{39}$$

*5.4.2. Uniaxial tension and compression*

To test the performance of the ML based plasticity model, the 1D uniaxial tension and compression test is conducted firstly. The von Mises plasticity with the linear isotropic hardening is applied as the target model. The material parameters of the plasticity model are set as: Young's modulus $E = 700\ \text{N/mm}^2$, yield stress $\sigma_y = 100\ \text{MPa}$ and isotropic hardening parameter $H_{iso} = 10$. To prepare the training data, 11 sets of strain–stress sequence data are collected from the target model with strain increments being increased linearly from 0.02 to 0.03. The stress sequences are then transformed into the coefficient sequence by the POD.

The feed forward neural network with the architecture of (2-20-20-1) is applied to predict the coefficient, where the input layer containing 2 neurons is connected with two hidden layers containing 20 neurons each. The output layer contains one neuron. The input of the neural network is the total strain together with the accumulated absolute strain, and the output of the neural network is the coefficient transformed from the stress sequence. The Levenberg–Marquardt algorithm is applied as the optimizer in training. The weights are initialized by the Nguyen–Widrow method. After 4000 epochs, the mean squared error decreased to 0.0393 which costs the training time of 2 m 24 s.

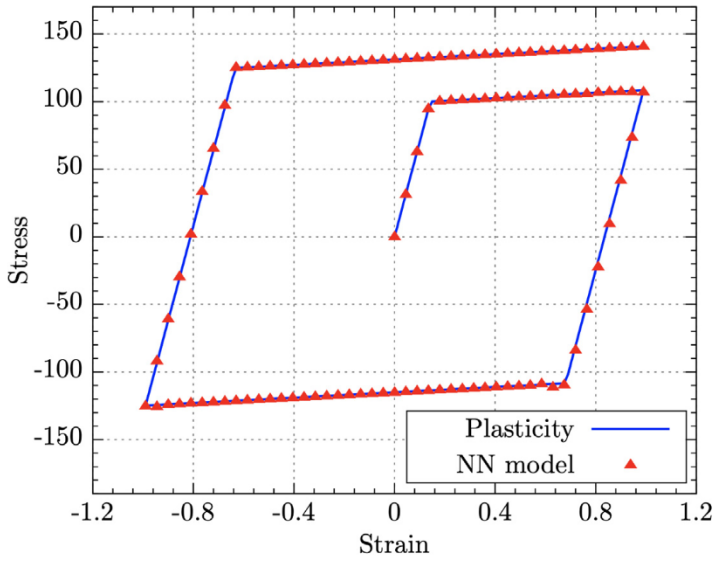



**Fig. 16.** Uniaxial tension and compression.

The testing strain sequence is generated by setting the strain increment as 0.15 so that it is different from the training data. The stress computed from the ML based plasticity model is compared with the stress from the target plasticity model, as shown in Fig. 16. It can be seen that the predicted stress follows the exact solution well, which validates the accuracy of the proposed machine learning approach for plasticity. This test shows that the accumulated absolute total strain as a history variable captures the loading history for the cyclic loading condition.

*5.4.3. Applications in 2D finite element analysis*

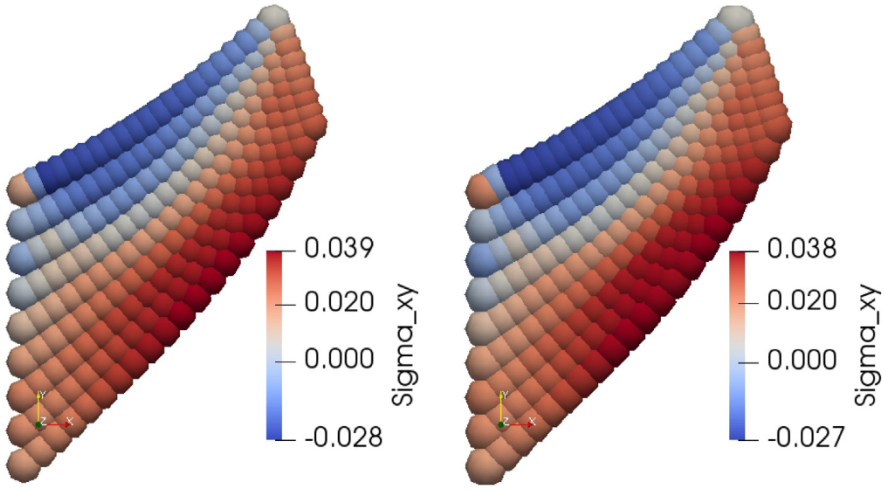
To evaluate the ML based plasticity model, benchmark tests in 2D are presented. The von Mises plasticity with an exponential isotropic hardening law $\sigma_y = y_0 + y_0(0.00002+\gamma)^{0.3}$ is set as the target model. The material parameters are set as: Young's modulus $E = 1$ N/mm$^2$, Poison's ratio $\nu = 0.33$, and the initial yield stress $y_0 = 0.05$ MPa. To collect the training data, 122 loading–unloading paths evenly distributed within the circles ($r_1 = 0.1, r_2 = 0.075$) in Fig. 12 are selected to conduct the biaxial tests, where 61 values are assigned to the angle $\phi$. Then the collected stress sequence data is transformed into the coefficient sequences using POD.

Since there are two coefficients referring to the two principle stress components in the 2D case, two FNNs will be required to predict the coefficients. In this part, the same network architecture (4-20-20-1) is applied for the two FNNs, where the input layer containing 4 neurons is connected with two hidden layers containing 20 neurons each. The output layer contains always one neuron. The total strain together with the accumulated absolute strain are applied as the input of the neural network. The output of the neural network is the coefficient transformed from the stress sequences. The Levenberg–Marquardt algorithm is applied as the optimizer as well. The training progress is terminated when the gradient of error is less than $10^{-7}$, where the mean squared error is decreased to $7.96 \times 10^{-9}$ for the first FNN and $6.35 \times 10^{-9}$ for the second FNN.

After the training process, the weights and biases of the neural network are output as the constant model parameters, by which the Cauchy stress is recovered according to the POD formulation. The tangent matrix and the residual vector are derived using the symbolic differentiation tool AceGen again.

Firstly, the 2D ML based plasticity model is tested by the Cook's membrane problem. The beam is clamped at the left end and loaded at the right end by a constant distributed vertical load $q_0 = 0.03$ Mpa, as depicted in Fig. 7. In the unloading process, the direction of vertical load is changed to be negative. The geometric domain of the structure is discretized by 40 quadratic 9-node quadrilateral elements leading to 189 nodes.

Before unloading, the stress field $\sigma_{xy}$ in the final deformation state of the beam using the proposed ML based plasticity model is compared with the target model, which is depicted in Fig. 17. It can be observed that the stress distributions of the two models are very close. The load displacement curve of the upper node (48, 60) at the right




(a) With plasticity model (b) With NN model

**Fig. 17.** Final deformation state of the 2D Cook's membrane.

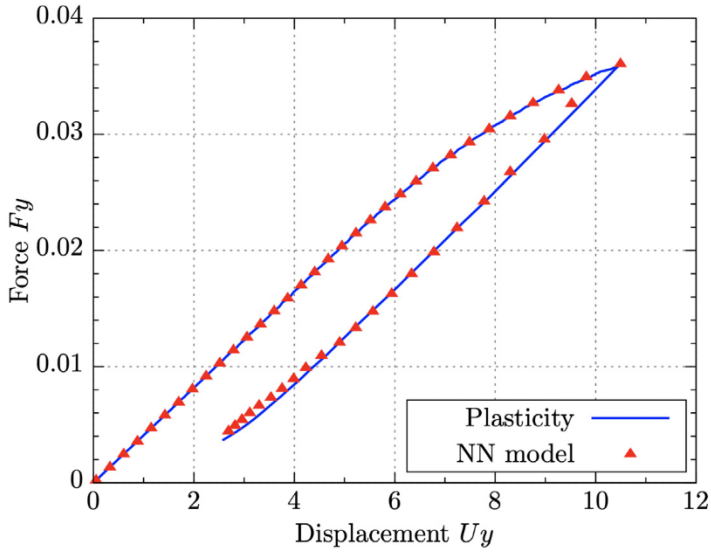



**Fig. 18.** Load deflection curve of the 2D Cook's membrane.

end of the cantilever beam is plotted in Fig. 18. The displacements in Fig. 18 and in the remaining figures of this work are given in millimetre. The figure shows that the ML based plasticity model captures the loading and unloading behaviour very well.

The second example to test the ML based plasticity model is a punch test as shown in Fig. 19, where the vertical displacement boundary condition ($u_0 = 0.07$ mm) is imposed on the top of the block and the bottom of the block is only fix in the vertical direction. In the unloading process, the direction of vertical displacement boundary is changed to be positive. The block is discretized with 100 quadratic 9-node quadrilateral elements leading to 441 nodes.

Before unloading, the stress field $\sigma_{xy}$ in the final deformation state of the block using the proposed ML based plasticity model is compared with the target model, which is depicted in Fig. 20. It can be observed that the stress distributions of the two models are very close.

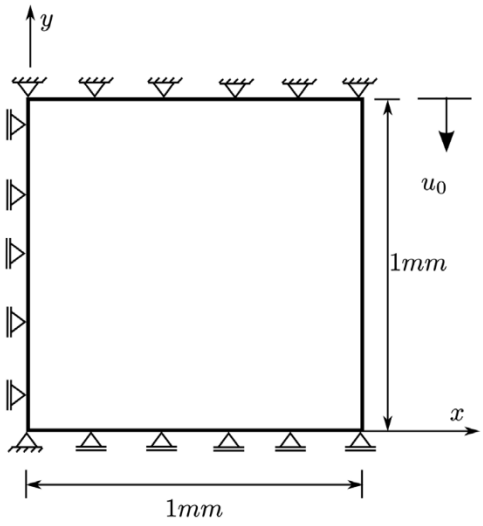
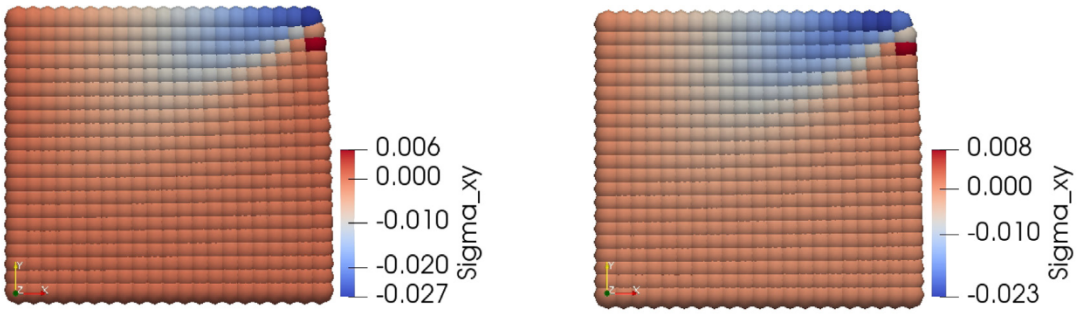



**Fig. 19.** 2D punch problem.




(a) With plasticity model

(b) With NN model

**Fig. 20.** Final deformation state of the 2D block.

The load displacement curve of the upper node (0, 1) at the left end of the block is plotted in Fig. 21, where it can be seen that the ML based model follows the plasticity model well both in loading and unloading.

*5.4.4. Applications in 3D finite element analysis*

In this section, the ML based plasticity model is extended to 3D applications. To generate the training data, one hexahedron finite element is applied to different loading situations as described in Fig. 13. The von Mises plasticity with an exponential isotropic hardening law $\sigma_y = y_0 + y_0(0.00002 + \gamma)^{0.1}$ is set as the target model. The material parameters are set as: Young's modulus $E = 10$ N/mm$^2$, Poison's ratio $\nu = 0.33$, and the initial yield stress $y_0 = 0.3$ MPa. During the training data preparation, strain–stress sequences along 8100 loading paths are generated based on the sphere ($r = 0.02$) in Fig. 14, where 90 values are assigned to the angles $\phi$ and $\theta$ respectively. Since the huge amount of data have to be collected for unloading in 3D, only the loading data is collected here and the unloading is not considered in this part.

In the three-dimensional case, three FNNs are required to predict the coefficients, which are corresponding to the principle Cauchy stress components. The same network architecture (6-16-16-1) is employed for all FNNs, where three total strain components together with three accumulated absolute strains are applied as the input of the networks. The output of the neural network is the coefficient.

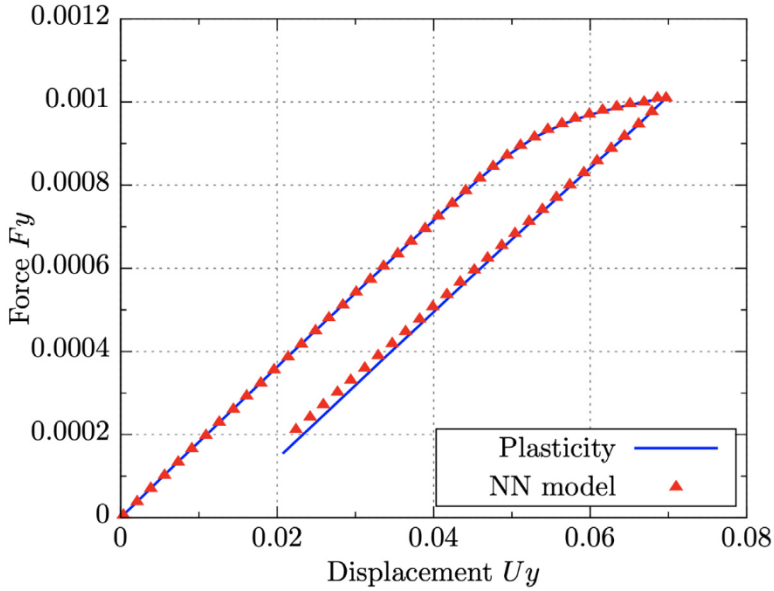



**Fig. 21.** Load deflection curve of the 2D block.

The weights are initialized by the Nguyen–Widrow method. During training, the Levenberg–Marquardt algorithm is applied as the optimizer, where the training progress is terminated when the gradient of global error is less than $10^{-7}$. The mean squared errors are decreased to $5.70 \times 10^{-8}$, $1.13 \times 10^{-9}$ and $6.78 \times 10^{-10}$ for the first, second and third FNN respectively. The training process costs time of 1 h 20 m 21 s, 1 h 18 m 46 s and 52 m 47 s for the first, second and third FNN respectively.

To evaluate the performance of the POD representation, the performances of training strain–stress model with one FNN(6-16-16-3) and training three strain-coefficient models with three FNNs(6-16-16-1) are compared. Without POD, only one FNN is required to approximate the mapping, where the output includes 3 stress components. With POD, three independent FNNs will be applied, where the output of FNN includes only one POD coefficient. The training performances within 2000 epochs are shown in Fig. 22. It can be seen that the average of the mean squared errors of the three FNNs is smaller than that without POD. Additionally, training a FNN with architecture of (6-16-16-3) costs computation time of 4h22m43s whereas the average training time of the three FNNs (6-16-16-1) is 1h26m27. It can be observed that the POD approach leads to less training time and better training performance.

The first example to test the 3D machine learning based plasticity model is the necking of a bar as shown in Fig. 23, where the left end of the bar is fixed and the displacement boundary $u_0 = 0.05$ mm is imposed at the right end along its axial direction. An artificial imperfection is set in the centre of the bar to trigger the necking, where the radius at the centre is chosen to be $R_c = 0.98R$. The bar is discretized with 200 quadratic 27-node elements leading to 2193 nodes.

Fig. 24 shows the final deformation of the bar after tension, where only one quarter of the bar is computed due to the symmetry. It can be observed that the amounts of the necking computed by the two models are close to each other.

The load displacement curve of the bar under the uniaxial tension is plotted in Fig. 25, where the neural network based model follows the plasticity model quite well.

The second example is the punch test, where the vertical displacement boundary condition ($u_0 = 0.15$ mm) is imposed on the top of the block and the bottom of the block is only fixed in the vertical direction, as shown in Fig. 26. The block is discretized with 100 quadratic 27-node elements leading to 441 nodes.

Fig. 27(a) shows the final deformation of the block after compression. It can be observed that the displacements in the horizontal direction computed by the two models are close to each other. The load displacement curve of the block under compression is plotted in Fig. 28. It can be seen that the neural network based model follows the plasticity model quite well.

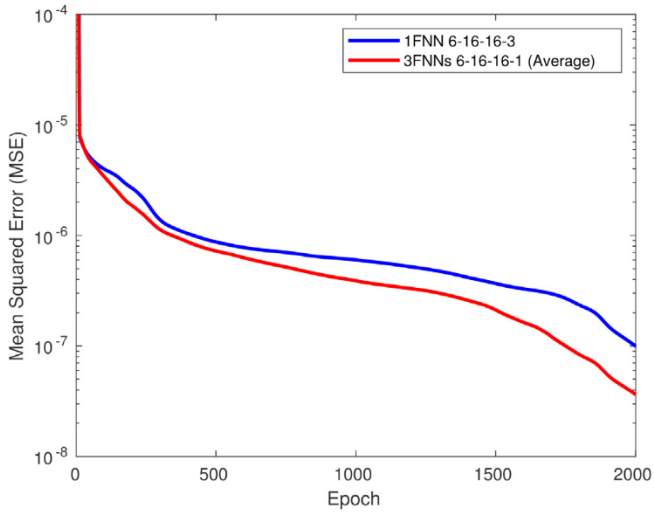



**Fig. 22.** MSE of training FNN(6-16-16-3) and the average MSE of training 3 FNNs(6-16-16-1).

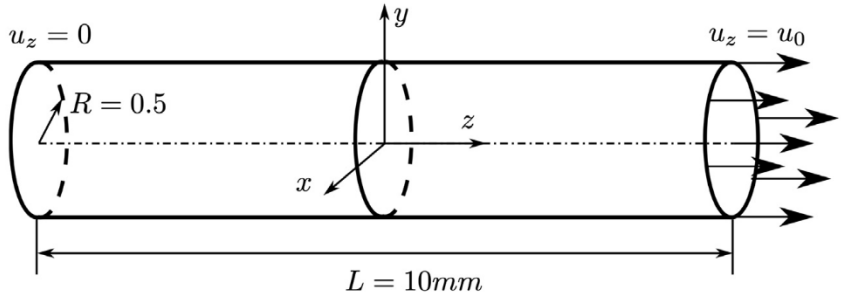
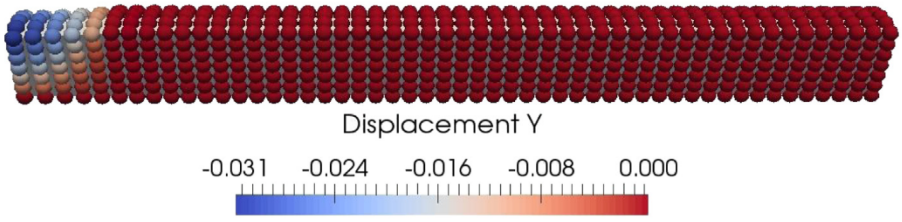
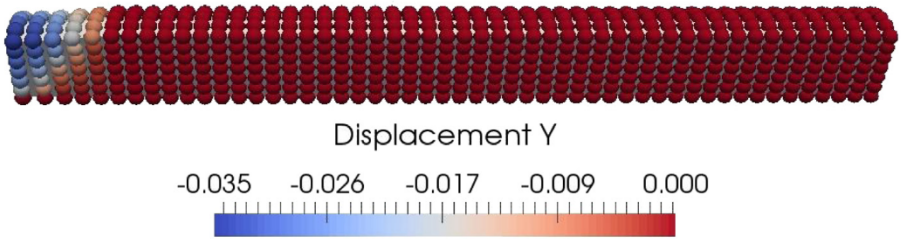



**Fig. 23.** Geometry and boundary conditions of the bar.




(a) With plasticity model




(b) With NN model

**Fig. 24.** Final deformation state of the cylindrical bar.

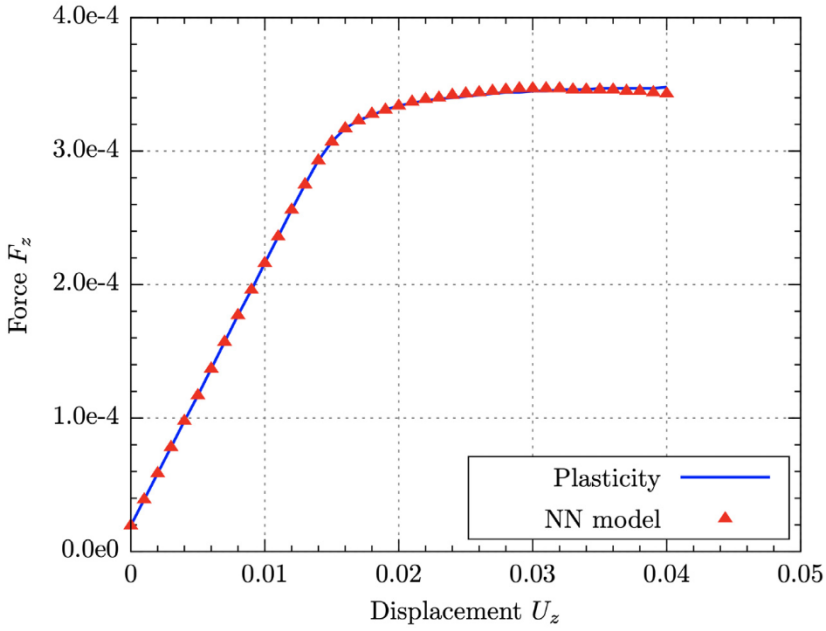



**Fig. 25.** Load deflection curve of the cylindrical bar.

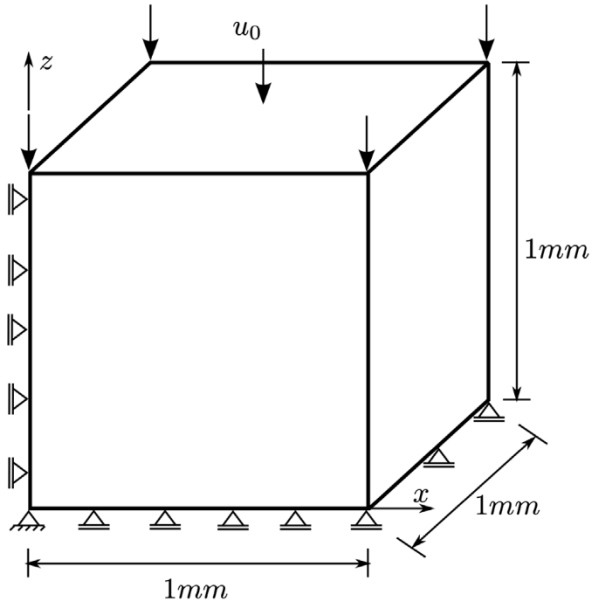
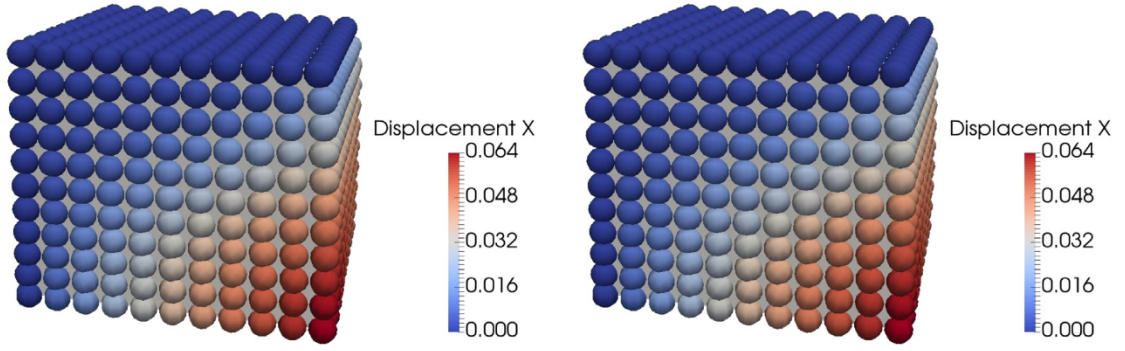



**Fig. 26.** 3D punch problem.

The last example for the 3D neural network based model is the Cook's membrane problem. The 3D beam is clamped at the left end and loaded at the right end by a constant distributed vertical load $q_0 = 0.03$ MPa, as depicted in Fig. 29. By use of the quadratic finite element with 27 nodes, the beam is discretized using 1080 elements leading to 10309 nodes.

Fig. 30 shows the final deformation of the Cook's membrane. It can be observed that the displacements in the vertical direction computed by the two models are almost identical. The load displacement curve of the upper node




(a) With plasticity model (b) With NN model

**Fig. 27.** Final deformation state of the 3D punch problem.

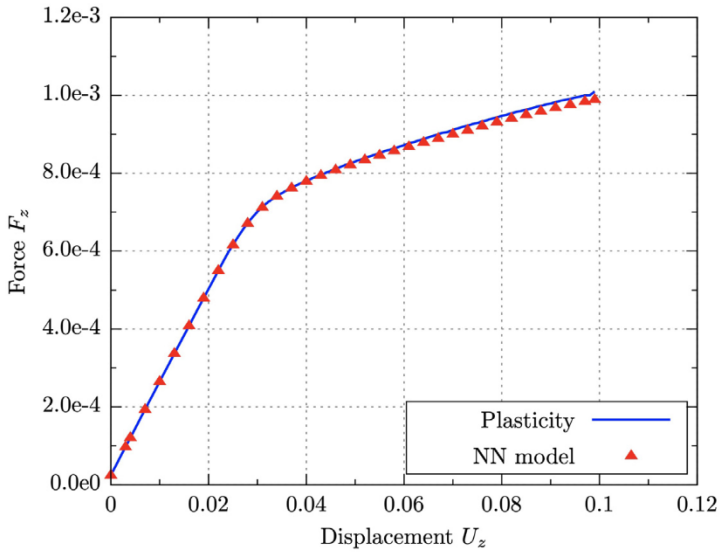



**Fig. 28.** Load deflection curve of the 3D punch problem.

(48,0,60) is plotted in Fig. 31. It can be seen that the machine learning based model follows the plasticity model quite well.

## 6. Conclusions

In this work, a machine learning based material modelling approach for hyper-elasticity and plasticity is proposed. Common tools such as FNNs show proficient performances for capturing the mapping between strain and stress in the case of elasticity. However, history variables are required to distinguish the loading history in case of plasticity. In this context, FNNs show subpar performances. Thus, the accumulated absolute strain is proposed to be the history variable, which captures the loading history well without requirement for additional data. Here we present a novel method called Proper Orthogonal Decomposition Feed forward Neural Network (PODFNN), which in combination with the introduced history variable is able to overcome this problem. By use of the POD, less training time and better training performance are obtained in the network training. Additionally, it has been shown that the training data collected only from the multi-axial loading tests are enough to capture the von Mises yield surface and the hardening law. The automatic symbolic differentiation tool AceGen provides a very convenient way to derive the

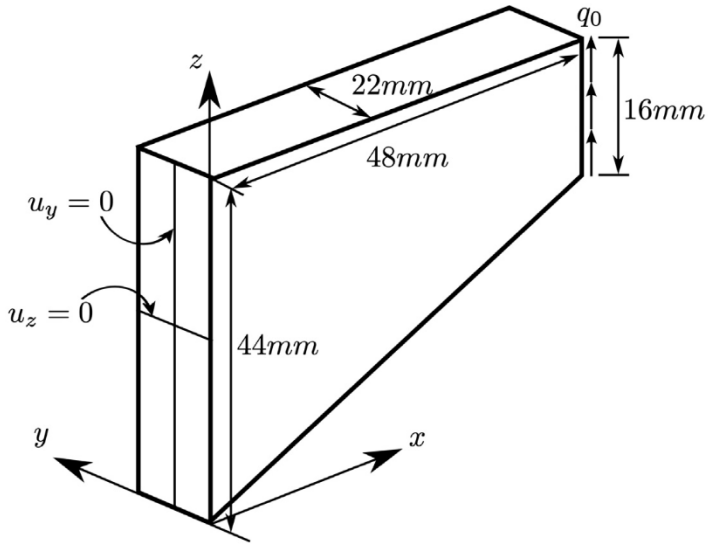
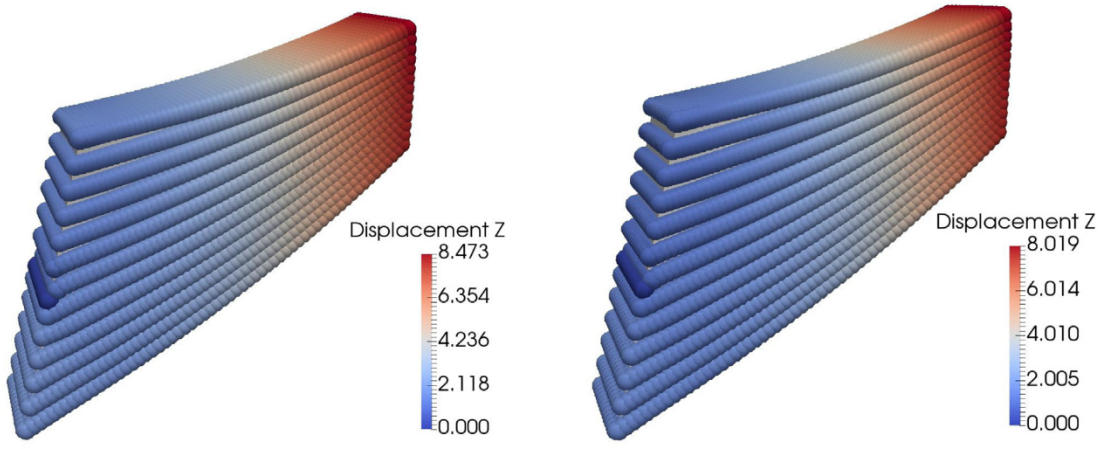



**Fig. 29.** 3D Cook's membrane problem.




(a) With plasticity model (b) With NN model

**Fig. 30.** Final deformation state of the 3D Cook's membrane.

tangent matrix for the machine learning based material model. The generalization and accuracy of the presented model as well as the data generation strategy have been verified by finite element applications both in 2D and 3D.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The first author would like to thank the China Scholarship Council (CSC) and the Graduate Academy of Leibniz Universität Hannover for the financial support. The second author acknowledges the financial support from the Deutsche Forschungsgemeinschaft under Germanys Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453). The last author acknowledges the support of Deutsche Forschungsgemeinschaft for the project C2 within the collaborative research center/Transregio TR73.

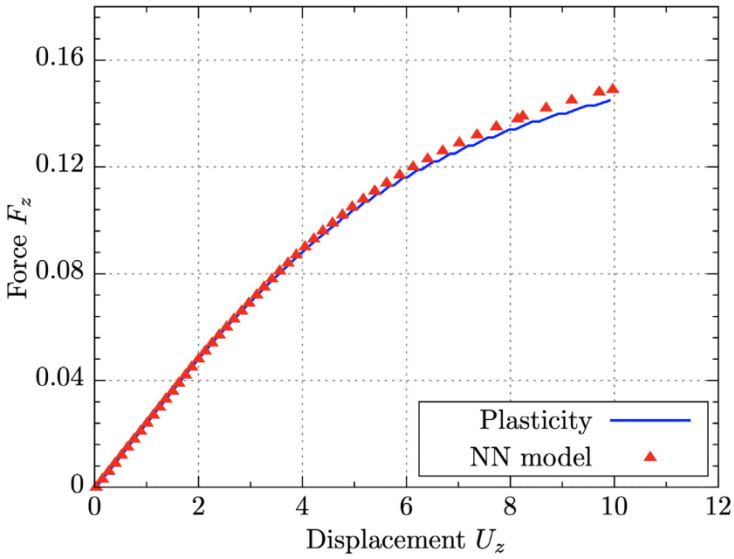



**Fig. 31.** Load deflection curve of the 3D Cook's membrane.

## References

[1] Trong Son Cao, Models for ductile damage and fracture prediction in cold bulk metal forming processes: a review, Int. J. Mater. Form. 10 (2) (2017) 139–171.
[2] Mohamad H. Hassoun, et al., Fundamentals of Artificial Neural Networks, MIT press, 1995.
[3] Carl Edward Rasmussen, GaussIan processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.
[4] Atsuya Oishi, Genki Yagawa, Computational mechanics enhanced by deep learning, Comput. Methods Appl. Mech. Engrg. 327 (2017) 327–351.
[5] Trenton Kirchdoerfer, Michael Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[6] T. Kirchdoerfer, M. Ortiz, Data-driven computing in dynamics, Internat. J. Numer. Methods Engrg. 113 (11) (2018) 1697–1710, http://dx.doi.org/10.1002/nme.5716, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.5716.
[7] R. Eggersmann, T. Kirchdoerfer, S. Reese, L. Stainier, M. Ortiz, Model-free data-driven inelasticity, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 350 (2019) 81–99, http://dx.doi.org/10.1016/j.cma.2019.02.016, URL http://www.sciencedirect.com/science/article/pii/S0045782519300878.
[8] Laurent Stainier, Adrien Leygue, Michael Ortiz, Model-free data-driven methods in mechanics: material data identification and solvers, Comput. Mech. (2019) 1–13.
[9] Ruben Ibañez, Domenico Borzacchiello, Jose Vicente Aguado, Emmanuelle Abisset-Chavanne, Elias Cueto, Pierre Ladeveze, Francisco Chinesta, Data-driven non-linear elasticity: constitutive manifold construction and problem discretization, Comput. Mech. (ISSN: 1432-0924) 60 (5) (2017) 813–826, http://dx.doi.org/10.1007/s00466-017-1440-1, URL https://doi.org/10.1007/s00466-017-1440-1.
[10] Ruben Ibañez, Emmanuelle Abisset-Chavanne, Jose Vicente Aguado, David Gonzalez, Elias Cueto, Francisco Chinesta, A manifold learning approach to data-driven computational elasticity and inelasticity, Arch. Comput. Methods Eng. 25 (1) (2018) 47–57.
[11] Ruben Ibañez, Emmanuelle Abisset-Chavanne, Jean-Louis González, Elias Cueto, Francisco Chinesta, Hybrid constitutive modeling: data-driven learning of corrections to plasticity models, Int. J. Mater. Form. 12 (4) (2019) 717–725.
[12] Zeliang Liu, M.A. Bessa, Wing Kam Liu, Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials, Comput. Methods Appl. Mech. Engrg. 306 (2016) 319–341.
[13] Modesar Shakoor, Orion L. Kafka, Cheng Yu, Wing Kam Liu, Data science for finite strain mechanical science of ductile materials, Comput. Mech. 64 (1) (2019) 33–45.
[14] Shan Tang, Gang Zhang, Hang Yang, Ying Li, Wing Kam Liu, Xu Guo, Map123: A data-driven approach to use 1d data for 3d nonlinear elastic materials modeling, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112587.
[15] Jacobo Ayensa-Jiménez, Mohamed H. Doweidar, Jose A. Sanz-Herrera, Manuel Doblaré, A new reliability-based data-driven approach for noisy experimental data with physical constraints, Comput. Methods Appl. Mech. Engrg. 328 (2018) 752–774.
[16] Jacobo Ayensa-Jiménez, Mohamed H. Doweidar, Jose A. Sanz-Herrera, Manuel Doblaré, An unsupervised data completion method for physically-based data-driven models, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 344 (2019) 120–143, http://dx.doi.org/10.1016/j.cma.2018.09.035, URL http://www.sciencedirect.com/science/article/pii/S0045782518304882.
[17] David González, Francisco Chinesta, Elías Cueto, Learning corrections for hyperelastic models from data, 2019.
[18] David González, Francisco Chinesta, Elías Cueto, Thermodynamically consistent data-driven computational mechanics, Contin. Mech. Thermodyn. 31 (1) (2019) 239–253.

[19] J. Ghaboussi, D.E. Sidarta, New nested adaptive neural networks (nann) for constitutive modeling, Comput. Geotech. 22 (1) (1998) 29–52.
[20] Y.M.A. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, Internat. J. Numer. Methods Engrg. 59 (7) (2004) 989–1005.
[21] M. Lefik, B.A. Schrefler, Artificial neural network as an incremental non-linear constitutive model for a finite element code, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3265–3283.
[22] Jamshid Ghaboussi, David A. Pecknold, Mingfu Zhang, Rami M. Haj-Ali, Autoprogressive training of neural network constitutive models, Internat. J. Numer. Methods Engrg. 42 (1) (1998) 105–126.
[23] Usman Ali, Waqas Muhammad, Abhijit Brahme, Oxana Skiba, Kaan Inal, Application of artificial neural networks in micromechanics for polycrystalline metals, Int. J. Plast. (ISSN: 0749-6419) (2019) http://dx.doi.org/10.1016/j.ijplas.2019.05.001, URL http://www.sciencedirect.com/science/article/pii/S0749641918307290.
[24] Ewa Pabisek, Self-learning fem/nmm approach to identification of equivalent material models for plane stress problem, Comput. Assist. Mech. Eng. Sci. 15 (1) (2008) 67.
[25] B.A. Le, Julien Yvonnet, Q.-C. He, Computational homogenization of nonlinear elastic materials using neural networks, Internat. J. Numer. Methods Engrg. 104 (12) (2015) 1061–1084.
[26] Xiaoxin Lu, Dimitris G. Giovanis, Julien Yvonnet, Vissarion Papadopoulos, Fabrice Detrez, Jinbo Bai, A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites, Comput. Mech. 64 (2) (2019) 307–321.
[27] Xiang Li, Zhanli Liu, Shaoqing Cui, Chengcheng Luo, Chenfeng Li, Zhuo Zhuang, Predicting the effective mechanical property of heterogeneous materials by image based modeling and deep learning, Comput. Methods Appl. Mech. Engrg. 347 (2019) 735–753.
[28] Zeliang Liu, C.T. Wu, M. Koishi, A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials, Comput. Methods Appl. Mech. Engrg. 345 (2019) 1138–1168.
[29] Hang Yang, Xu Guo, Shan Tang, Wing Kam Liu, Derivation of heterogeneous material laws via data-driven principal component expansions, Comput. Mech. (2019) 1–15.
[30] Joze Korelc, Peter Wriggers, Automation OfFinite Element Methods, Springer, 2016.
[31] Derrick Nguyen, Bernard Widrow, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, in: 1990 IJCNN International Joint Conference on Neural Networks, IEEE, 1990, pp. 21–26.
[32] Martin T. Hagan, Training feed forward networks with the marquardt algorithm, IEEE Trans. Neural Netw. 5 (6) (1994) 989–993.
[33] Dirk Mohr, Matthieu Dunand, Keun-Hwan Kim, Evaluation of associated and non-associated quadratic plasticity models for advanced high strength steel sheets under multi-axial loading, Int. J. Plast. 26 (7) (2010) 939–956.
[34] A. Goel, A. Sherafati, Mehrdad Negahban, A. Azizinamini, Yenan Wang, A finite deformation nonlinear thermo-elastic model that mimics plasticity during monotonic loading, Int. J. Solids Struct. 48 (20) (2011) 2977–2986.
[35] Manyu Xiao, Piotr Breitkopf, Rajan Filomeno Coelho, Catherine Knopf-Lenoir, Maryan Sidorkiewicz, Pierre Villon, Model reduction by cpod and kriging, Struct. Multidiscip. Optim. 41 (4) (2010) 555–574.
[36] Arvind T. Mohan, Datta V. Gaitonde., A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks, 2018, arXiv preprint arXiv:1804.09269.