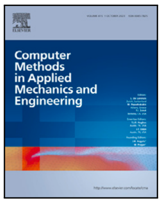
Computer Methods in Applied Mechanics and Engineering 420 (2024) 116672

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma



A framework for neural network based constitutive modelling of inelastic materials

Wulf G. Dettmer [*], Eugenio J. Muttio, Reem Alhayki, Djordje Perić

Faculty of Science And Engineering, Swansea University, Bay Campus, Fabian Way, Swansea, SA1 8EN, Wales, United Kingdom

ARTICLE INFO

Keywords:
Data driven computational mechanics
Neural network
Constitutive modelling
Elasto-plasticity
Damage mechanics

ABSTRACT

Given the significant recent advances in added layer manufacturing and materials engineering, new types of materials or new material micro-structures are becoming available at a fast rate. The finite element analysis of structures or structural components requires a constitutive model that describes the behaviour of the new materials. The formulation of accurate constitutive equations is generally complex and time consuming. Hence, suitable machine learning strategies may be used to render this process obsolete and bridge the gap between experimental data and finite element analysis. In this work, a generic stress update procedure is presented that is suitable for the modelling of rate-independent, elastic or inelastic, isotropic or anisotropic material behaviour. The proposed strategy is based on a recurrent neural network architecture and must be trained on stress and strain data sequences that represent physical or numerical experiments. A training strategy based on gradient-free optimisation is presented. It is shown that piecewise linear behaviour, such as uniaxial elasto-plasticity, can be represented *exactly*. Further numerical examples include uniaxial damage mechanics and elasto-plasticity under plane strain conditions. An efficient criterion for the verification of thermodynamic consistency is proposed and applied to the trained stress update models. The strategy is compared to GRU or LSTM based architectures and shown to offer advantages.

1. Introduction

Data-driven constitutive modelling has been the subject of many scientific publications during the last two decades. While many models are based on machine learning and designed to replace the local stress update procedure within the finite element setting, some strategies follow different approaches and result in overall computational procedures that differ from standard finite element methodologies, see for instance [1,2]. Strategies based on machine learning have been proposed for hyper-elasticity [3–5], viscoelasticity [6], elasto-plasticity [7–13], visco-plasticity [14,15], cyclic plasticity [16–18], traction–separation [19], continuum damage [20], interface mechanics [19,21], tensile damage in flexure [22], rate-dependent materials [23] and others.

The objective of this work is the development of a generalised strategy for neural network based stress update models. The strategy is useful in different contexts, including the following two scenarios:

- *From physical experiments to numerical stress update procedure:* Samples of a material are subjected to different load paths in a series of physical experiments. The data is then used to train the proposed neural network based stress update procedure.

[*] Corresponding author.
E-mail addresses: w.g.dettmer@swansea.ac.uk (W.G. Dettmer), e.j.muttiozavala@swansea.ac.uk (E.J. Muttio), r.s.h.j.alhayki@swansea.ac.uk (R. Alhayki), d.peric@swansea.ac.uk (D. Perić).

https://doi.org/10.1016/j.cma.2023.116672
Received 5 February 2023; Received in revised form 6 October 2023; Accepted 27 November 2023

Finally, the trained model is integrated in finite element simulations and can be used to study real-world applications. It is noted that the design and execution of a sufficient number of suitable experiments represents a challenge that will require substantial research effort.

- *Numerical homogenisation:* The complex micro structure of a material is represented by a suitable numerical strategy (representative volume element RVE, see for instance [24]) and a series of numerical experiments are performed, where the stress responses on macro level are computed for several macro strain data sequences, *i.e.* load paths. The stress–strain data is used to train the proposed stress update model, which can then be used in finite element based analysis to study real-world applications. See, for instance, [4,25–28].

In both cases, the methodology to be proposed "bridges the gap" between experiments and finite element analysis, *i.e.* it makes the generally complicated and tedious formulation of an accurate constitutive model and its algorithmic implementation obsolete.

While elastic or hyper-elastic models can be based on feedforward neural networks, this is not possible for inelastic material behaviour which is generally history or path-dependent [29,30]. Hence, recurrent model architectures with suitable memory mechanisms must be used [6,25,31–33]. A key feature of the proposed methodology is its basic network architecture. While it fundamentally relies on internal state variables, it does not make use of any gated network elements such as Long-Short Term Memory (LSTM) networks or Gated Recurrent Units (GRU). The latter are key ingredients of, for instance, the strategy proposed by Bonatti and Mohr [32] that also presents a framework suitable for generic material behaviour. It is shown that the proposed methodology requires fewer network parameters and possesses an accurate memory of the internal material state. The proposed model is trained on a parallel computer hardware with multiple instances of gradient-free optimisation procedures that are organised in a worker–supervisor framework recently developed by the authors and presented in detail in [34]. In its present form the proposed methodology is restricted to rate-independent material behaviour, but otherwise generically applicable. Therefore and despite the focus of the examples presented in this article on elasto-plasticity, the strategy is formally capable of modelling anisotropy, elasticity, plasticity and damage mechanics. The proposed methodology features the same inputs and outputs as any standard algorithmic continuum mechanical stress update procedure. It is therefore suitable for the integration in a finite element setting. However, this exceeds the scope of the present article and will be discussed in a forthcoming publication. Other strategies that share the format of standard continuum mechanical stress update algorithms include [18,23,32].

This article is organised as follows: The proposed stress update procedure is presented in Section 2. The architecture of the neural networks, the recurrent nature of the algorithm and their implications on the training strategy are also described. Moreover, a computationally efficient criterion for the verification of thermodynamic consistency is proposed. In Section 3, it is shown that the strategy is capable of representing *exactly* the standard stress integration algorithm for elasto-plasticity in the uniaxial case. Section 4 describes the network training strategy adopted in this work. In Section 5, it is shown that the training process recovers the *exact* stress data for uniaxial elasto-plasticity. Applications of the strategy to other nonlinear problems are presented in Sections 6 and 7, namely uniaxial elasto-plastic damage and plane strain elasto-plasticity. Conclusions are summarised in Section 8.

2. Neural network based constitutive model

The mechanical response of inelastic materials is path-dependent, *i.e.* the local stress depends on the current deformation and on its history. Hence, physically or phenomenologically based constitutive models are traditionally formulated which provide evolution laws for sets of carefully chosen internal variables.

In Section 2.1, a *neural network based* algorithmic constitutive model, or stress update procedure, for rate-independent inelastic material is proposed that also uses the concept of internal variables to account for the history dependency of the material response. However, the methodology is designed without any knowledge of the material to be represented. The internal variables do not possess any known physical meaning and their nature and evolution are left entirely to the training process. As described in Section 1, the training data may be available from physical experiments or from numerical homogenisation. The network architecture employed in this work and its implications on the network training process are described in Sections 2.2 and 2.3, respectively, while a criterion for the thermodynamic consistency of the model is derived in Section 2.4.

2.1. Neural network based stress update

The dependency of the stress components $\boldsymbol{\sigma}$ on the strain components $\boldsymbol{\varepsilon}$ and on the scalar internal variables $\boldsymbol{\xi} = \{\xi^{(1)}, \xi^{(2)}, \dots, \xi^{(M)}\}$, is expressed as

$$\boldsymbol{\sigma} = \mathscr{F}\left(\boldsymbol{\varepsilon}, \boldsymbol{\xi}\right) , \tag{1}$$

where $\mathscr{F}$ represents a suitable feedforward artificial neural network. For rate-independent materials the update of the internal variables from time instant t_n to time instant t_{n+1} is written as

$$\boldsymbol{\Delta\xi} = \mathscr{G}\left(\boldsymbol{\varepsilon}_{n+1}, \boldsymbol{\varepsilon}_n, \boldsymbol{\xi}_n\right) \tag{2}$$

and

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}_n + \boldsymbol{\Delta\xi} , \tag{3}$$

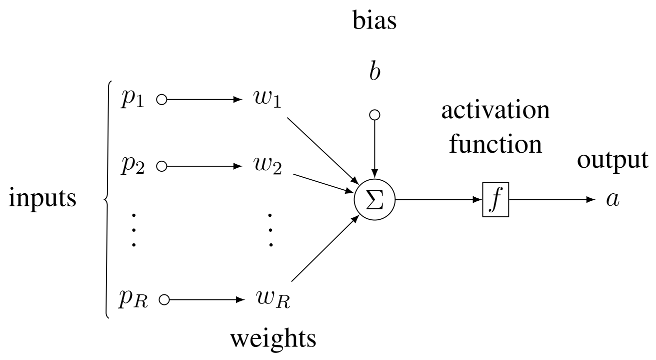


Fig. 1. Artificial neuron.

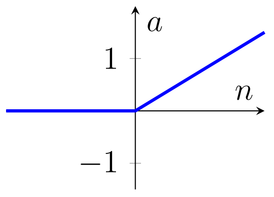


Fig. 2. ReLU activation function.

where $\mathscr{G}$ represents a second suitable feedforward artificial neural network. Given the new strains ε_{n+1} and the historic quantities ε_n and ξ_n, the complete stress update procedure consists of evaluating Eqs. (2) and (3) followed by applying Eq. (1) to the quantities at t_{n+1}, *i.e.*

$$\sigma_{n+1} = \mathscr{F}\left(\varepsilon_{n+1}, \xi_{n+1}\right) . \tag{4}$$

In the following, the Networks $\mathscr{G}$ and $\mathscr{F}$ are referred to as the *state* and the *response* networks, respectively.

2.2. Neural network architecture

The artificial state and response neural networks $\mathscr{F}$ and $\mathscr{G}$ consist of an input layer, a small number of fully connected hidden layers and an output layer. The response network $\mathscr{F}$ may not possess any hidden layers but correspond to a basic linear operator. Fig. 1 shows one of the neurons used in a hidden layer.

The generic internal process of a neuron is described by

$$a_i = f(s_i) \quad \text{with} \quad s_i = \sum_j^R w_{ij}\, p_j + b_i \tag{5}$$

where a_i is the output of the neuron i, $f(\bullet)$ is the activation function, R is the number of inputs, w_{ij} are the weights connected to the inputs p_j and b_i is the bias term. In this work the piecewise linear function known as the Rectified Linear Unit (ReLU) and presented in Fig. 2 is used as activation function, *i.e.*

$$f(s) = \max(0, s) . \tag{6}$$

Output neurons are pure linear operators, *i.e.* they are similar to neurons in the hidden layers, but do not possess an activation function.

Fig. 3 illustrates the stress update procedure as defined by Eqs. (2) to (4) for the uniaxial load case where stresses and strains are each represented by a single coefficient. In the figure, the state network $\mathscr{G}$ possesses one hidden layer of neurons and the response network $\mathscr{F}$ consists only of the input and output neurons. For the three dimensional case, the stresses and strains are represented by six coefficients each and the number of input and output neurons must be increased accordingly. For the plane strain case, three strain coefficients induce four stress coefficients and, again, the number of network inputs and outputs must be adjusted accordingly.

Remark on physical constraints: The networks $\mathscr{F}$ and $\mathscr{G}$ are subject to the following *physically* motivated constraints:

1. If all inputs are zero, all outputs of the state network $\mathscr{G}$ must also be zero. This results in a dependency of the biases of the output neurons on network weights and the biases in the hidden layers.

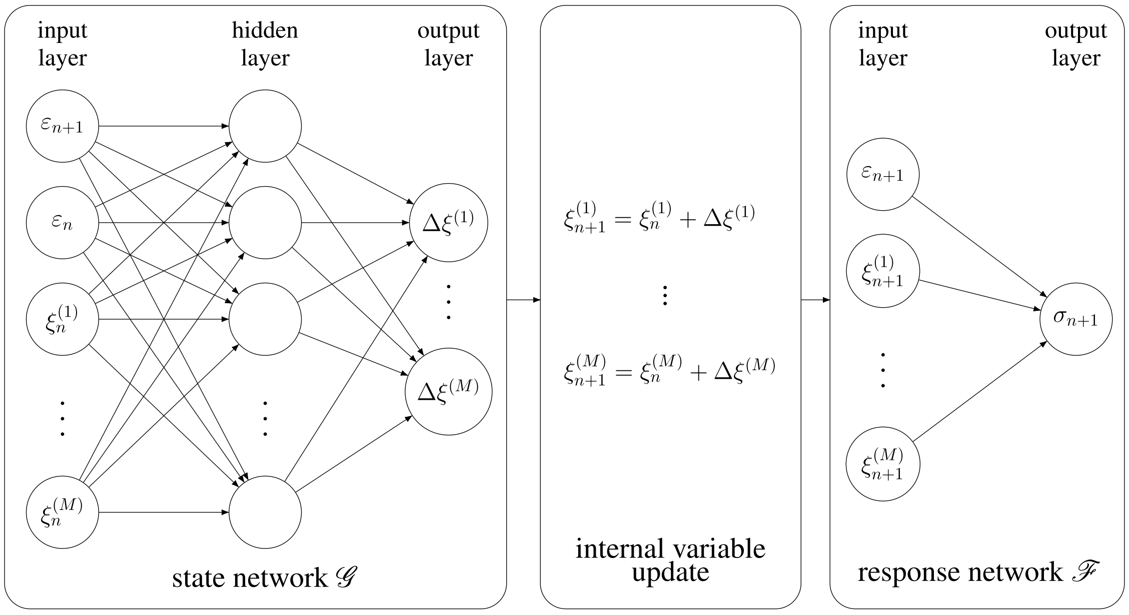


Fig. 3. Stress update procedure based on the state and response neural networks $\mathscr{G}$ and $\mathscr{F}$.

2. If all inputs are zero, all outputs of the response network $\mathscr{F}$ must also be zero. This results in a dependency of the biases of the output neurons on the network weights and the biases in the hidden layers. In the absence of any hidden layers, the biases in the output neurons must be zero.
3. If the new strain ε_{n+1} is equal to the previous strain ε_n (in all its components), then all outputs of the state network $\mathscr{G}$ must be equal to zero. If this constraint was satisfied a priori, the memory of the internal material state would be exact.

In this work, the first and the second constraint are implemented, *i.e.* the output biases are eliminated from the training process. The third constraint is more complex and not accounted for in this work. Nevertheless, the proposed model is shown in Section 7.7 to possess superior capability of memorising the internal material state.

2.3. Recurrent structure and training

In order to represent accurately the behaviour of the material under consideration, the system must be *trained*. The data used for the training must represent the material behaviour sufficiently accurately and comprehensively. During the process of training suitable values for the weights and biases of the state and response networks $\mathscr{F}$ and $\mathscr{G}$ are determined. Prior to designing an appropriate training strategy the nature of the overall system must be considered carefully.

Training on data sequences: Despite introducing the neural networks $\mathscr{F}$ and $\mathscr{G}$ as feedforward networks, the *recurrent* nature of the overall strategy is obvious: The updated internal variables which result from network output of the previous load step are used as network input in the next load step. Therefore, the internal variables play the role of hidden states similar to those in LSTM networks or in GRUs. It is impossible to train the system on a set of pairs of input and output data. Instead, the overall strategy must be trained on a number of sequences of stress–strain data. These data sequences must represent strain paths and the associated stress responses as obtained from numerical or physical experiments. Due to the unavailability of suitable information the internal variables are always initialised with the value of zero. Hence, the training data must be generated with experiments starting from unworked and undeformed material.

Avoiding exploding/vanishing gradients by evolutionary optimisation: It is well known that, in the context of recurrent networks, standard gradient-based training methods, such as backpropagation, suffer from exploding or vanishing gradients. These arise from the repeated application of the chain rule required to back-track the gradients through the entire training data series. If unchecked, exploding or vanishing gradients generally lead to the failure of the training process. In this work they are avoided by basing the training process on a gradient-free evolutionary optimisation procedure, *i.e.* the network weights and biases are sought such that the network response to the training strain sequences matches the associated training stress data as best as possible. Details of the evolutionary optimisation strategy adopted in this work are provided in Section 4.

Remarks on alternative strategies: Alternatively to the strategy proposed in this work, a methodology for the stress update can be based on LSTM networks or networks with GRUs, see for instance [32,35,36]. These interrelated techniques are today well established and involve hidden states that play the same role as the internal variables ξ in Eq. (1), *i.e.* they represent the memory of the material.

Crucially, due to built-in mechanisms of memory loss, such methodologies do not suffer from exploding or vanishing gradients and therefore allow for the employment of efficient gradient-based training strategies. The following remarks explain why it has been chosen not to employ them in this work:

- LSTM or GRU based networks add a significant amount of complexity. Numerical experimentation presented in Section 7.6 shows that this additional complexity is unnecessary. The proposed strategy renders accurate results based on fewer hidden states and network parameters.
- In Section 3 it is argued that the basic network architectures used in this work possess features that are advantageous specifically for constitutive modelling.
- The proposed strategy does not possess any direct mechanisms of memory loss, which are fundamental to LSTM or GRU based architectures. This is advantageous in the context of constitutive modelling where memory loss would be physically incorrect.
- The proposed strategy is designed for the integration in the finite element method for solid mechanics, where it replaces the traditional constitutive model based stress update algorithm on Gauß point level and is executed typically several thousand times in each load step. Hence, the relatively small and basic architecture of the proposed stress update model ensures computational efficiency. Moreover, existing implementations of LSTM or GRU networks, such as those offered by PyTorch or TensorFlow, are most easily used in the scripting language Python. Their efficient integration in programming language based finite element code is challenging.
- The only arguable disadvantage of the proposed strategy in comparison with LSTM or GRU based networks is the challenge associated with gradient-free training. Yet, gradient-free network training based on evolutionary optimisation procedures has been used in a number of publications with reference to a variety of applications, see for instance [37–42]. The strategy used in this work is described in Section 4.

2.4. Thermodynamic consistency

The thermodynamic consistency of a constitutive model is generally ensured if the Clausius–Duhem inequality is satisfied, see for instance [43]. In the small strain regime, this can be expressed as *i.e.*

$$D \;=\; \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} \;-\; \rho\dot{\Psi} \;\geq\; 0 \tag{7}$$

where D is the mechanical dissipation, while Ψ denotes the specific free energy, which depends on the internal variables, and ρ is the material density.

The algorithmic constitutive model proposed in Eqs. (2)–(4) does not provide access to the free energy function Ψ. Hence, the inequality in (7) cannot be used to ensure thermodynamic consistency of the proposed model during or after the training. Instead, it is suggested to test the thermodynamic consistency by the following argument:

The dissipation inequality in (7) is satisfied if the work done along any closed path in the deformation or strain space is non-negative, *i.e.*

$$D^\circ \;=\; \oint \boldsymbol{\sigma} : \mathrm{d}\boldsymbol{\varepsilon} \;\geq\; 0\,. \tag{8}$$

This inequality is equivalent to requiring that any hysteresis area in the stress–strain diagram is positive. The algorithmic stress update as given by Eqs. (2)– (4) is based on incrementing the total strain from $\boldsymbol{\varepsilon}_n$ to $\boldsymbol{\varepsilon}_{n+1}$. Hence, one can construct the following closed path in the strain space: After stepping from $\boldsymbol{\varepsilon}_n$ to $\boldsymbol{\varepsilon}_{n+1}$, a second step is performed, back to $\boldsymbol{\varepsilon}_n$. This is displayed qualitatively in Fig. 4. The dissipation in Eq. (8) can then be expressed as

$$D^\circ \;=\; \bar{\boldsymbol{\sigma}}_1 : \boldsymbol{\Delta\varepsilon}_1 \;+\; \bar{\boldsymbol{\sigma}}_2 : \boldsymbol{\Delta\varepsilon}_2 \;\geq\; 0\,, \tag{9}$$

where $\boldsymbol{\Delta\varepsilon}_1 = \boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_n$ and $\boldsymbol{\Delta\varepsilon}_2 = -\boldsymbol{\Delta\varepsilon}_1$. The stresses $\bar{\boldsymbol{\sigma}}_1$ and $\bar{\boldsymbol{\sigma}}_2$ represent appropriate average values. A second order approximation of the work done along this two-step path can be obtained by using

$$\bar{\boldsymbol{\sigma}}_1 = \frac{1}{2}\left(\boldsymbol{\sigma}_n + \boldsymbol{\sigma}_{n+1}\right) \qquad \text{and} \qquad \bar{\boldsymbol{\sigma}}_2 = \frac{1}{2}\left(\boldsymbol{\sigma}_{n+1} + \boldsymbol{\sigma}^*_n\right)\;, \tag{10}$$

where $\boldsymbol{\sigma}^*_n$ is the stress response of the model after returning to the origin of the path at $\boldsymbol{\varepsilon}_n$. Hence, Eqs. (9) and (10) reduce to

$$D^\circ \;=\; \frac{1}{2}(\boldsymbol{\sigma}_n - \boldsymbol{\sigma}^*_n) : (\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_n) \;\geq\; 0\,. \tag{11}$$

Since it is based on two finite steps, this inequality is weaker than the expression given in Eq. (7). Yet, it can be easily implemented and requires only the evaluation of the model response to an additional fictitious step in the strain space. Fig. 5 demonstrates the relation between the expressions D and D° based on the example of uniaxial elasto-plasticity.

In the remainder of this work, Eq. (11) is used to test the thermodynamic consistency of the trained constitutive models. Moreover, the integration of the criterion into the training process may help to discard inconsistent network parameters at an early stage and thereby accelerate the training process. This will be the subject of future investigation.

3. Exact representation of uniaxial elasto-plasticity

Uniaxial von Mises elasto-plasticity can be represented *exactly* with the proposed neural network based strategy. The capability to exactly represent piecewise linear constitutive behaviour is an inherent property of the strategy and is demonstrated in the following.

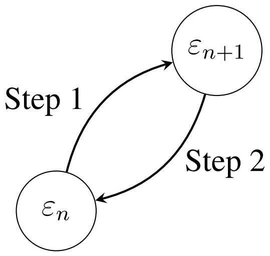


Fig. 4. Closed path in the strain space consisting of two steps.

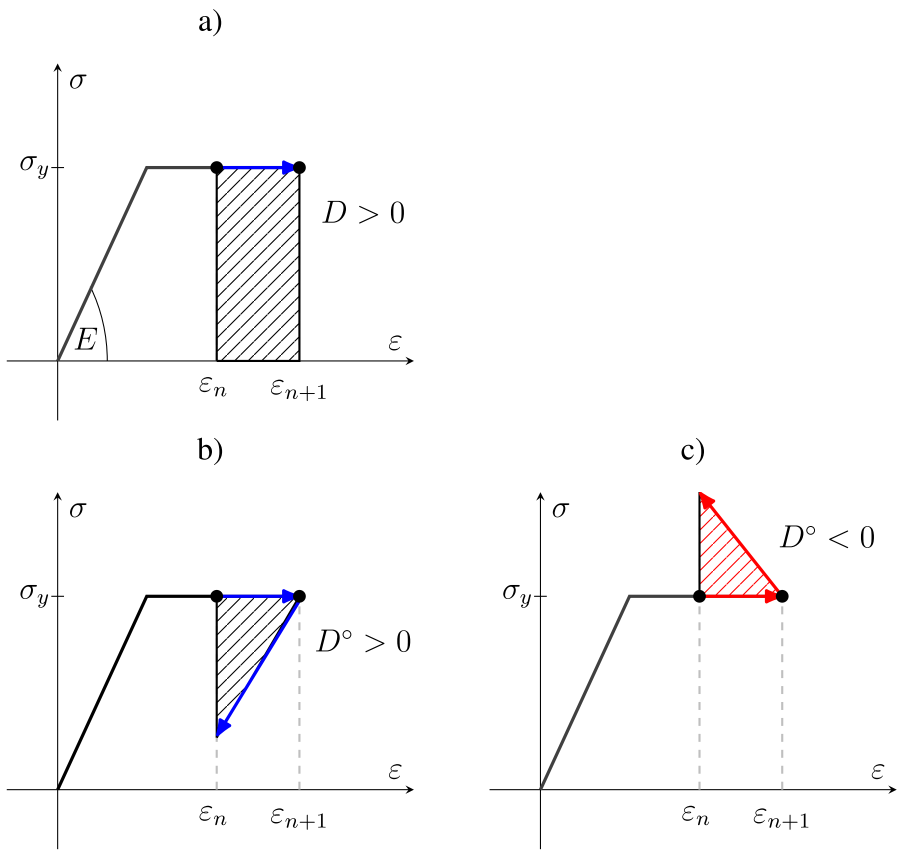


Fig. 5. A fully plastic step in perfect uniaxial elasto-plasticity with $\varepsilon^{(i)}_{n+1} - \varepsilon^{(i)}_{n} = \varepsilon_{n+1} - \varepsilon_n$: (a) dissipation $D = \sigma_Y\,(\varepsilon^{(i)}_{n+1} - \varepsilon^{(i)}_{n})$ from Eq. (7), (b) dissipation $D^\circ = \frac{E}{2}\,(\varepsilon^{(i)}_{n+1} - \varepsilon^{(i)}_{n})^2$ from Eq. (11), (c) dissipation D° violating Eq. (11).

3.1. Standard return mapping algorithm

The widely used return mapping algorithm for von Mises elasto-plasticity with linear isotropic hardening is described in detail for instance in [29,30]. For the uniaxial case it reduces to the following expressions for the update of the stress and the internal variables

$$\sigma^{\text{trial}} = E\,\left(\varepsilon_{n+1} - \varepsilon^{(p)}_{n}\right) \tag{12}$$

$$\Phi^{\text{trial}} = |\sigma^{\text{trial}}| - (\sigma_Y + H\,\alpha_n) \tag{13}$$

$$\Delta\gamma = \left\langle \frac{\Phi^{\text{trial}}}{E+H} \right\rangle \tag{14}$$

$$\Delta\varepsilon^{(p)} = \Delta\gamma\,\operatorname{sign}(\sigma^{\text{trial}}) \tag{15}$$

$$\Delta\alpha = \Delta\gamma \tag{16}$$

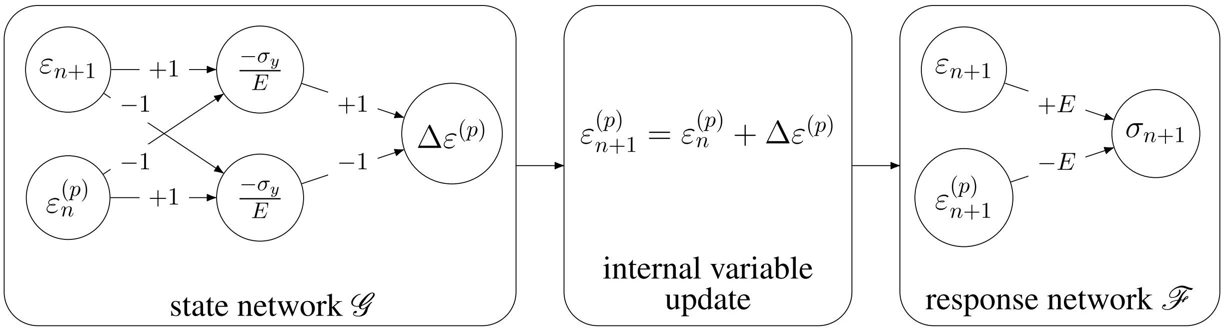

ε_{n+1}
$+1$
-1
$\frac{-\sigma_y}{E}$
$\varepsilon_n^{(p)}$
-1
$+1$
$\frac{-\sigma_y}{E}$
$+1$
-1
$\Delta\varepsilon^{(p)}$
state network $\mathscr{G}$
$\varepsilon_{n+1}^{(p)} = \varepsilon_n^{(p)} + \Delta\varepsilon^{(p)}$
internal variable update
ε_{n+1}
$\varepsilon_{n+1}^{(p)}$
$+E$
$-E$
σ_{n+1}
response network $\mathscr{F}$

Fig. 6. Uniaxial perfect elasto-plasticity model in network form.

$$\varepsilon_{n+1}^{(p)} = \varepsilon_n^{(p)} + \Delta\varepsilon^{(p)} \tag{17}$$

$$\alpha_{n+1} = \alpha_n + \Delta\alpha \tag{18}$$

$$\sigma_{n+1} = E\left(\varepsilon_{n+1} - \varepsilon_{n+1}^{(p)}\right) \tag{19}$$

where E, σ_Y and H are, respectively, the Young's modulus, the yield stress and the hardening modulus. The plastic strain is represented by $\varepsilon^{(p)}$, while the plastic increment is denoted by $\Delta\gamma$ and α denotes the accumulated plastic strain. The ramp function is defined as $\langle\bullet\rangle = \max(0,\bullet)$.

3.2. *Perfect elasto-plasticity*

For perfect elasto-plasticity with $H = 0$ the accumulated plastic strain can be omitted. Eq. (15) can then be written as

$$\Delta\varepsilon^{(p)} = \operatorname{sign}\left(\varepsilon_{n+1} - \varepsilon_n^{(p)}\right)\left\langle\left|\varepsilon_{n+1} - \varepsilon_n^{(p)}\right| - \frac{\sigma_Y}{E}\right\rangle , \tag{20}$$

which is equivalent to

$$\Delta\varepsilon^{(p)} = +\left\langle\varepsilon_{n+1} - \varepsilon_n^{(p)} - \frac{\sigma_Y}{E}\right\rangle - \left\langle\varepsilon_n^{(p)} - \varepsilon_{n+1} - \frac{\sigma_Y}{E}\right\rangle . \tag{21}$$

Recalling the correspondence between the ramp function and the ReLU activation function in Eq. (6) it follows that Eq. (21) can be represented *exactly* by the state network $\mathscr{G}$. By identifying the plastic strain as the internal variable, *i.e.* $\xi = \varepsilon^{(p)}$, and by using the linear expression for the stress from Eq. (19) the return mapping scheme can be cast exactly in the format of the proposed neural network based stress update procedure. This is shown in Fig. 6 where the network weights are displayed on the arrows and the biases are shown inside the circles which represent the hidden neurons. Note that the input of ε_n to the state network $\mathscr{G}$ is not required here. Fig. 7 shows a typical example of the well known stress–strain behaviour of the system in which it is evident that the neural network prediction matches *exactly* the stress evolution. Note that the weights and biases shown in Fig. 6 are not unique. There exists an infinite number of alternative sets of network parameters that may change the nature of the internal variable but result in exactly the same stress response. The process of network training will converge to any of these sets of parameters.

3.3. *Isotropic linear hardening elasto-plasticity*

The transformation of the system of Eqs. (12) to (19) with $H > 0$ into a format suitable for the design of the equivalent neural network based stress update is more tedious than for the perfectly elasto-plastic case, but still a straightforward exercise. For the sake of brevity it suffices to show the resulting network based system in Fig. 8. Clearly, two internal variables are required, *i.e.* $\xi^{(1)} = \varepsilon^{(p)}$ and $\xi^{(2)} = \alpha$. The well-known response of the system to symmetric strain cycling is shown in Fig. 9 which confirms that, for the case of uniaxial linear hardening elasto-plasticity, the proposed neural network represents *exactly* the stress evolution data.

4. Network training

In the process of training, the network weights and biases are determined that ensure the accurate fitting of the training data sequences. The authors' choice of a gradient-free training strategy based on evolutionary optimisation is motivated in Section 2.3. The methodology adopted in this work employs a Multi-Objective Particle Swarm Optimisation (MOPSO) strategy which is embedded in a wrapper function that controls or supervises a number of parallel and sequential instances of the MOPSO algorithm. Prior to describing the strategy in detail, the cost or objective functions $J_s^{(k)}(x)$ are defined, where $x = \{w_{ij}, b_i\}$ represents the weights and biases of the state and response networks. In this work, the cost functions are

$$J_s^{(k)}(x) = \sqrt{\sum_{n=1}^{N_s}\left(\sigma_n^{(k)}(x) - \hat{\sigma}_n^{(k)}\right)^2} \tag{22}$$

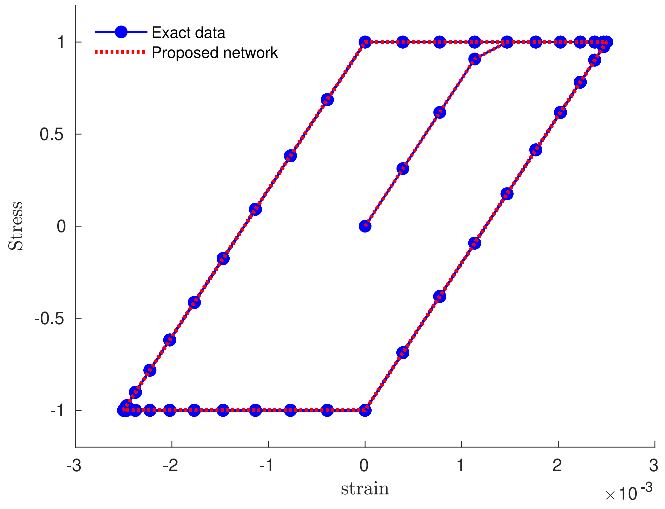


Fig. 7. A typical load cycle for uniaxial perfect elasto-plasticity.

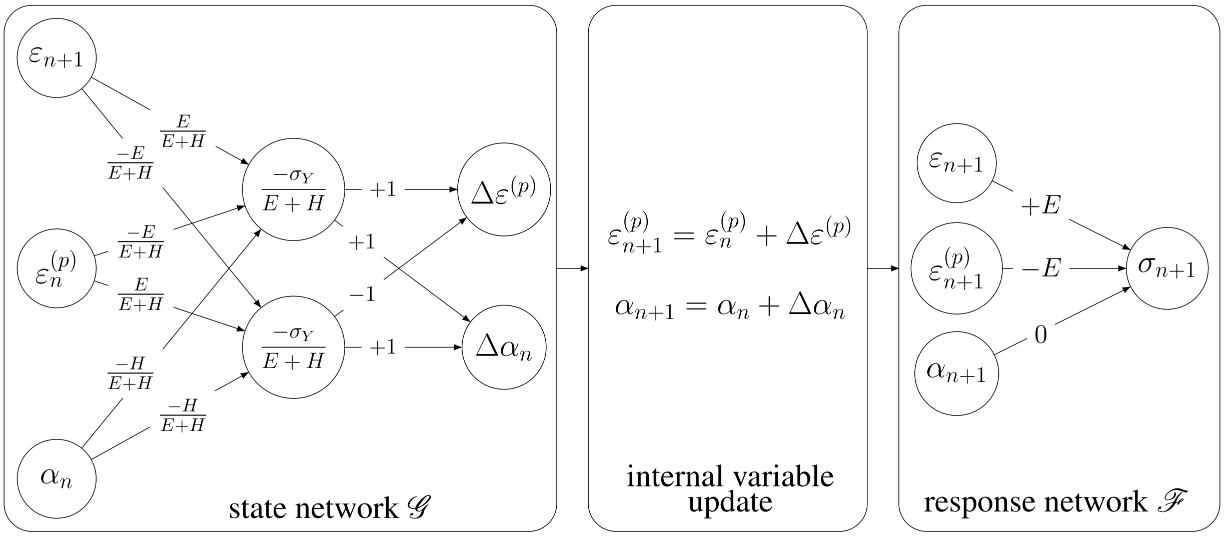


Fig. 8. Uniaxial elasto-plasticity model with linear isotropic hardening in network form.

where the subscript s identifies the training data series, *i.e.* the experiment, under consideration, while the superscript k identifies the stress coefficient. Furthermore, N_s denotes the number of load steps in the data series, while the terms $\sigma_n^{(k)}(x)$ and $\hat{\sigma}_n^{(k)}$ represent, respectively, the network based stress response to the strain data sequence and the given stress data that must be fitted.

4.1. *Supervised parallel and sequential evolutionary optimisation*

The challenges in the optimisation problem set by the training of the proposed methodology are manifold: The number of design variables is large (there are many weights and biases to be determined), the number of objectives is large (several stress–strain data series, *i.e.* experiments, with several spatial coefficients, must be approximated) and the problem is highly nonlinear. Such problems often feature several or many local minima which cause premature convergence of numerical optimisation strategies and make the task of detecting the global minimum difficult.

In fact, the authors' experimentation with the training of the proposed strategy by means of evolutionary optimisation clearly suggests the presence of a large number of local minima of the approximation error, *i.e.* cost function given in Eq. (22). The particle swarm strategies or genetic algorithms employed by the authors converge to many different configurations associated with significantly different levels of approximation errors.

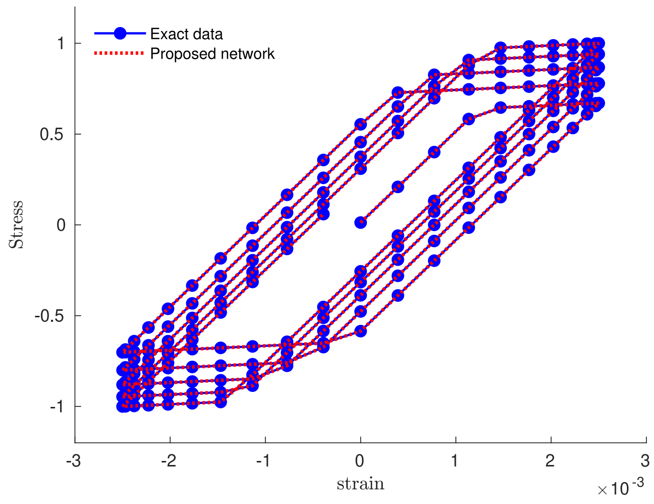


Fig. 9. Typical load cycles for uniaxial elasto-plasticity with isotropic linear hardening.

Therefore, in order to detect premature convergence at an early stage and run large numbers of optimisation procedures efficiently, the authors have developed and implemented a strategy for supervised parallel and sequential evolutionary optimisation. It is described briefly below, while a detailed presentation and assessment are provided in [34].

On a parallel computing platform with N_{CPU} CPUs, one processor is the supervisor, while the others are grouped into N_{team} teams. Each team works together on the particle swarm optimisation procedure described in Section 4.2. This is schematically displayed in Fig. 10. The supervisor communicates with the team leaders and thereby repeatedly starts, monitors and stops the optimisation procedures. In this way $N_{\text{run}} > N_{\text{team}}$ instances of optimisation procedures are executed in parallel as well as sequentially. Each new instance of the particle swarm method is initialised with a swarm consisting of randomly generated particles.

The crucial feature of this strategy is the mechanism for the supervisor's decision to stop a particular instance of the particle swarm method:

The team leaders report regularly (every 100 generations) the cost and position of their best particle to the supervisor. If the supervisor does not observe sufficient improvement in the cost received from a particular team in the last N_{stall} messages, then the supervisor regards the optimisation process as stalled, *i.e.*

$$\frac{\epsilon_m}{\epsilon_{m-N_{\text{stall}}}} > 1 - tolerance \qquad \Rightarrow \qquad \text{Optimisation has stalled.} \tag{23}$$

In Eq. (23), ϵ_m represents the average of the cost values $J_s^{(k)}$ from Eq. (22) received by the supervisor in the mth message from the corresponding optimisation team. Importantly, N_{stall} is related to the current error level ϵ_m, *i.e.*

$$N_{\text{stall}} = \bar{N}_{\text{stall}} \left(\frac{\bar{\epsilon}}{\epsilon_m} \right)^p , \tag{24}$$

where $\bar{N}_{\text{stall}} = 5$ and $\bar{\epsilon}$ are reference values and the exponent $p > 0$ is typically chosen as 1, 2 or 3. It is beneficial to set $\bar{\epsilon}$ to the value of the error associated with the first instance of stall based on $N_{\text{stall}} = \bar{N}_{\text{stall}}$. When the supervisor detects that a certain optimisation process has stalled, it compares the error ϵ_m to the error values recently reported by the other optimisation processes. If it is one of the $N_{\text{keep}} < N_{\text{team}}$ better performing processes it is allowed to continue, otherwise the supervisor requests that it is stopped and a new optimisation process is started until N_{run} procedures have been completed.

The methodology outlined here has proven crucial to the success of the proposed stress update strategy. It is currently further developed by the authors and will be reported in more detail in a forthcoming publication. Alternative strategies for combining several instances of optimisation procedures are described, for instance, in [44–50].

4.2. Multi-objective particle swarm optimisation

The multi-objective particle swarm optimisation strategy employed in this work evolved from extensive numerical experimentation of the authors with different methodologies. While it is currently being further developed and not the focus of this work, selected key features of the strategy are described in the following paragraphs. Other strategies may be as suitable or more suitable for the training of the proposed stress update procedure. A survey of particle swarm based strategies for multi-objective optimisation is provided in [51].

Continuous weight aggregation: If S denotes the number of data series, it i.e. experiments, available for training and K is the number of stress coefficients ($K = 1$, 4 or 6 for, respectively, uniaxial, plane strain or three dimensional case), then the number of objective

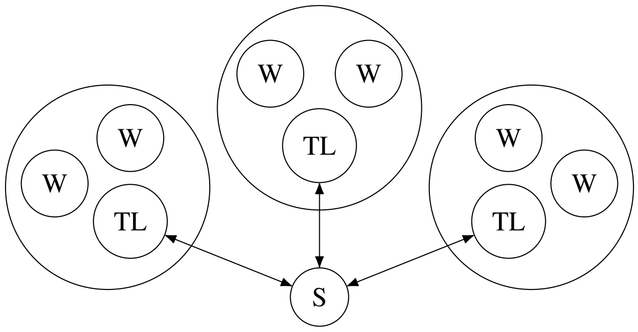


Fig. 10. The different roles of the CPUs in the proposed supervised optimisation strategy: supervisor (S), team leaders (TL) and workers (W). The example shows $N_{\text{CPU}} = 10$ CPUs and $N_{\text{team}} = 3$ teams.

functions represented by Eq. (22) is $N_{\text{obj}} = S \times K$. A randomised but continuous and smooth weight aggregation is used which repeatedly, but in varying order, shifts the focus of the optimisation to each single objective as well as to different weighted averages. While the objective of the training process is the minimisation of the average approximation error, the continuous shifting of the emphasis to different areas of the pareto front is found to significantly accelerate the overall convergence.

Selection of best particles and parallelisation: The swarm is distributed equally over the CPUs that constitute the team as shown in Fig. 10. Before each velocity and position update the team leader collects the best particles from all workers, determines the global best and sends it back to each worker. In each generation the best particles are selected with respect to the current weighting. A smooth and slow variation of the weighting factors is ensured by repeatedly setting random target weights and using typically 100 generations to smoothly shift from one set of target weights to the next. The team leader controls the target weights and communicates them to the workers. A repository of best particles, *i.e.* the most advanced pareto front, is stored and managed by the team leader. The global best is occasionally found in this repository which keeps the swarm focused and prevents it from drifting away from good solutions obtained already.

Scaling of training data: Prior to each training process the stress data is scaled to the interval $[-1, +1]$ such that the value of zero is maintained. This can be achieved in three different ways: (i) Each stress coefficient series is scaled individually. This results in $N_{\text{obj}} = S \times K$ scaling factors. (ii) Alternatively, each stress coefficient is scaled uniformly across all experiments. This results in K scaling factors. (iii) Finally, all stress coefficients may be scaled in exactly the same way, which requires a single scaling factor. In the examples presented in this work the latter strategy is adopted, since the maximum and minimum values of all stress coefficients are of the same order of magnitude. For certain materials or applications, the extreme values of axial stresses may reach much larger values than those of the shear stresses. In this case the second scaling strategy may be more suitable.

Communication with supervisor: In regular intervals the team leader sends the current global best particle's cost and position to the supervisor and, in return, receives the request to continue or to restart the optimisation process, see Fig. 10.

5. Validation: Uniaxial elasto-plasticity

The described methodology is applied to uniaxial von Mises elasto-plasticity. It is shown in Section 3 that the standard stress integration scheme can be *exactly* represented by the proposed neural network based stress update strategy. Therefore, the purpose of the following numerical test is exclusively the validation of the training process. If it results in a model that does not exactly represent the data, the training procedure cannot be applied with confidence to more complex cases. In accordance with the continuum mechanical model given by Eqs. (12) to (19), two internal variables are used in the test. Therefore, the state network employed has four input neurons, one hidden layer with two neurons and two output neurons (4→2→2). The response network possesses three input and one output neurons only (3→1). Hence, the networks have the same architecture as shown in Fig. 8 except for the additional input of ε_n to the state network. The variables to be determined by training therefore comprise two biases and 15 weights.

5.1. Training and validation data

The standard return mapping algorithm shown in Eqs. (12) to (19) is used to generate the training and validation data. The parameters used are $E = 200$ GPa, $H = 10$ GPa and $\sigma_Y = 250$ MPa, which correspond to the properties of Steel A36. Fig. 11(a) and (b) show, respectively, 16 cyclic strain data series and the corresponding stress–strain diagrams as obtained from the standard stress integration algorithm. Each data series consists of 200 data points. Twelve data series are used for training and four for validation.

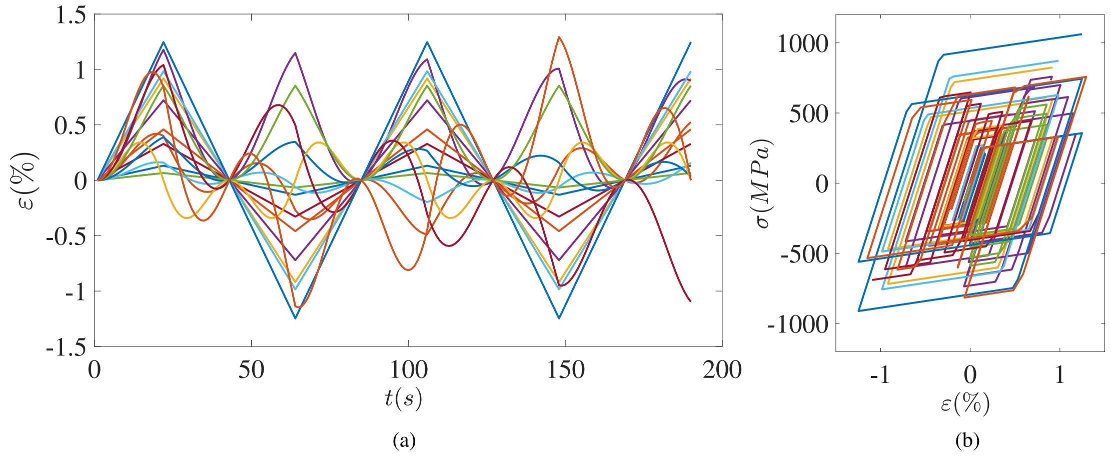


Fig. 11. Different strain sequences (a) and associated stress–strain diagrams (b) for uniaxial elasto-plasticity.

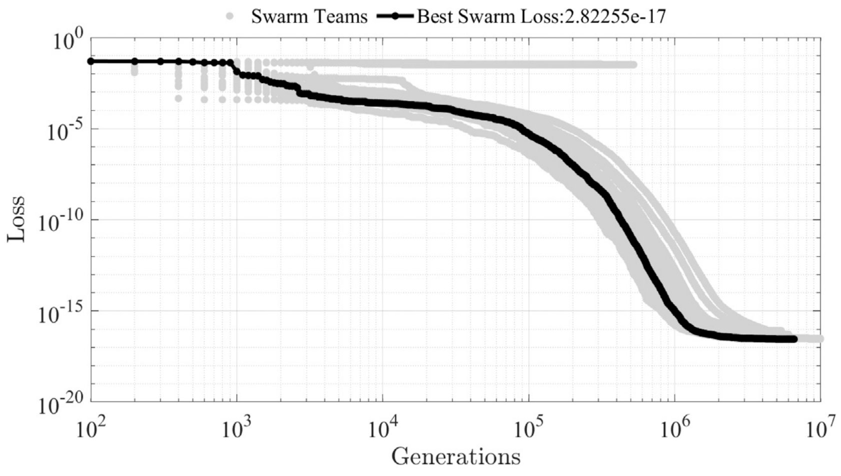


Fig. 12. Training convergence for uniaxial elasto-plasticity.

5.2. Convergence

The convergence of the average error during training is illustrated in Fig. 12. Clearly the approximation error reduces to the level of machine accuracy. Note that the training effort shown in the figure was exaggerated for demonstrational purposes, *i.e.* the training was aborted only when machine accuracy was reached. A highly accurate response, which may be deemed exact for practical purposes, is obtained within seconds or a few minutes on a standard laptop.

5.3. Results

As expected the training process renders a network based stress update procedure which generates *exactly* the same stress responses as the standard stress integration scheme. Figs. 13 to 16 show the agreement of the model response with the training and validation data. The evolution of the hidden internal variables is studied in Section 5.5.

5.4. Thermodynamic consistency

Based on the trained model the dissipation D° as defined in Eq. (11) can be evaluated at each step of any load path. Fig. 17 shows the resulting values of D° along four different strain data series. It is observed that, as expected, the largest positive values coincide with the load steps that induce large plastic deformation, while the largest negative values occur when the direction of the loading changes, *i.e.* during the first elastic load step following a series of plastic steps. However, since the maximum violation of the dissipation criterion from Eq. (11) lies well within machine accuracy, the tests do not reveal any thermodynamic inconsistency of the trained model.

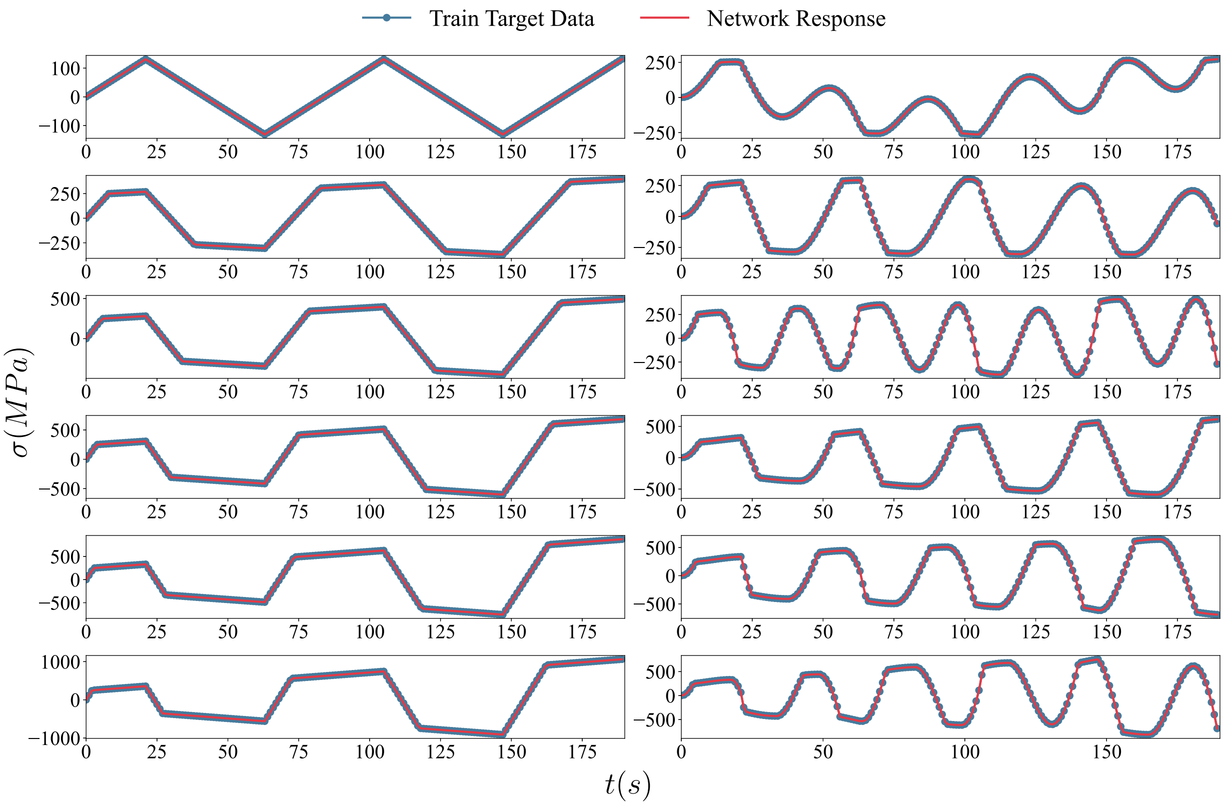


Fig. 13. Stress sequences for training of uniaxial elasto-plasticity.

5.5. *Internal variables*

For some of the training data sequences, the evolution of the internal variables $\xi^{(1)}$ and $\xi^{(2)}$ is shown in Fig. 18. The figure also shows the evolution of the plastic strain $\varepsilon^{(p)}$ and the accumulated plastic strain α as obtained from the data generation process based on Eqs. (12) to (19). The two sets of internal variables have been scaled to the interval $[-1,+1]$ to facilitate their comparison. A strong correlation between $\xi^{(1)}$ and $\varepsilon^{(p)}$ is easily observed and it may be argued that there exists some negative correlation between $\xi^{(2)}$ and α. The correlation factors are computed and displayed in Fig. 19, confirming a correlation between $\xi^{(1)}$ and $\varepsilon^{(p)}$ and between $\xi^{(2)}$ and α, respectively. This is consistent with the remark made in Section 3.2 that the network parameters that result in exact stress values are not unique and that the internal variables $\xi^{(1)}$ and $\xi^{(2)}$ can be different from $\varepsilon^{(p)}$ and α, but are likely to be somewhat aligned.

6. Example 1: Uniaxial elasto-plastic damage

The algorithmic return mapping scheme presented in [52] for elasto-plastic damage is adapted to the special case of uniaxial stress and used to generate the training and validation data. The algorithm is inherently nonlinear and cannot be represented exactly by the network based stress update strategy. Therefore this example tests the approximation capacity of the proposed approach as well as the performance of the training procedure. The number of internal state variables used is chosen identical to the number of internal variables required by the constitutive model, *i.e.* three internal states are used. The state network employed therefore has five input neurons, one hidden layer with nine neurons and three output neurons (5→9→3), while the response network requires four input neurons and one output neuron (4→1). The variables to be determined by training comprise nine biases and 76 weights.

6.1. *Training and validation data*

The parameters are set to the same values as in [52], *i.e.* $E = 21{,}000$ kN/cm^2, $H = 1500$ kN/cm^2, $\sigma_Y = 70$ kN/cm^2 and $r_0 = 0.117$ kN cm. Twelve cyclic strain data series are generated and the associated stresses are computed from the algorithmic stress update presented in [52]. Each data series consists of 200 data points. Eight data series are used for training, while the remaining four are used for validation. Figs. 22 and 24 show the training and validation data in terms of stress–strain diagrams.

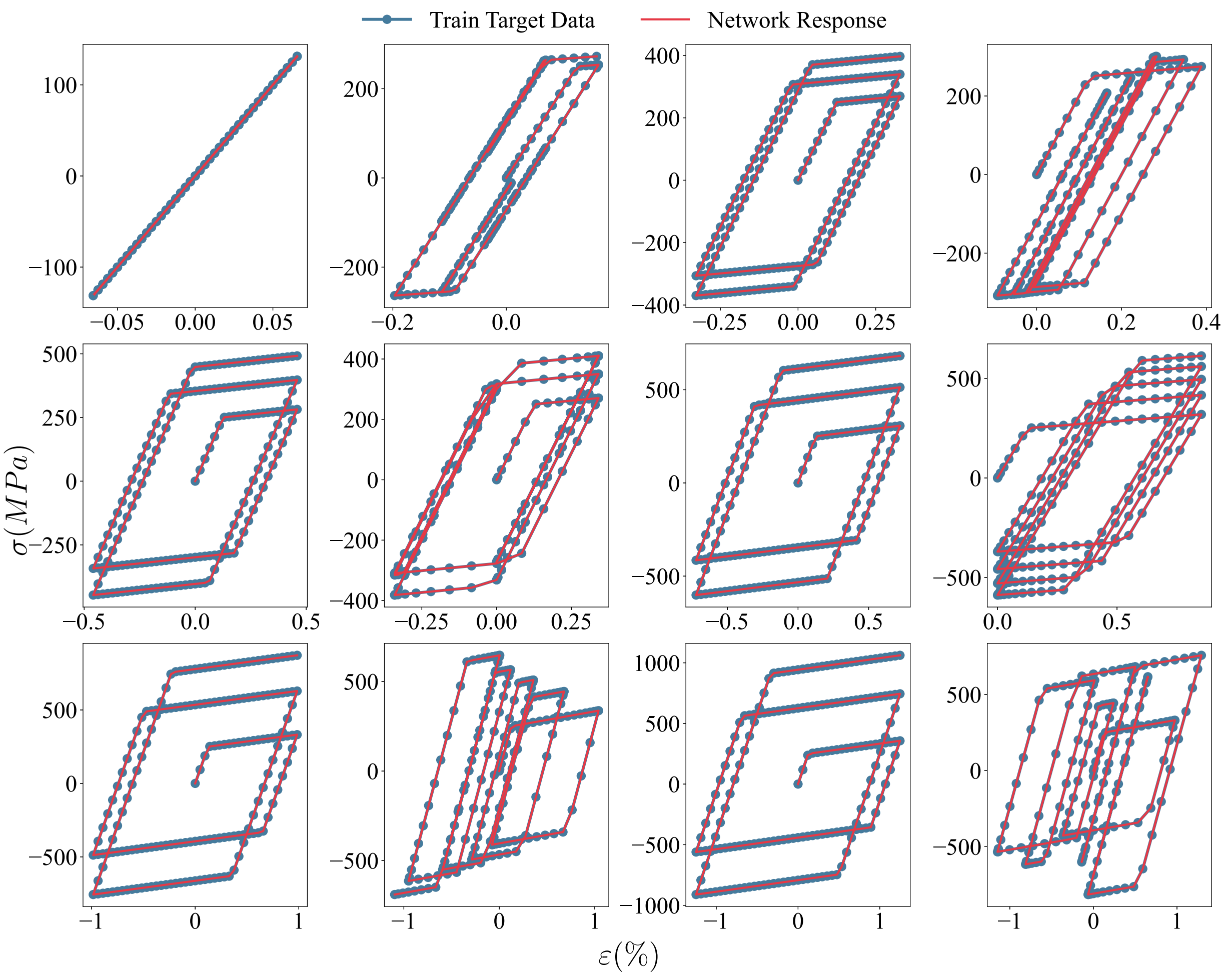


Fig. 14. Stress–strain diagrams for training of uniaxial elasto-plasticity.

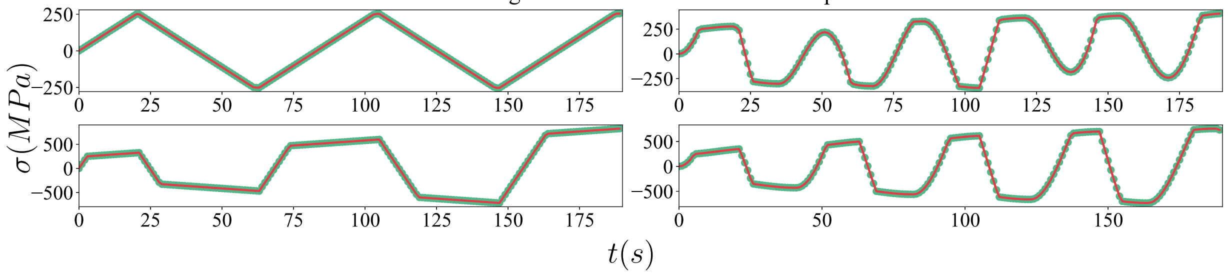


Fig. 15. Stress sequences for validation of uniaxial elasto-plasticity.

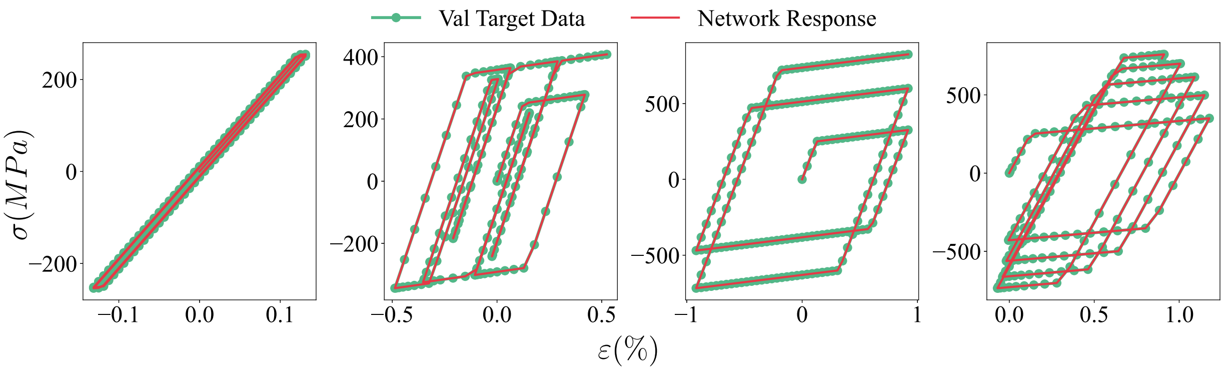


Fig. 16. Stress–strain diagrams for validation of uniaxial elasto-plasticity.

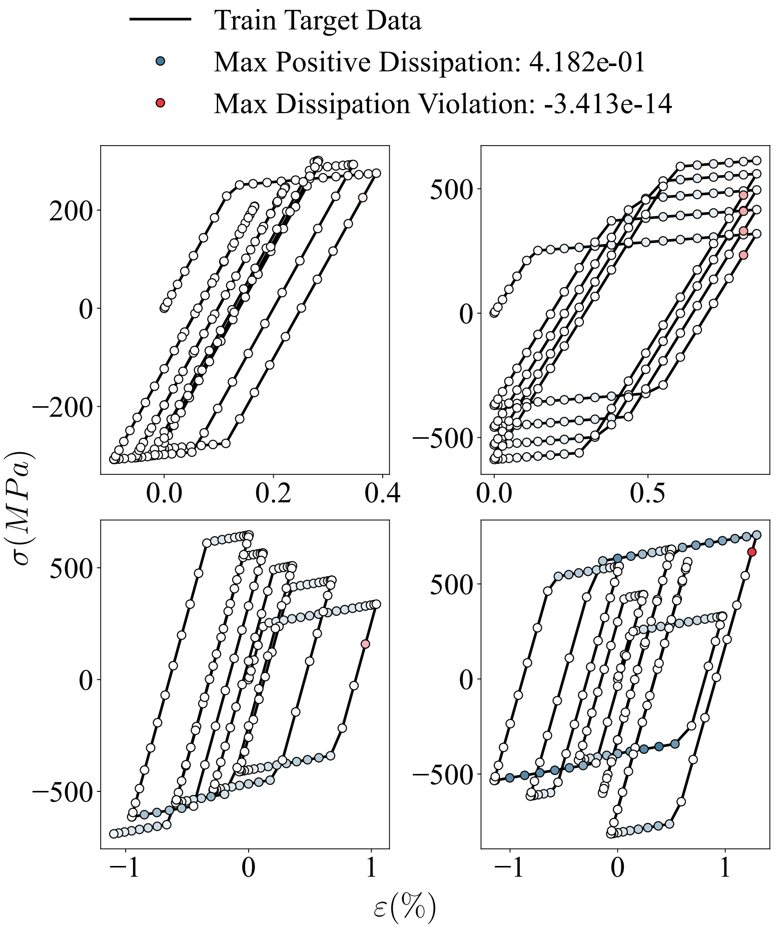


Fig. 17. Stress–strain diagrams with dissipation criterion for uniaxial elasto-plasticity.

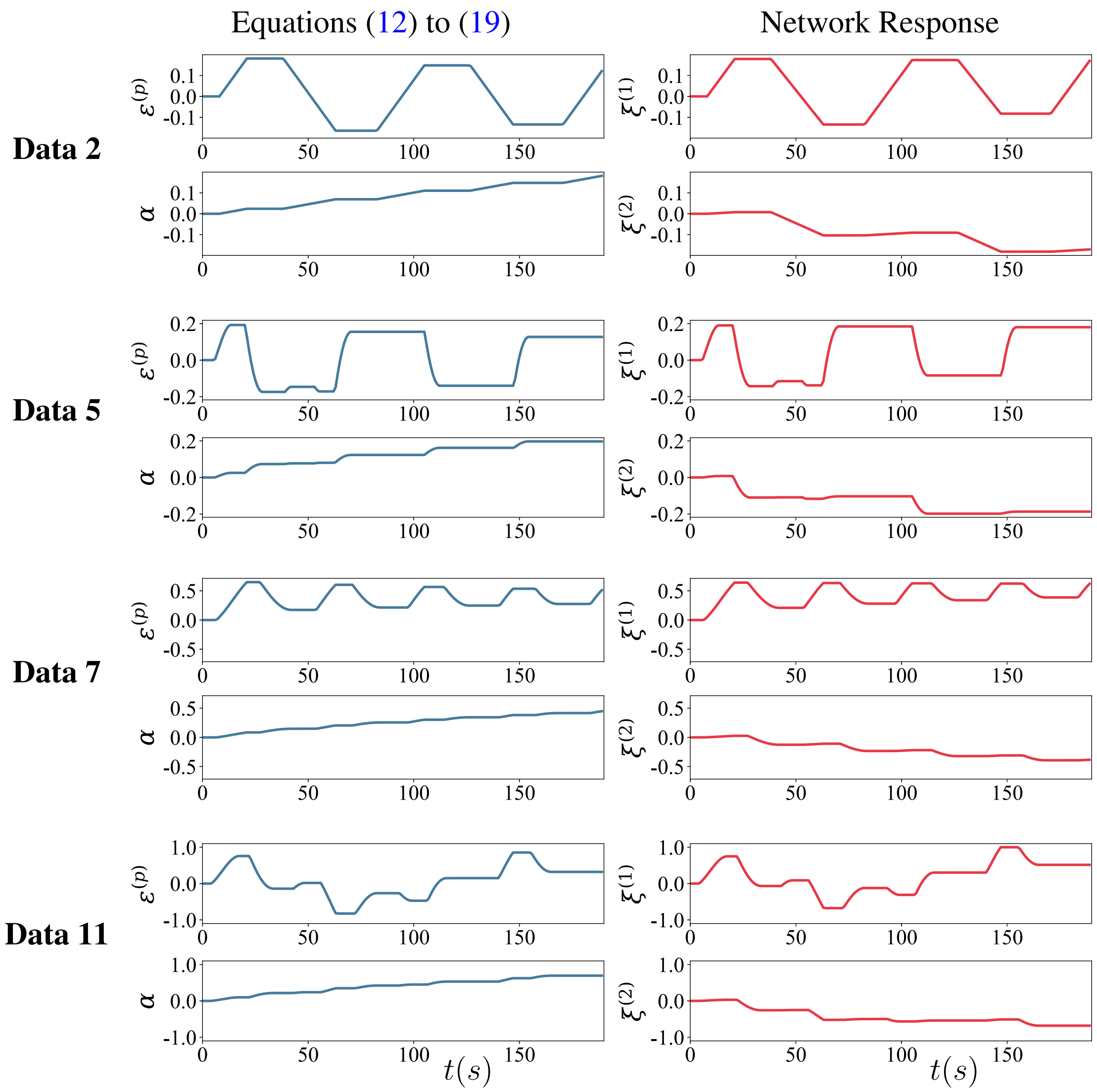


Fig. 18. Comparison of the internal variables for uniaxial elasto-plasticity.

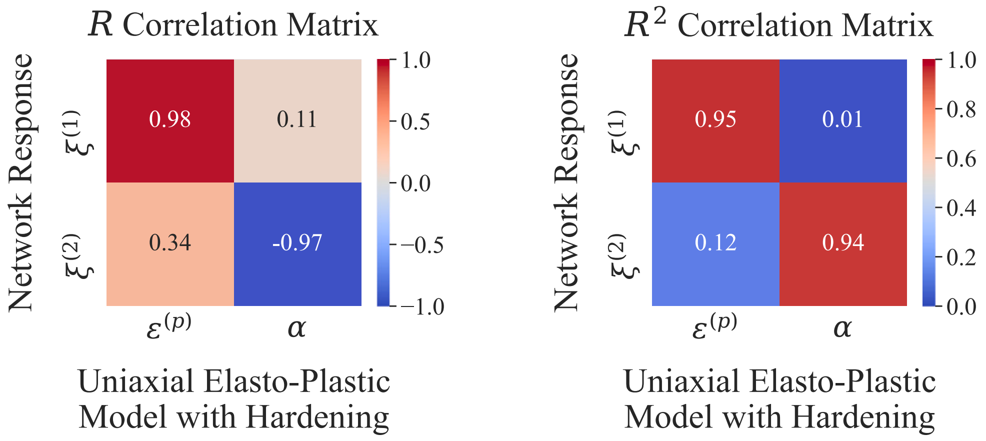


Fig. 19. Correlation analysis of the internal variables for uniaxial elasto-plasticity.

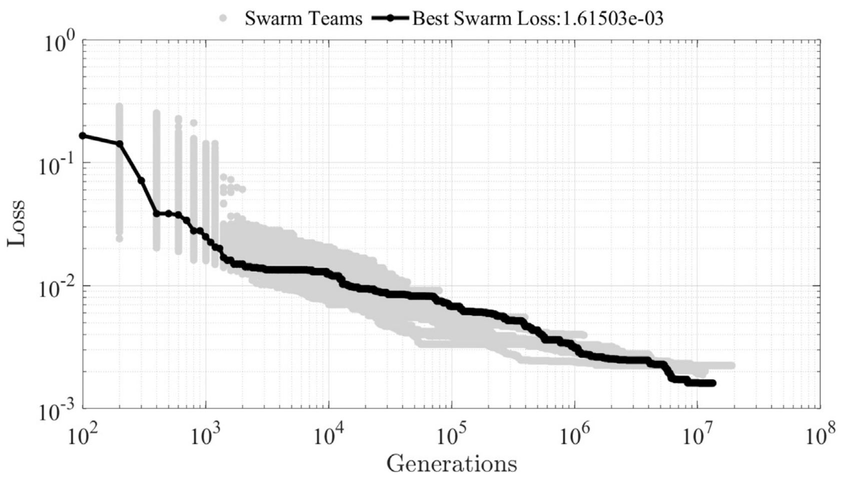


Fig. 20. Training convergence for uniaxial elasto-plastic damage.

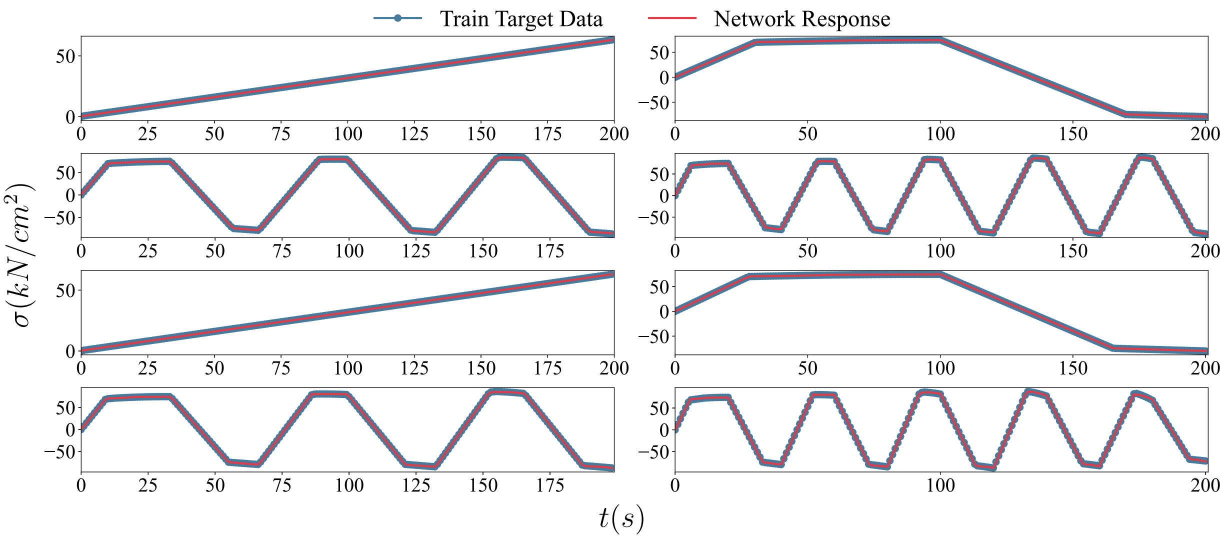


Fig. 21. Stress sequences for training of uniaxial elasto-plastic damage.

6.2. Convergence

The convergence of the average error during training is illustrated in Fig. 20. The training was performed on $N_{\text{CPU}} = 131$ CPUs of a high performance cluster with $N_{\text{team}} = 13$ and a team size of 10 CPUs and achieved the error level shown in the figure.

6.3. Results

Figs. 21 to 24 show the responses of the trained methodology together with the training and validation data. The agreement is observed to be of excellent accuracy. Recall that, as opposed to the case of uniaxial elasto-plasticity discussed in Section 5, the problem is not piecewise linear and the proposed methodology can only provide an approximation.

6.4. Thermodynamic consistency

The dissipation criterion proposed in Eq. (11) is evaluated for four data series and illustrated in Fig. 25. It can be observed that violation of the criterion is restricted to the load steps immediately following changes of loading direction. The largest negative value does not exceed 9.5% of the largest value of the dissipation D° computed for any of the dissipative load steps.

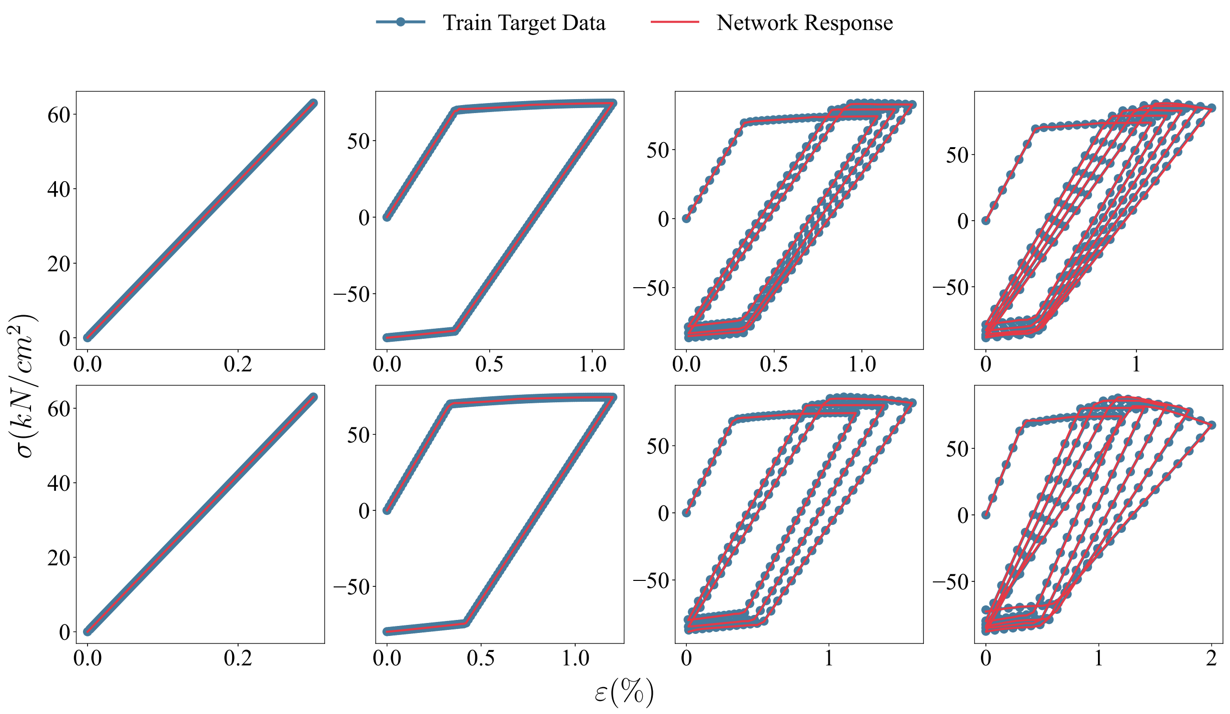


Fig. 22. Stress–strain diagrams for training of uniaxial elasto-plastic damage.

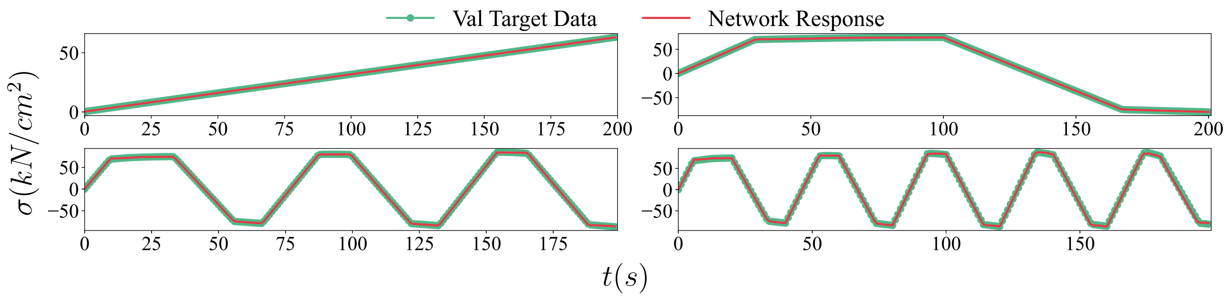


Fig. 23. Stress sequences for validation of uniaxial elasto-plastic damage.

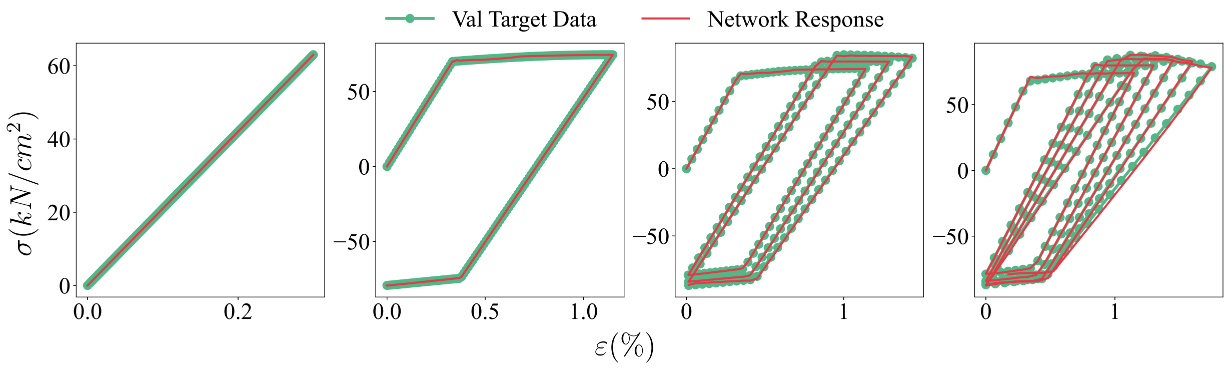


Fig. 24. Stress–strain diagrams for validation of uniaxial elasto-plastic damage.

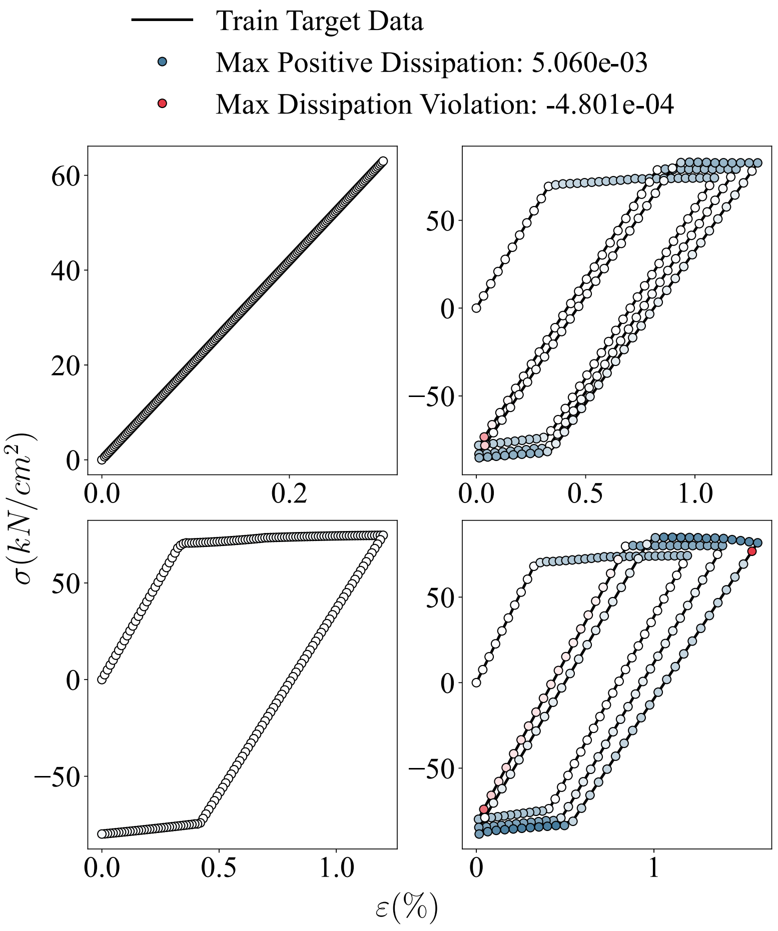


Fig. 25. Stress–strain diagrams with dissipation criterion for uniaxial elasto-plastic damage.

7. Example 2: Plane strain elasto-plasticity

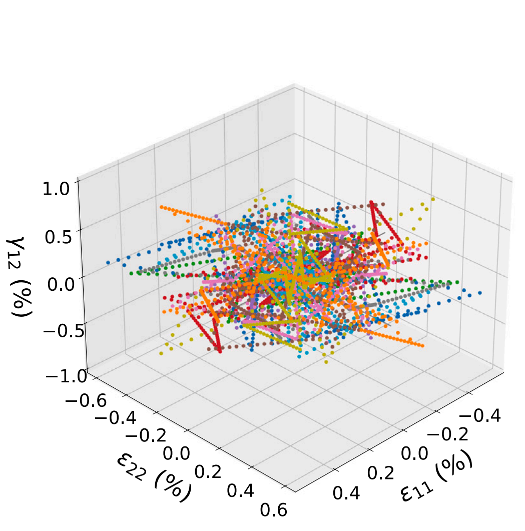
The return mapping algorithm for von Mises elasto-plasticity is presented, for instance, in [29,30]. For the multi-dimensional case it is inherently nonlinear due to the tensor norm operator that acts on the trial stress, and the proposed strategy cannot recover the stress and strain data exactly. Therefore, this example tests the approximation capacity as well as the quality of the training procedure. In the plane strain state, $\varepsilon_{yz} = \varepsilon_{zx} = \varepsilon_{zz} = 0$ and $\sigma_{yz} = \sigma_{zx} = 0$. Hence, three strain coefficients $\boldsymbol{\varepsilon} = \{\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy}\}$ induce a response consisting of four stress coefficients $\boldsymbol{\sigma} = \{\sigma_{xx}, \sigma_{yy}, \sigma_{xy}, \sigma_{zz}\}$. It is chosen to provide the network based stress update model with four internal state variables, thus allowing for the representation of some form of internal plastic strains. The state network therefore requires ten input neurons and four output neurons. The hidden layer is provided with 15 neurons (10→15→4). The response network requires seven input neurons and four output neurons (7→4). Similar to the problems presented in Sections 5 and 6 a hidden layer is not deemed necessary for the response network. It follows that 15 biases and 238 weights must be determined by the training process.

The following sections present details of the data generation, the convergence of the training process, the results and the evolution of the internal variables, but also include a comparison with alternative network architectures and demonstrate the effect of memory loss in strategies based on gated network elements.

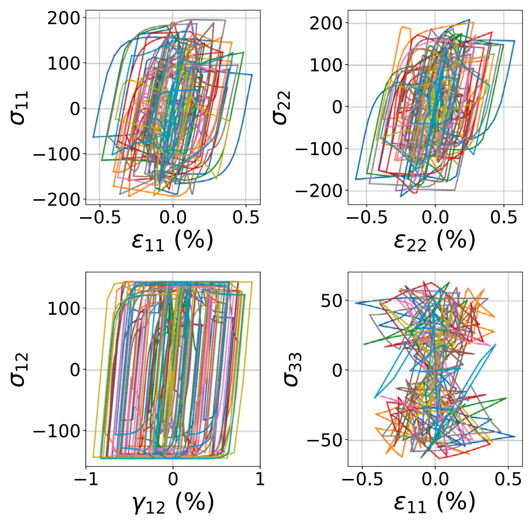
7.1. Training and validation data

The material parameters employed for the generation of the training and validation data are $E = 200$ GPa, $\nu = 0.26$ and $\sigma_Y = 250$ MPa. Strain hardening is not considered in this example, *i.e.* $H = 0$.

Similarly to Sections 5 and 6 the training and validation data consists of suitable strain data series and the associated stresses. The latter are computed from the strains with the return mapping scheme presented in [29,30]. The former need to be generated with care. If the trained model is to be robust and widely applicable, they must be representative of the entire physically feasible deformation space. Here, each strain data series is based on the linear interpolation between positions in the strain space that do not



(a)



(b)

Fig. 26. Different strain sequences (a) and associated stress–strain diagrams (b) for plane strain elasto-plasticity.

exceed 0.1% of volumetric strain $\varepsilon_V = \varepsilon_{xx}+\varepsilon_{yy}$ and do not exceed the value of 1% in any of the strain components, but are otherwise arbitrary. Suitable interpolation ensures that the load step sizes are moderate and the strain data represents a clear deformation path. Each data series consists of 750 data points. Ten data series are used for training and five for validation. The training and validation data is visualised in Fig. 26 and also represented in the results in Section 7.3.

In the problem under consideration, it is known that the data represents elasto-plastic material behaviour. In such cases it is generally of particular interest to capture the yield surface accurately. Hence, it is chosen to include two additional quantities in the training, namely the hydrostatic pressure

$$p = \frac{\sigma_{xx}+\sigma_{yy}+\sigma_{zz}}{3} \tag{25}$$

and the norm of the deviatoric stress

$$q = \sqrt{\frac{1}{2}\left((\sigma_{xx}-\sigma_{yy})^2+(\sigma_{yy}-\sigma_{zz})^2+(\sigma_{zz}-\sigma_{xx})^2+6\left(\sigma_{xy}^2+\sigma_{yz}^2+\sigma_{zx}^2\right)\right)}\,. \tag{26}$$

For each data series, p and q are computed and corresponding approximation errors are formulated analogously to Eq. (22). Hence, ten training data series, four stress coefficients together with p and q result in 60 objectives to be minimised by the multi-objective particle swarm optimisation described in Section 4.2. Recalling the supervised optimisation framework described in Section 4.1, it is noted that it was chosen to base the messages passed from the team leaders to the supervisor on a weighted average, *i.e.* the approximation errors of p and q are weighted five times more heavily than each of the stress coefficients.

7.2. Convergence

The system is trained on $N_{\text{CPU}} = 101$ CPUs of a high performance computer cluster and left to run for 72 h. The resulting convergence is shown in Fig. 27(a). In Fig. 27(b) all stress responses are displayed against their data counterparts. For the data used in training the accuracy is high and the correlation coefficient exceeds the value of 99.6%. For the validation data the correlation coefficient exceeds 98.5%, but a small number of data points show large errors.

Therefore, a second training stage, *i.e.* fine tuning, based on Backpropagation Through Time BPTT is performed, using the Python library PyTorch. The Adam optimiser is employed in combination with a small learning rate of 0.0001. The optimisation is initialised with the network configuration obtained from the PSO based, first stage of training. The additional objectives relating to p and q are ignored and the average approximation error of the stress coefficients is used as the single objective. The PyTorch fine tuning is carried out on an Nvidia V100 GPU for 48 h with the intention to perform a slow training to avoid exploding gradient problems. Fig. 28(a) shows the convergence of the fine tuning process. Note that the loss measure is different from the one in Fig. 27(a) due to not including p and q. It is easily observed in Fig. 28(b) that the fine tuning effectively removed all of the poor quality responses to the validation data. For validation as well as for training data the correlation coefficients based on all stress coefficients have improved.

Neural network training, based on evolutionary optimisation followed by fine tuning with a gradient based strategy, has also been employed in, for instance, Ding et al. [39] and Such et al. [42].

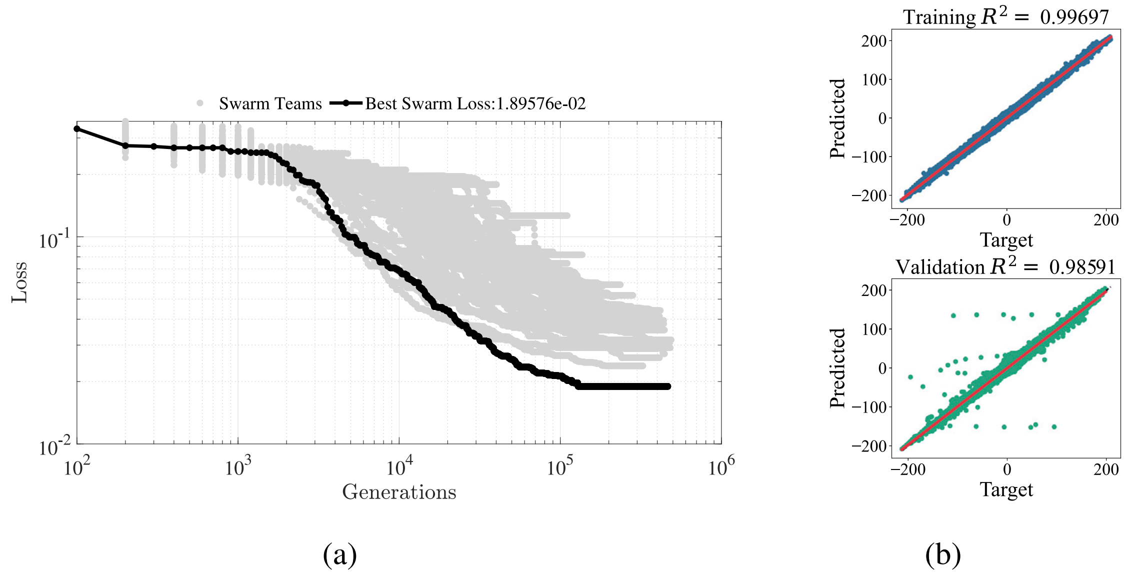


Fig. 27. Training convergence for plane strain elasto-plasticity: Supervised optimisation based on multiple instances of MOPSO (a) and stress responses versus stress data (b).

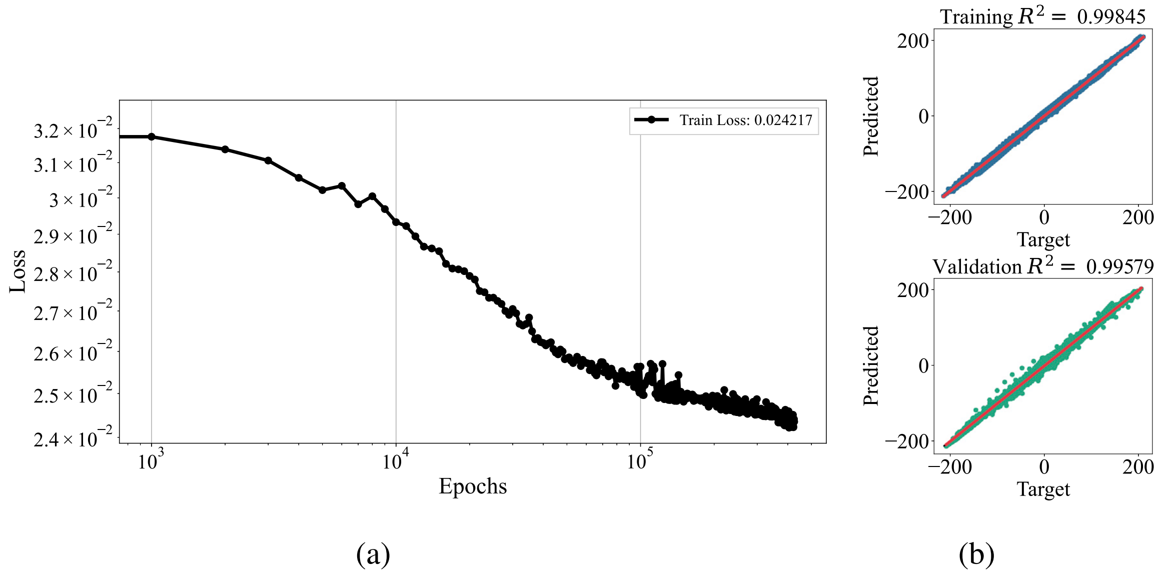


Fig. 28. Training convergence for plane strain elasto-plasticity: Fine tuning with PyTorch (a) and stress responses versus stress data after fine tuning (b).

7.3. Results

Fig. 29 shows the stress response and the stress data for a typical training data series. Figs. 30 and 31 show the stress–strain diagrams for all ten training data series. The associated p-q diagrams are presented in Fig. 32. Figs. 33 to 35 show the corresponding diagrams for the validation data.

The responses obtained from the proposed neural network based stress update procedure are generally observed to agree accurately with the training and validation data. The most notable deviation is noticed in the norm of the deviatoric stresses, q, in Figs. 32 and 35.

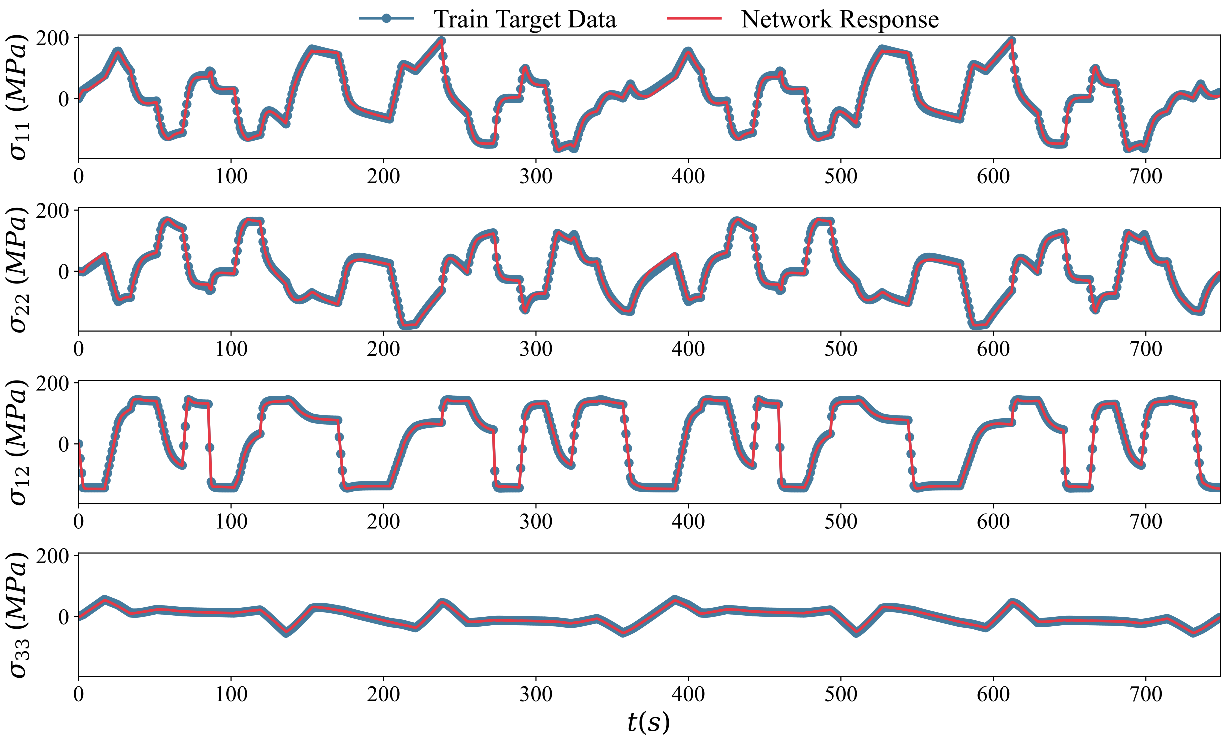


Fig. 29. Stress sequences for training of plane strain elasto-plasticity, Data Series 6.

7.4. Thermodynamic consistency

The dissipation criterion proposed in Eq. (11) is evaluated for two data series and illustrated in Figs. 36 and 37. It can be observed that violation of the criterion is restricted to the load steps immediately following changes of loading direction. The largest negative value does not exceed 6% of the largest value of the dissipation D° computed for any of the dissipative load steps.

7.5. Internal variables

For one of the training data sequences, the evolution of the internal variables $\xi^{(i)}$, $i = 1, 2, 3, 4$ is shown in Fig. 38. The figure also shows the evolution of the coefficients of the plastic strain $\boldsymbol{\varepsilon}^{(p)}$ as obtained from the data generation process based on the continuum mechanical return mapping scheme. The two sets of internal variables have been scaled to the interval $[-1, +1]$ to facilitate their comparison. A strong correlation between $\xi^{(1)}$ and $\varepsilon^{(p)}_{22}$ and between $\xi^{(4)}$ and $\varepsilon^{(p)}_{12}$, respectively, can be observed. The correlation factors are computed and displayed in Fig. 39. They show a mixed pattern of strong and weak correlation. On the one hand the comparison in this specific example therefore confirms that the proposed strategy captures key features of the material state, while on the other hand it shows that it is generally not possible to identify the physical meaning of the internal variables ξ.

7.6. Comparison to alternative network architectures: Network size

In this section the performance of the presented strategy is compared to other generic recurrent neural network architectures. In particular, the following methodologies are considered:

- Basic Recurrent Neural Network (RNN)
- Basic Recurrent Neural Network with Gradient Clipping (RNN+clip)
- Gated Recurrent Units (GRU)
- Long-Short Term Memory (LSTM)

All architectures are tested in a Python environment based on the library PyTorch and trained with back-propagation on the same data presented in Section 7.1. For each strategy, network architectures with a range of sizes are considered, resulting in different numbers of state variables and network parameters.

Fig. 40 shows the approximation errors achieved by the training processes, displayed over the number of state variables or network parameters. Notably the proposed methodology renders the smallest approximation error for the smallest number of internal

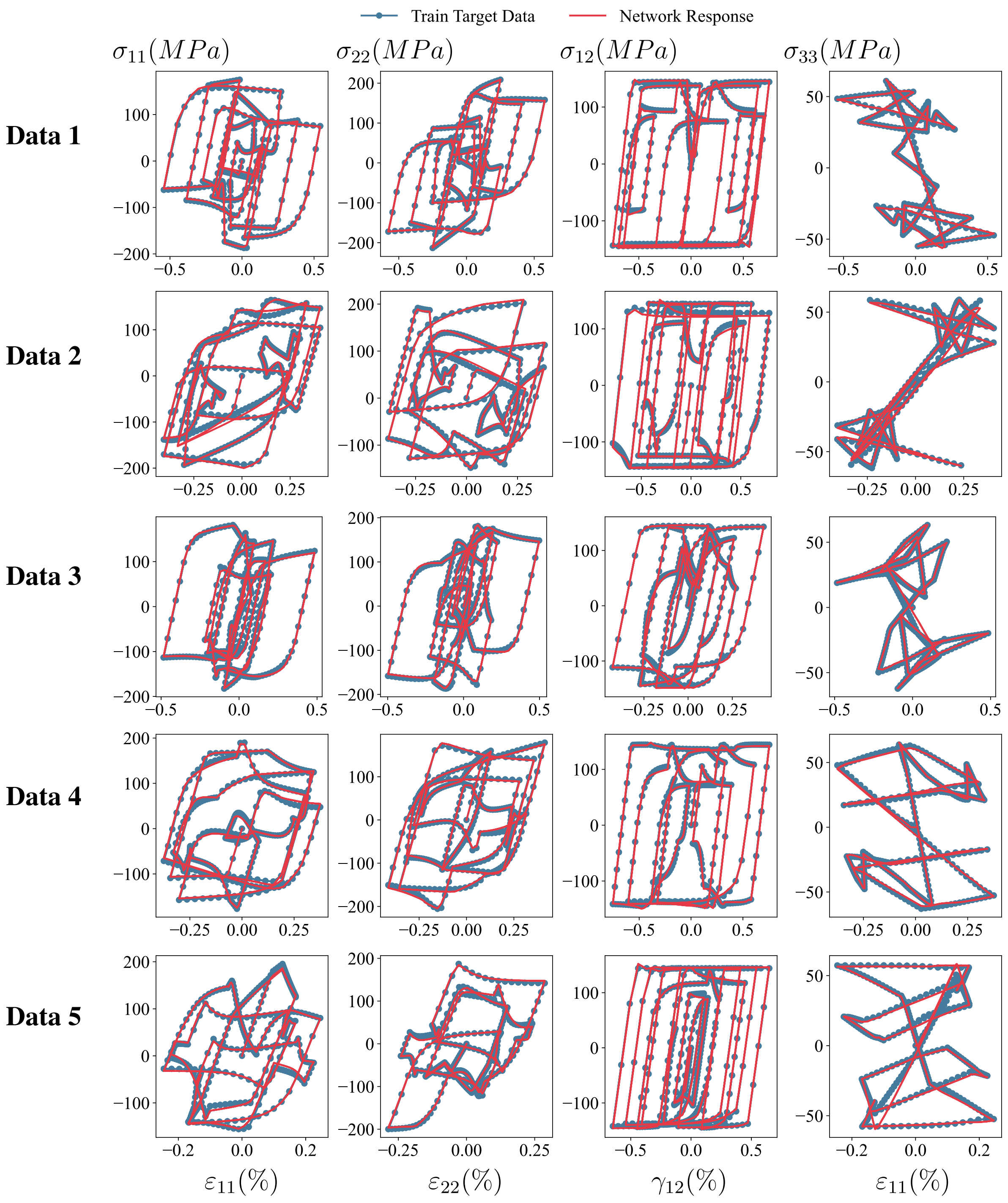


Fig. 30. Stress–strain diagrams for training of plane strain elasto-plasticity, Data Series 1 to 5.

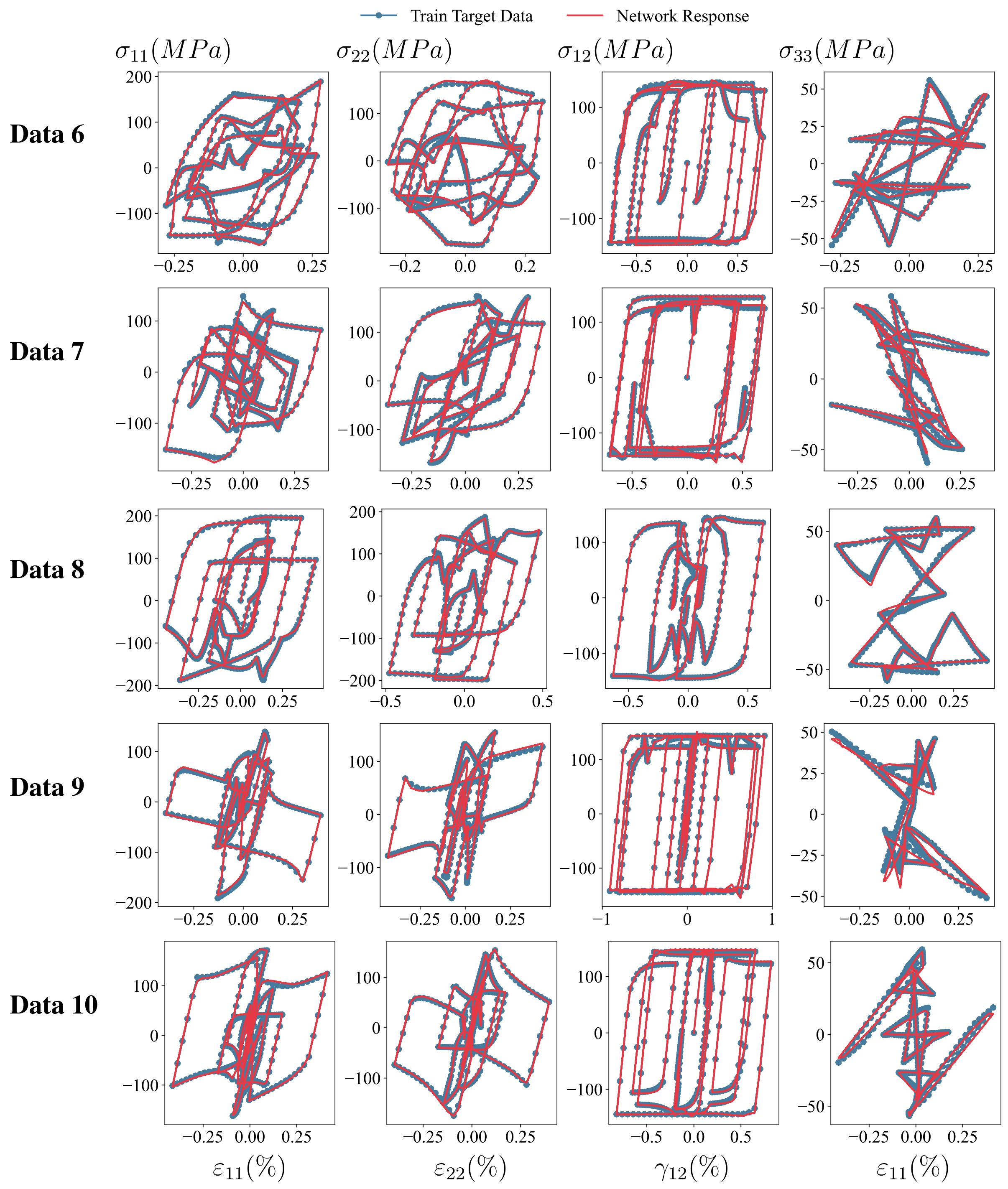


Fig. 31. Stress–strain diagrams for training of plane strain elasto-plasticity, Data Series 6 to 10.

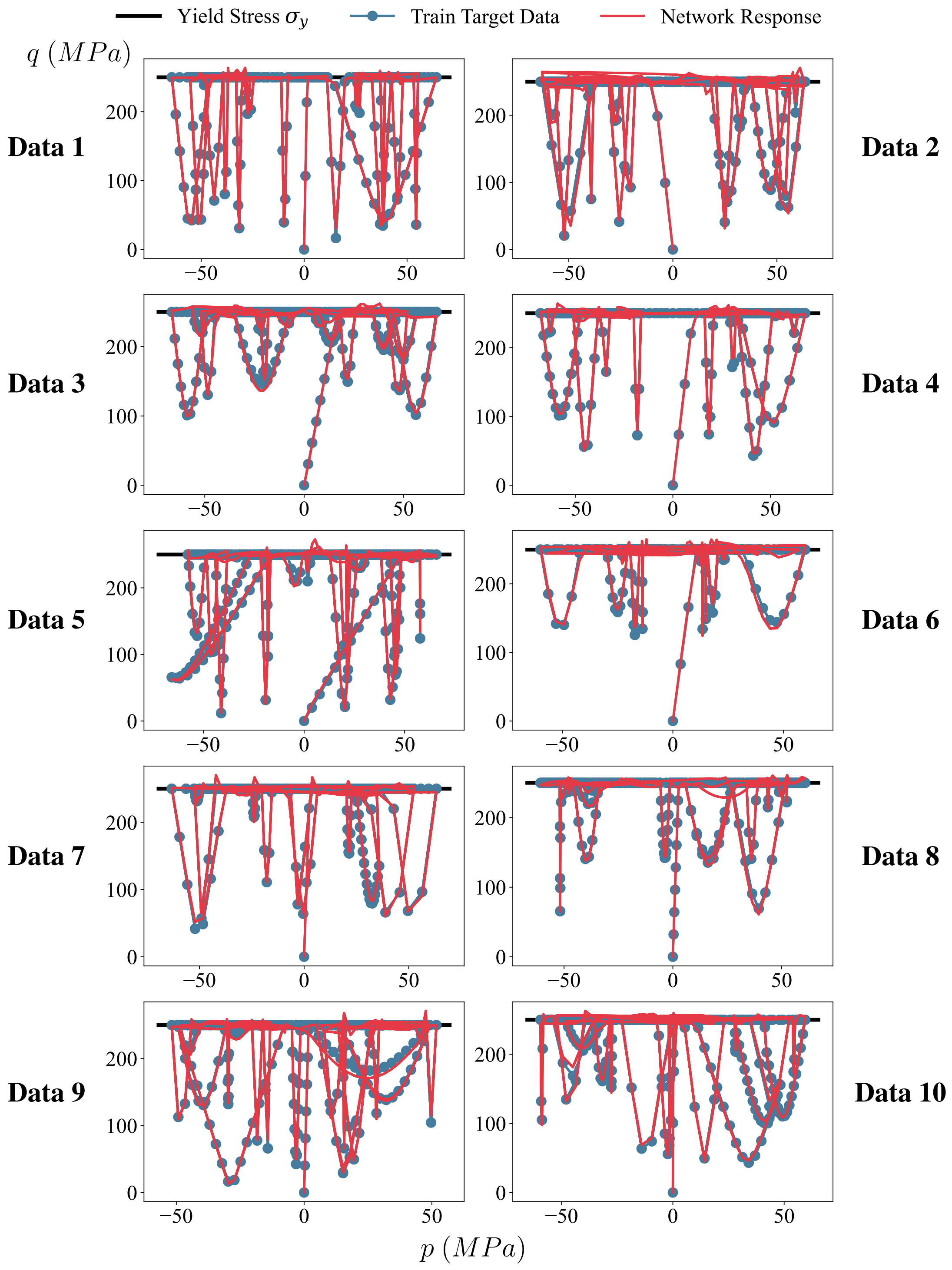


Fig. 32. *p-q* diagrams for training of plane strain elasto-plasticity.

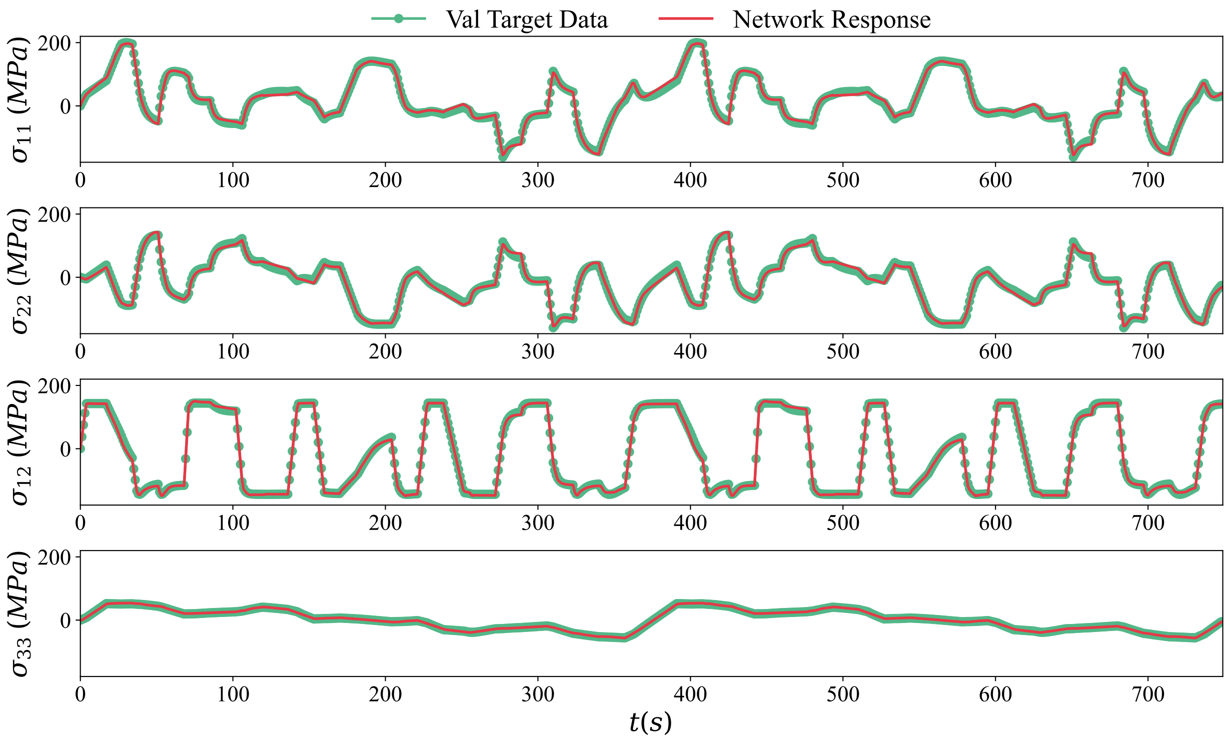


Fig. 33. Stress sequences for validation of plane strain elasto-plasticity, Data Series 2.

state variables. All other methods considered require significantly more network parameters to achieve the same level of accuracy. In Figs. 41 and 42 the correlation coefficient R^2 is displayed over the number of state variables and the number of network parameters for, respectively, the training and the validation data sets. In all diagrams it is observed that the level of accuracy of the present results is not achieved by any of the other methods unless the size of the network is significantly increased.

7.7. Comparison to alternative network architectures: Internal state memory

Models based on network architectures with gated elements such as GRU or LSTM must be trained not to forget the material state, *i.e.* there is a strong dependency of the trained model on the step size in the training data. Due to the absence of gated network elements, the proposed strategy is less prone to forget the internal material state. This is illustrated in the following.

Two trained GRU based alternative network architectures introduced in Section 7.6 are considered, namely the one with 10 and the one with 40 state variables, and denoted by GRU_{10} and GRU_{40}, respectively. Their response to a modified version of training Data Set 5 is computed and compared to the response of the trained proposed model. The data was modified by inserting 100 copies of the same data point at each of two different positions in the data series, *i.e.* the stress and strain data are held constant over 100 data points at two positions of the data series. Note that the modified data sequence had not been used in the training process. This test therefore is a special case of validation.

The data and the different model responses are shown in Figs. 43 and 44. Clearly, the proposed model renders the most accurate response. Note that by enforcing the third of the physical constraints described in Section 2.2, it is possible to set up the proposed model such that it *exactly* remembers the material state.

8. Conclusions

A neural network based stress update procedure for rate-independent inelastic solid materials is presented. The strategy employs internal state variables and is trained on sequences of stress and strain data, which represent physical or numerical experiments. The architecture of the stress update model is designed such that piecewise linear inelastic material behaviour can be represented *exactly*. Therefore, the model recovers the analytical expressions describing the standard stress integration scheme for uniaxial elasto-plasticity.

A training methodology based on gradient-free optimisation of the network weights and biases is described. In a number of numerical examples, the strategy is trained on data representing uniaxial elasto-plasticity, uniaxial elasto-plastic damage and plane

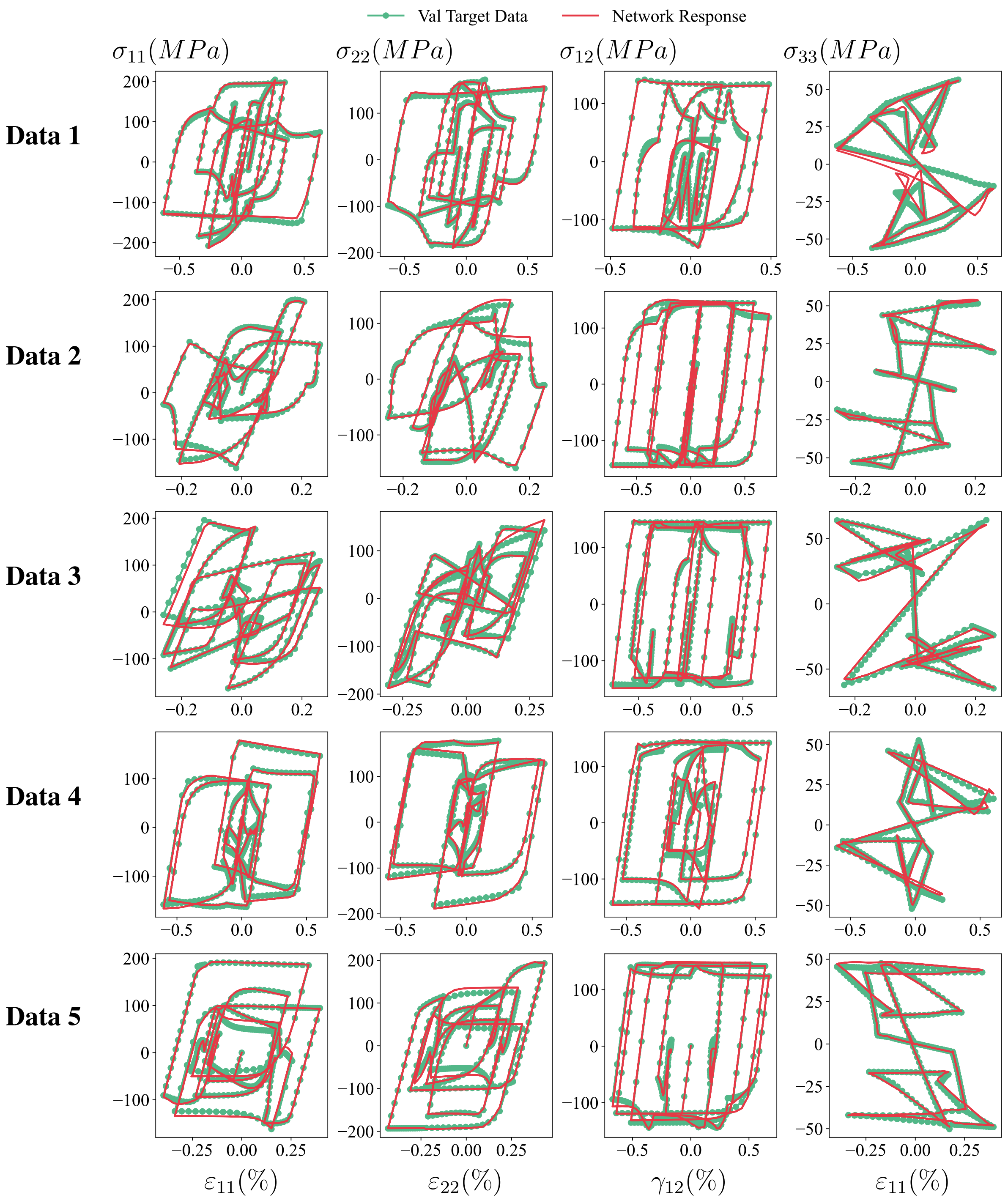


Fig. 34. Stress–strain diagrams for validation of plane strain elasto-plasticity.

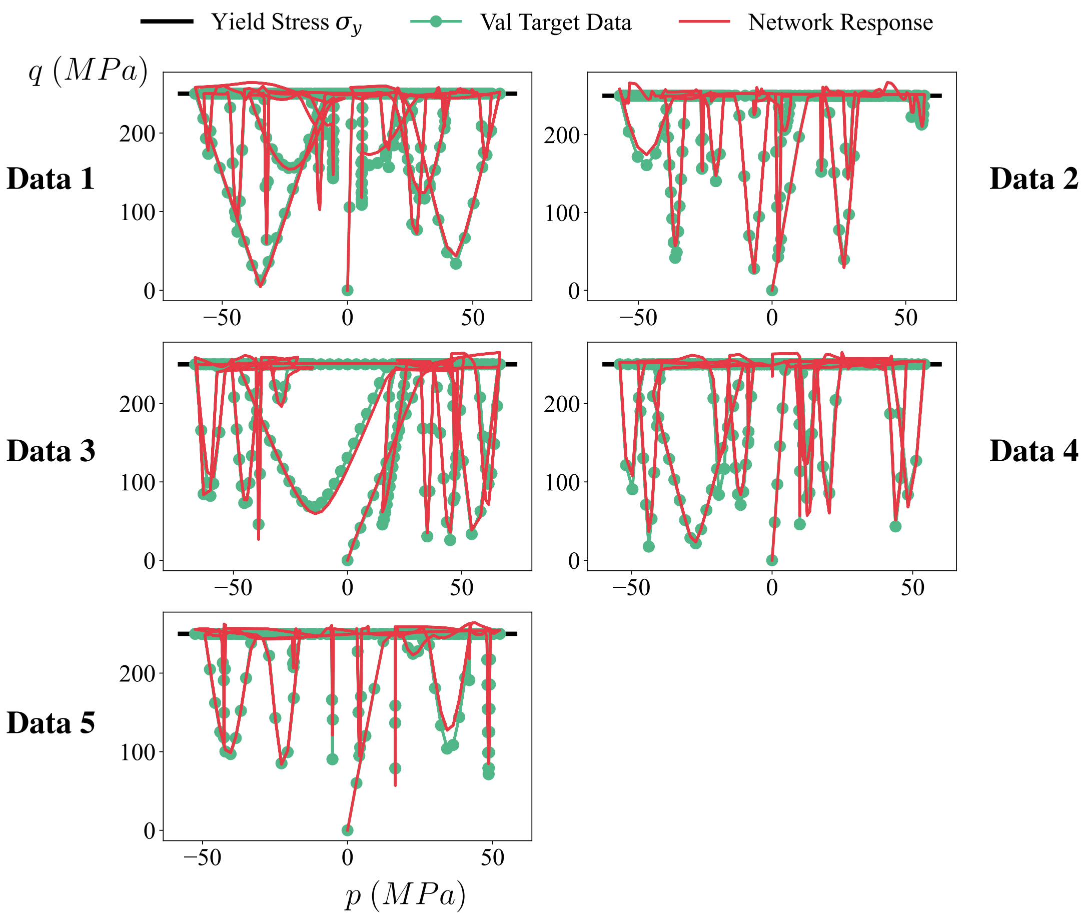


Fig. 35. *p*-*q* diagrams for validation of plane strain elasto-plasticity.

strain elasto-plasticity. A computationally efficient criterion to test thermodynamic consistency is presented and applied to the trained stress update models. The strategy is shown to remember the internal material state more accurately and to require less complex and smaller network architectures than corresponding GRU or LSTM based methodologies. This allows for smaller sets of training data and is beneficial for its integration within the finite element solution procedure, which will be discussed in forthcoming publications.

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.

Data availability

Data will be made available on request.

Acknowledgements

Eugenio J. Muttio gratefully acknowledges research support provided by UKAEA and EPSRC through the Doctoral Training Partnership (DTP) scheme. This work has been part-funded by the EPSRC Energy Programme [grant number EP/W006839/1].

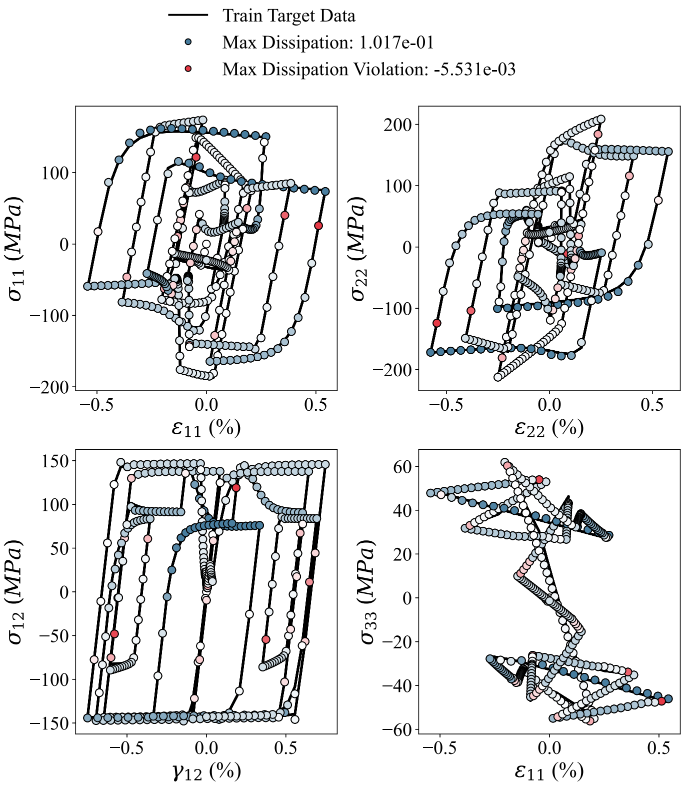


Fig. 36. Stress–strain diagrams with dissipation criterion for plane strain elasto-plasticity, Data Series 1.

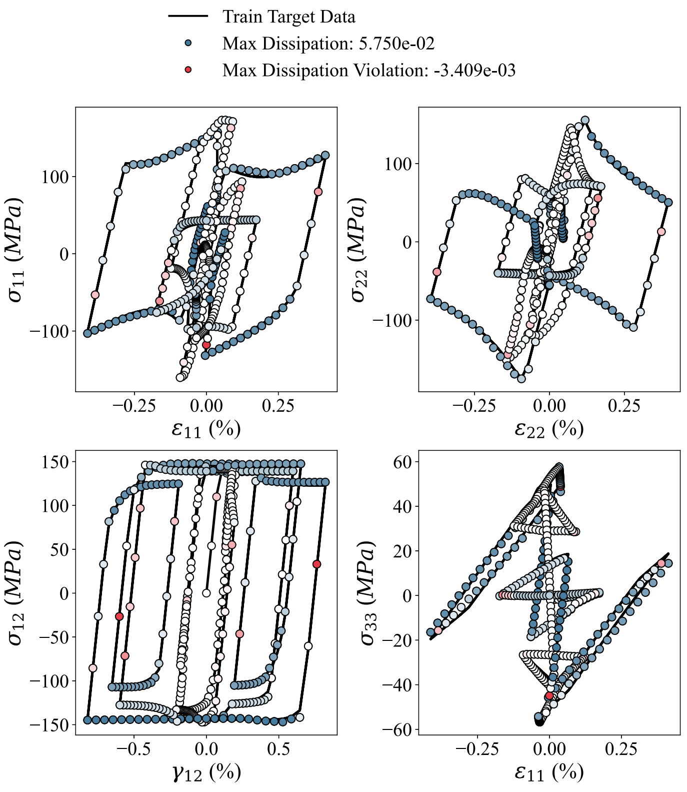


Fig. 37. Stress–strain diagrams with dissipation criterion for plane strain elasto-plasticity, Data Series 10.

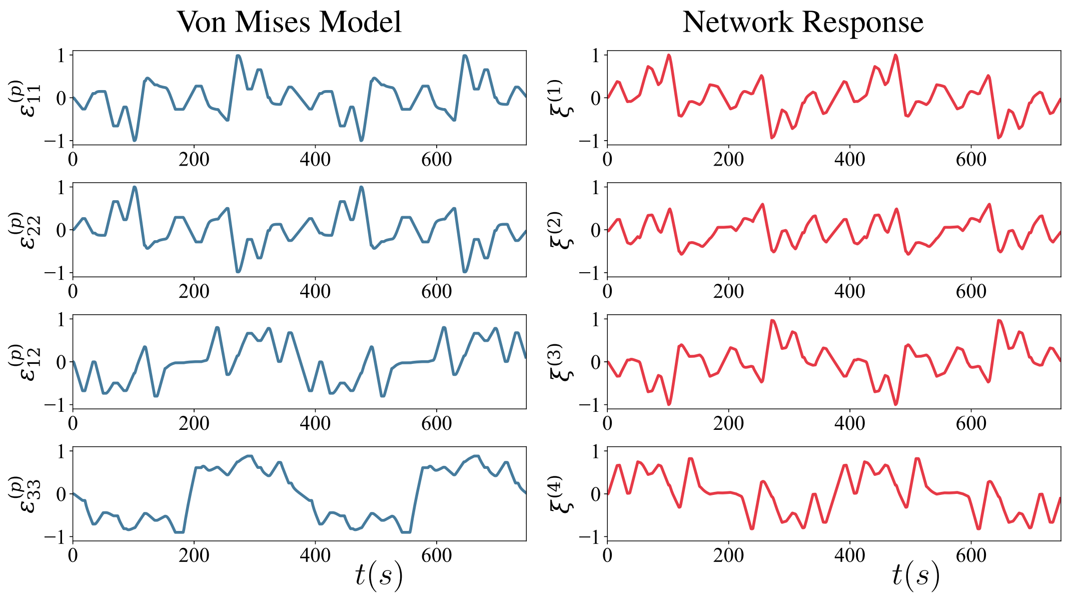
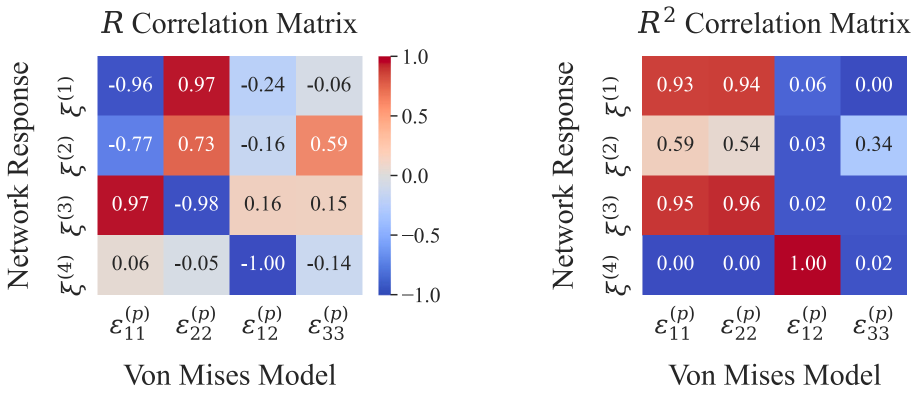


Fig. 38. Comparison of the internal variables for plane strain elasto-plasticity, Data Series 1.



	$\varepsilon_{11}^{(p)}$	$\varepsilon_{22}^{(p)}$	$\varepsilon_{12}^{(p)}$	$\varepsilon_{33}^{(p)}$
$\xi^{(1)}$	-0.96	0.97	-0.24	-0.06
$\xi^{(2)}$	-0.77	0.73	-0.16	0.59
$\xi^{(3)}$	0.97	-0.98	0.16	0.15
$\xi^{(4)}$	0.06	-0.05	-1.00	-0.14

	$\varepsilon_{11}^{(p)}$	$\varepsilon_{22}^{(p)}$	$\varepsilon_{12}^{(p)}$	$\varepsilon_{33}^{(p)}$
$\xi^{(1)}$	0.93	0.94	0.06	0.00
$\xi^{(2)}$	0.59	0.54	0.03	0.34
$\xi^{(3)}$	0.95	0.96	0.02	0.02
$\xi^{(4)}$	0.00	0.00	1.00	0.02

Fig. 39. Correlation analysis of the internal variables for plane strain elasto-plasticity.

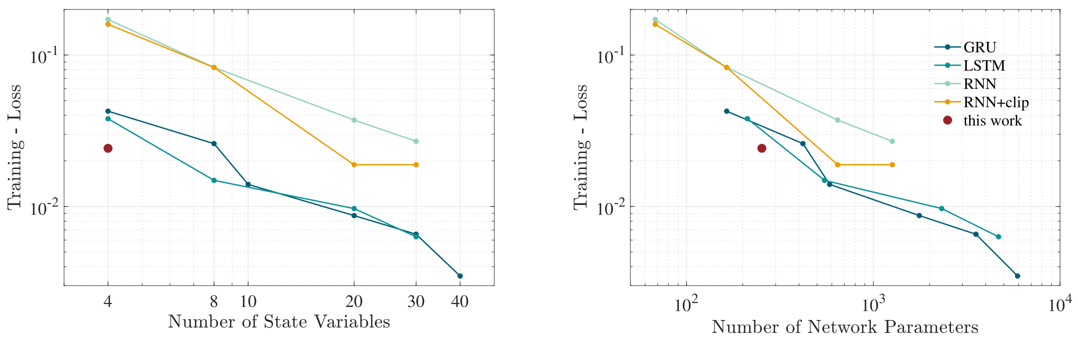


Fig. 40. Loss obtained with different network architectures for plane strain elasto-plasticity.

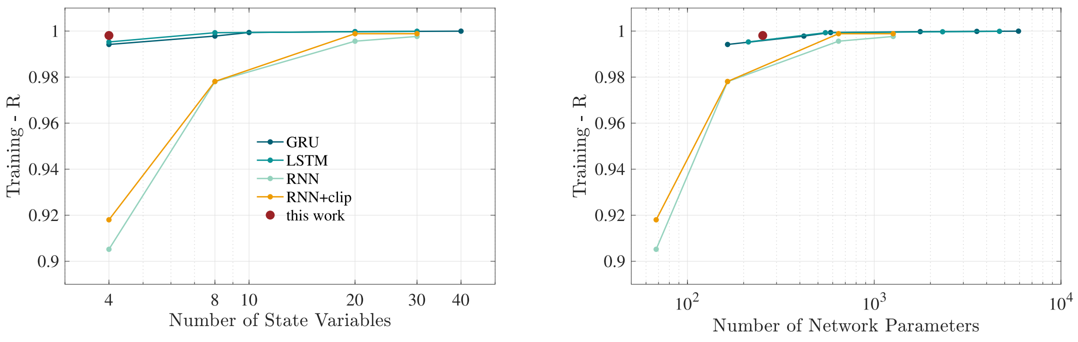


Fig. 41. Training accuracy obtained with different network architectures for plane strain elasto-plasticity. The correlation coefficient R^2 is similar to the coefficient shown in Figs. 27(b) and 28(b).

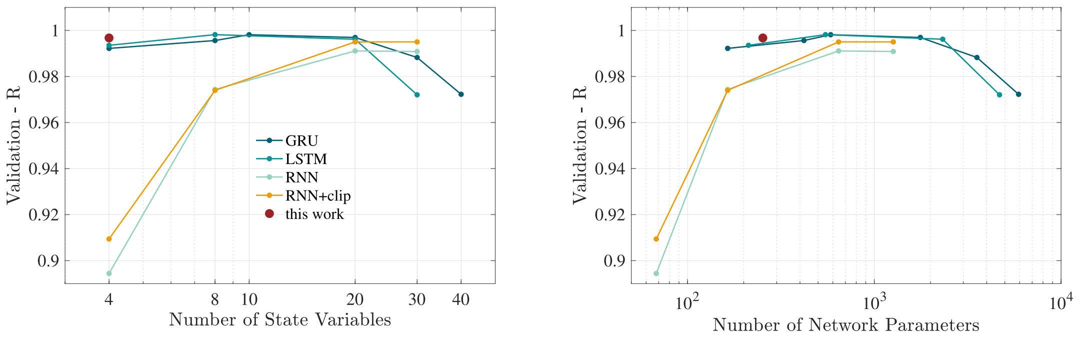


Fig. 42. Validation accuracy obtained with different network architectures for plane strain elasto-plasticity. The correlation coefficient R^2 is similar to the coefficient shown in Figs. 27(b) and 28(b).

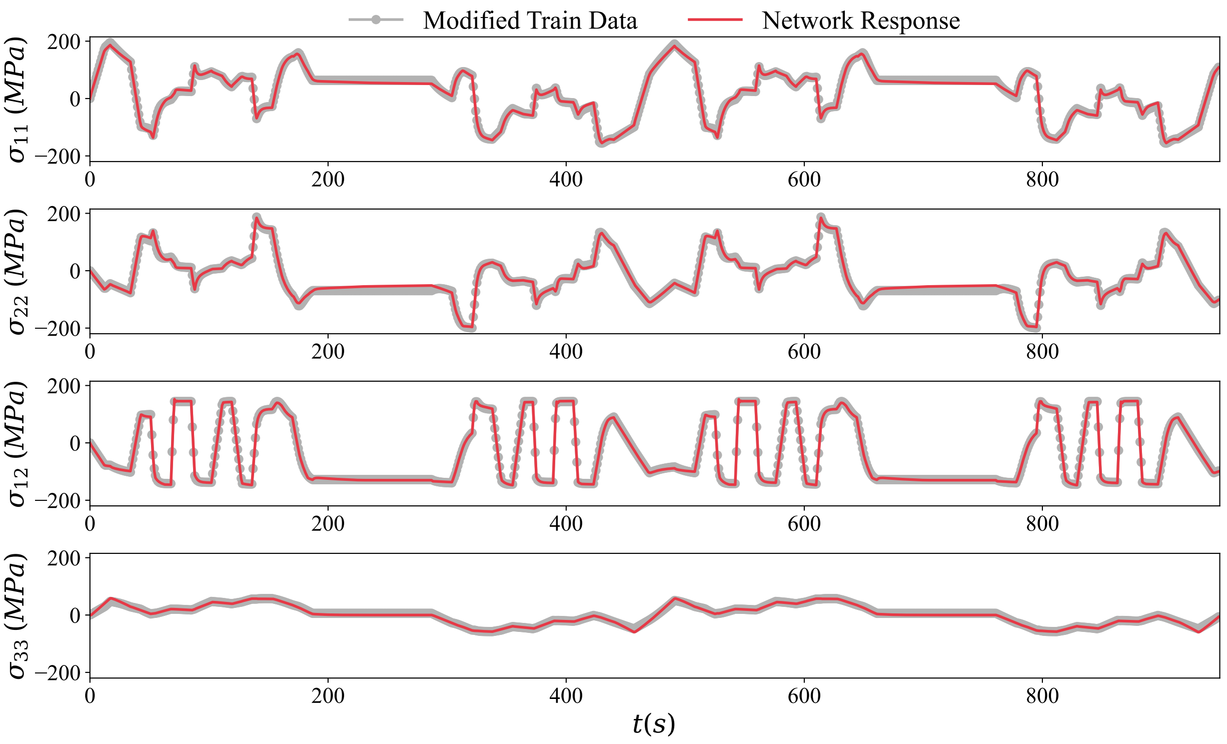


Fig. 43. Internal state memory test for plane strain elasto-plasticity, performance of the proposed model.

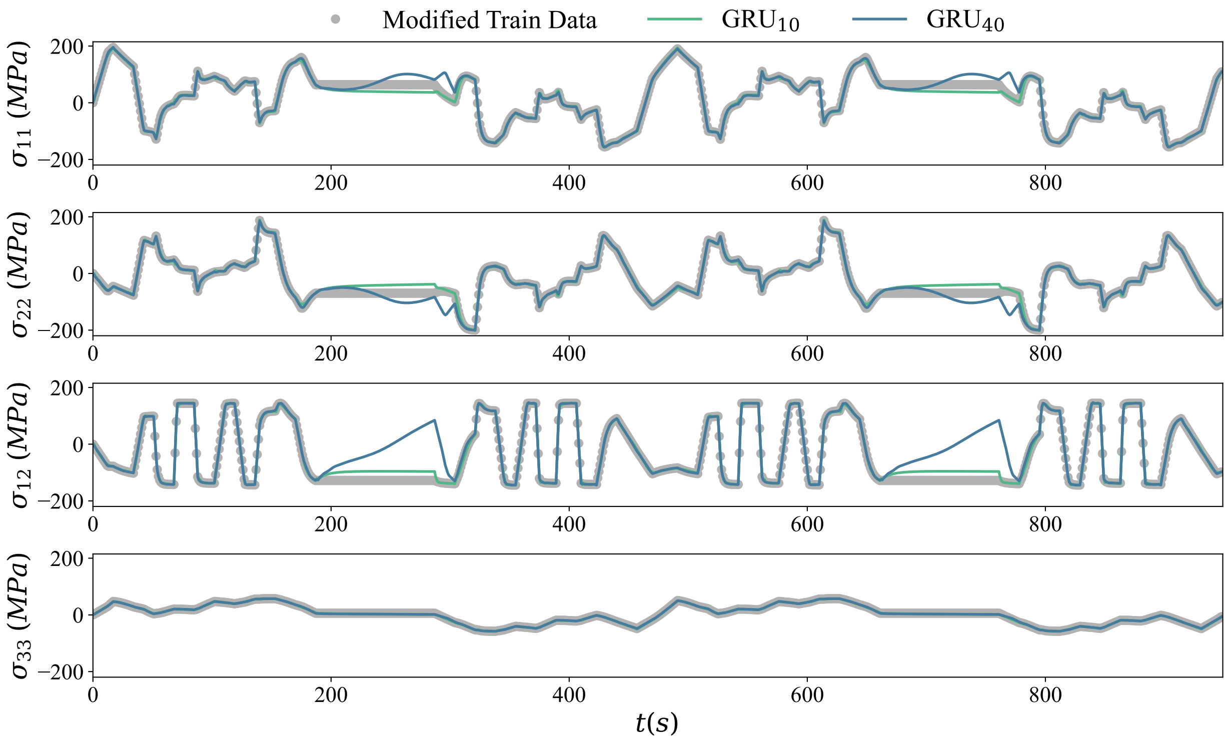


Fig. 44. Internal state memory test for plane strain elasto-plasticity, performance of GRU_{10} and GRU_{40}.

References

[1] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.

[2] R. Eggersmann, T. Kirchdoerfer, S. Reese, L. Stainier, M. Ortiz, Model-free data-driven inelasticity, Comput. Methods Appl. Mech. Engrg. 350 (2019) 81–99.

[3] Y. Shen, K. Chandrashekhara, W. Breig, L. Oliver, Finite element analysis of V-ribbed belts using neural network based hyperelastic material model, Int. J. Non-Linear Mech. 40 (6) (2005) 875–890.

[4] B. Le, J. Yvonnet, Q.-C. He, Computational homogenization of nonlinear elastic materials using neural networks, Internat. J. Numer. Methods Engrg. 104 (12) (2015) 1061–1084.

[5] L. Nguyen, M. Keip, A data-driven approach to nonlinear elasticity, Comput. Struct. 194 (2018) 97–115, http://dx.doi.org/10.1016/j.compstruc.2017.07.031.

[6] G. Chen, Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity, Comput. Mech. 67 (2021) 1009–1019.

[7] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. Bessa, Deep learning predicts path-dependent plasticity, Proc. Natl. Acad. Sci. 116 (52) (2019) 26414–26420.

[8] R. Eggersmann, T. Kirchdoerfer, S. Reese, L. Stainier, M. Ortiz, Model-free data-driven inelasticity, Comput. Methods Appl. Mech. Engrg. 350 (2019) 81–99, http://dx.doi.org/10.1016/j.cma.2019.02.016.

[9] D. Huang, J.N. Fuhg, C. Weißenfels, P. Wriggers, A machine learning based plasticity model using proper orthogonal decomposition, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113008.

[10] Y. Heider, K. Wang, W. Sun, SO (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112875.

[11] A. Zhang, D. Mohr, Using neural networks to represent von Mises plasticity with isotropic hardening, Int. J. Plast. 132 (2020) 102732, http://dx.doi.org/10.1016/j.ijplas.2020.102732.

[12] N. Vlassis, W. Sun, Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening, Comput. Methods Appl. Mech. Engrg. 377 (2021) http://dx.doi.org/10.1016/j.cma.2021.113695.

[13] P.-W. Gerbaud, D. Néron, P. Ladevèze, Data-driven elasto-(visco)-plasticity involving hidden state variables, Comput. Mech. 402 (2022) 115394.

[14] T. Furukawa, G. Yagawa, Implicit constitutive modelling for viscoplasticity using neural networks, Internat. J. Numer. Methods Engrg. 43 (2) (1998) 195–219.

[15] D.W. Abueidda, S. Koric, N. Sobh, H. Sehitoglu, Deep learning for plasticity and thermo-viscoplasticity, Int. J. Plast. 136 (2021) 102852, http://dx.doi.org/10.1016/j.ijplas.2020.102852.

[16] T. Furukawa, M. Hoffman, Accurate cyclic plastic analysis using a neural network material model, Eng. Anal. Bound. Elem. 28 (3) (2004) 195–204.

[17] M. Teranishi, Neural network constitutive model for uniaxial cyclic plasticity based on return mapping algorithm, Mech. Res. Commun. 119 (2022) 103815.

[18] G.J. Yun, J. Ghaboussi, A. Elnashai, A new neural network-based model for hysteretic behavior of materials, Internat. J. Numer. Methods Engrg. 73 (4) (2008) 447–469, http://dx.doi.org/10.1002/nme.2082.

[19] M. Fernández, S. Rezaei, J.R. Mianroodi, F. Fritzen, S. Reese, Application of artificial neural networks for the prediction of interface mechanics: a study on grain boundary constitutive behavior, Adv. Model. Simul. Eng. Sci. 7 (1) (2020) 1–27.

[20] X. Liu, F. Tao, H. Du, W. Yu, K. Xu, Learning nonlinear constitutive laws using neural network models based on indirectly measurable data, J. Appl. Mech. 87 (8) (2020) 081003.

[21] K. Wang, W. Sun, Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning, Comput. Methods Appl. Mech. Engrg. 346 (2019) 216–241.

[22] G. Kaklauskas, J. Ghaboussi, X. Wu, Neural network modelling of stress-strain relationships for tensile concrete in flexure, Statyba 5 (5) (1999) 295–301, http://dx.doi.org/10.1080/13921525.1999.10531479.
[23] S. Jung, J. Ghaboussi, Neural network constitutive model for rate-dependent materials, Comput. Struct. 84 (2006) 955–963, http://dx.doi.org/10.1016/j.compstruc.2006.02.015.
[24] D. Perić, E.A. de Souza Neto, R.A. Feijóo, M. Partovi, A.J. Carneiro Molina, On micro-to-macro transitions for multi-scale analysis of non-linear heterogeneous materials: unified variational basis and finite element implementation, Internat. J. Numer. Methods Engrg. 87 (2011) 149–170.
[25] L. Wu, V.D. Nguyen, N.G. Kilingar, L. Noels, A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113234, http://dx.doi.org/10.1016/j.cma.2020.113234.
[26] K. Wang, W. Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380, http://dx.doi.org/10.1016/j.cma.2018.01.036.
[27] F. Ghavamian, A. Simone, Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network, Comput. Methods Appl. Mech. Engrg. 357 (2019) 112594, http://dx.doi.org/10.1016/j.cma.2019.112594.
[28] K. Karapiperis, L. Stainier, M. Ortiz, J.E. Andrade, Data-Driven multiscale modeling in mechanics, J. Mech. Phys. Solids 147 (2021) 104239, http://dx.doi.org/10.1016/j.jmps.2020.104239.
[29] J.C. Simo, T.J.R. Hughes, Computational Inelasticity, Springer, New York, USA, 1998.
[30] E.A. de Souza Neto, D. Peric, D.R.J. Owen, Computational Methods for Plasticity: Theory and Applications, Wiley, Chichester, 2008.
[31] M.B. Gorji, M. Mozaffar, J.N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, J. Mech. Phys. Solids 143 (2020) 103972.
[32] C. Bonatti, D. Mohr, One for all: Universal material model based on minimal state-space neural networks, Sci. Adv. (2021) 7, http://dx.doi.org/10.1126/sciadv.abf3658.
[33] C. Bonatti, D. Mohr, On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids, J. Mech. Phys. Solids 158 (2022) 104697.
[34] E.J. Muttio, W.G. Dettmer, J. Clarke, D. Perić, Z. Ren, L. Fletcher, A supervised parallel optimisation framework for metaheuristic algorithms, Swarm Evol. Comput. (2023) 101445, http://dx.doi.org/10.1016/j.swevo.2023.101445.
[35] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.
[36] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.
[37] A. Blanco, M. Delgado, M.C. Pegalajar, A real-coded genetic algorithm for training recurrent neural networks, Neural Netw. 14 (2001) 93–105.
[38] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, Trans. Syst. Man Cybern. B 34 (2004) 997–1006.
[39] S. Ding, C. Su, J. Yu, An optimizing BP neural network algorithm based on genetic algorithm, Artif. Intell. Rev. 36 (2011) 153–162.
[40] S.C. Duong, E. Uezato, H. Kinjo, T. Yamamoto, A hybrid evolutionary algorithm for recurrent neural network control of a three-dimensional tower crane, Autom. Constr. 23 (2012) 55–63.
[41] A. Benvidi, S. Abbasi, A. Gharaghani, M. Dehghan Tezerjani, S. Masoum, Spectrophotometric determination of synthetic colorants using PSO–GA-ANN, Food Chem. 220 (2017) 377–384.
[42] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, J. Clune, Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2017, http://dx.doi.org/10.48550/ARXIV.1712.06567, arXiv https://arxiv.org/abs/1712.06567.
[43] J. Lemaitre, J.-L. Chaboche, Mechanics of Solid Materials, Cambridge University Press, Cambridge, UK, 1990.
[44] J. Robinson, S. Sinton, Y. Rahmat-Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna, in: IEEE Antennas and Propagation Society International Symposium (IEEE Cat. No.02CH37313), Vol. 1, 2002, pp. 314–317, http://dx.doi.org/10.1109/APS.2002.1016311, vol.1.
[45] J.F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka, A.D. George, Parallel global optimization with the particle swarm algorithm, Internat. J. Numer. Methods Engrg. 61 (13) (2004) 2296–2315, http://dx.doi.org/10.1002/nme.1149, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1149.
[46] J.-F. Chang, S.-C. Chu, J. Roddick, J.-S. Pan, A parallel particle swarm optimization algorithm with communication strategies, J. Inf. Sci. Eng. 21 (2005) 809–818.
[47] M. Waintraub, R. Schirru, C.M.N.A. Pereira, Multiprocessor modeling of parallel particle swarm optimization applied to nuclear engineering problems, Prog. Nucl. Energy 51 (6) (2009) 680–688, http://dx.doi.org/10.1016/j.pnucene.2009.02.004, URL: https://www.sciencedirect.com/science/article/pii/S014919700900033X.
[48] A.A. Nik, F.M. Nejad, H. Zakeri, Hybrid PSO and GA approach for optimizing surveyed asphalt pavement inspection units in massive network, Autom. Constr. 71 (2016) 325–345, http://dx.doi.org/10.1016/j.autcon.2016.08.004, URL: https://www.sciencedirect.com/science/article/pii/S0926580516301571.
[49] K. Wansasueb, S. Bureerat, S. Kumar, Ensemble of four metaheuristic using a weighted sum technique for aircraft wing design, Eng. Appl. Sci. Res. 48 (4) (2021) 385–396, URL: https://ph01.tci-thaijo.org/index.php/easr/article/view/242706. Number: 4.
[50] P. Singh, R. Kottath, An ensemble approach to meta-heuristic algorithms: Comparative analysis and its applications, Comput. Ind. Eng. 162 (2021) 107739, http://dx.doi.org/10.1016/j.cie.2021.107739, URL: https://www.sciencedirect.com/science/article/pii/S0360835221006434.
[51] M. Reyes-Sierra, C.A. Coello Coello, Multi-objective particle swarm optimisers: A survey of the state-of-the-art, Int. J. Comput. Intell. Res. 2 (2006) 287–308.
[52] P. Venini, P. Morana, An adaptive wavelet-Galerkin method for an elastic-plastic-damage constitutive model: 1D problem, Comput. Methods Appl. Mech. Engrg. 190 (42) (2001) 5619–5638.