Computational Mechanics
https://doi.org/10.1007/s00466-024-02477-1

ORIGINAL PAPER

Viscoelasticty with physics-augmented neural networks: model formulation and training methods without prescribed internal variables

Max Rosenkranz[1,2] · Karl A. Kalina[2] · Jörg Brummund[2] · WaiChing Sun[3] · Markus Kästner[2]

Received: 24 January 2024 / Accepted: 20 March 2024

Abstract
We present an approach for the data-driven modeling of nonlinear viscoelastic materials at small strains which is based on physics-augmented neural networks (NNs) and requires only stress and strain paths for training. The model is built on the concept of generalized standard materials and is therefore thermodynamically consistent by construction. It consists of a free energy and a dissipation potential, which can be either expressed by the components of their tensor arguments or by a suitable set of invariants. The two potentials are described by fully/partially input convex neural networks. For training of the NN model by paths of stress and strain, an efficient and flexible training method based on a long short-term memory cell is developed to automatically generate the internal variable(s) during the training process. The proposed method is benchmarked and thoroughly compared with existing approaches. Different databases with either ideal or noisy stress data are generated for training by using a conventional nonlinear viscoelastic reference model. The coordinate-based and the invariant-based formulation are compared and the advantages of the latter are demonstrated. Afterwards, the invariant-based model is calibrated by applying the three training methods using ideal or noisy stress data. All methods yield good results, but differ in computation time and usability for large data sets. The presented training method based on a recurrent cell turns out to be particularly robust and widely applicable. We show that the presented model together with the recurrent cell for training yield complete and accurate 3D constitutive models even for sparse bi- or uniaxial training data.

Keywords Artificial neural networks · Viscoelasticity · Thermodynamic consistency · Internal variables · Recurrent neural networks · Input convex neural networks

1 Introduction

Despite a large number of existing classical constitutive models for nonlinear elastic and inelastic materials [1, 2], the description of novel materials with complex constitutive behavior remains a challenging task. The choice of a suitable model and the identification of the corresponding material parameters is time-consuming and does not necessarily lead to results with the desired accuracy, so that the development of new specialized models may be necessary. For this reason, a new class of approaches has emerged in recent years, which are often referred to as *data-driven* or *data-based* methods [3, 4].

1.1 Constitutive modeling with neural networks

Historically, the pioneering work of Ghaboussi et al. [5] from the beginning of the 1990s is particularly noteworthy. In this work, NNs, specifically *feedforward neural networks (FNNs)*, are used for the first time to predict hysteresis in uniaxial and multiaxial stress states. Therein, the FNN is fed with information from several previous time steps to enable it to learn history-dependent behavior. In Furukawa and Yagawa [6], a constitutive model based on an FNN is presented that can be used to train uniaxial viscoplastic behavior. However, this requires internal variables in the training process, whose experimental identification is also described. Despite

✉ Markus Kästner
markus.kaestner@tu-dresden.de

1 Institute of Artificial Intelligence, TU Dresden, 01062 Dresden, Germany

2 Institute of Solid Mechanics, TU Dresden, 01062 Dresden, Germany

3 Department of Civil Engineering and Engineering Mechanics, Columbia University, New York, NY 10027, USA

Published online: 06 May 2024

the rather simple nature of these models from today's point of view, approaches of this kind indicate the potential of data-driven constitutive modeling. Without having to decide on a specific model, it is possible to learn complex material behavior. With the recent rise in popularity of NNs and the associated rapid progress in efficiency and accessibility, many different methods have emerged in an extremely short time, extending and improving these approaches. For example, the works [7–11] propose advanced techniques using FNNs, while [10–14] make use of *recurrent neural networks (RNNs)*, which are capable of making predictions based on past events due to their internal structure [15].

Thus, recurrent architectures are an appealing way to model inelastic behavior, since the provision of history variables or internal variables can be avoided [17]. In particular, the development of advanced RNN cells such as *long short-term memory (LSTM)* [18] or *gated recurrent units (GRUs)* [19], which provide increased memory capacity and enable an efficient training, lead to great popularity and rapid progress in this field.

What remains a challenge for NN-based approaches is the lack of a fundamental inclusion of physics, especially the *second law of thermodynamics*. The networks map input quantities directly to the variable of interest, for example the stress, and thus approximate only the phenomenological relationship between input and output [5, 6]. Compared to most classical constitutive models, the physical motivation is completely missing. This has some significant downsides. Such models, often denoted as *black box models*, cannot guarantee robustness of the predictions beyond the training range covered by the used training data. Exceeding this range may not only lead to wrong but also to physically implausible predictions [20, 21]. Furthermore, the training process is entirely driven by the training data and is not supported by existing knowledge from classical constitutive modeling. This may lead to more complex optimization problems that exhibit gradient conflicts and require more training data [21]. For this reason, it seems natural to integrate the existing knowledge from continuum mechanics and constitutive modeling into the NN architecture to combine the advantages of both concepts. This kind of NN-based constitutive models or scientific machine learning approaches in general are labeled as *physics-informed* [22, 23], *mechanics-informed* [24], *physics-augmented* [25], *physics-constrained* [26], *thermodynamics-informed* [27] or *thermodynamics-based* [21]. These approaches have been intensively pursued for a few years with great success in constitutive modeling for elastic [20, 23, 25, 26, 28–30] elastoplastic [21, 27, 31–36] or viscoelastic behavior [36–40]. Thereby, a distinction must be made between methods with *weak fulfillment* of the principles by an additional term in the loss function and *strong fulfillment* with a priori compliance with the respective principle by constraining the architecture of the network [41]. According to the comparative study presented in Rosenkranz et al. [17], the second approach is more promising since it is more efficient in terms of required data, robust and can extrapolate very well due to the high degree of incorporated physics, but involves some difficulties. The challenge here is to efficiently restrict the network without loosing too much flexibility.

For elastic and especially hyperelastic materials, many works have been published on that topic, e.g., [20, 24, 26, 28, 29, 42–44], among others, in which different requirements for a constitutive model are incorporated in a strong sense. Thereby, an elastic potential is approximated by using an FNN with the deformation or strain invariants as input. To allow the calibration of the network directly by tuples of stress and strain, the derivative of the potential with respect to the deformation, i.e., the network's input, is included into the loss, which is also called Sobolev training [20, 45]. With easily accessible implementations for *automatic differentiation* [46] in popular libraries like TensorFlow, PyTorch or JAX, this is no longer a major difficulty and is used in a wide variety of research areas [22, 47]. Furthermore, *polyconvex* NN models [20, 29, 48] have been formulated by using *fully input convex neural networks (FICNNs)* as introduced by Amos et al. [49]. Also parametrized polyconvex models [50] have been formulated with *partially input convex neural networks (PICNNs)* [49].

Regarding NN-based constitutive modeling of inelastic behavior with a strong physical background, a variety of works have been published meanwhile. Thereby, approaches applying the concept of *internal variables* have shown to be particularly well suited. Remarkable works on the NN-based modeling of plasticity in recent years are, for example [21, 27, 31–33, 51–53], among others. In many of these works, however, the *thermodynamic consistency* is only fulfilled in a weak sense by adding a penalty term to the loss or internal variables must be known prior to training. When using the models in multiscale problems, internal variables can be determined, e.g., by autoencoders [54, 55], but in real experimental setups the determination might not be practical. Thus, there are still some challenges to overcome, in particular with regard to the fulfillment of physical conditions by construction or the provision of internal variables during training. Only the elasto-plastic NN models [32, 33] a priori fulfill the second law of thermodynamics and do not require internal variables in the training data set at the same time. A pioneering NN-based approach to model *viscoelastic* behavior is presented in [56]. Thereby, thermodynamic consistency is ensured by using a convex dissipation potential but internal variables are required for training. Several approaches based on a similar modeling strategy can be found [17, 39, 44, 57]. In contrast to [56], however, no prior knowledge of the internal variable(s) is needed for training there.

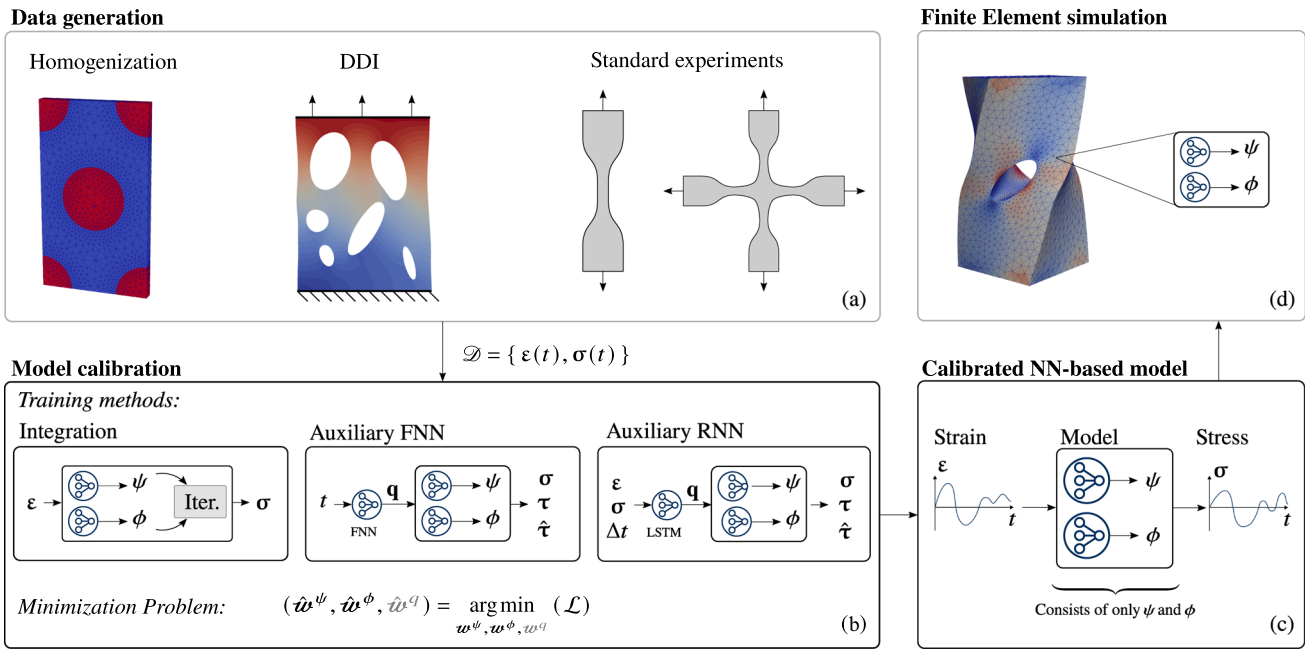


Fig. 1 Overview and classification of the work: **a** A data base containing time sequences of stresses $\sigma(t)$ and strains $\varepsilon(t)$ is required for the calibration of the constitutive model. This data can be obtained from computational homogenization simulations, data-driven identification (DDI) [16], or common experiments. In this work, the data is generated in a simplified way using a given constitutive model. **b** The data set is used for calibrating the NN-based model using two established methods and a novel RNN-based method. **c** The trained model can be tested with unseen sequences and **d** applied in Finite Element (FE) simulations. The FE simulations are not addressed herein. Thus, the focus of this work is on (**b**) and (**c**)

1.2 Objectives and contributions of this work

As outlined above, NN-based constitutive models for viscoelastic problems using internal variables and accounting for fundamental physics as the second law of thermodynamics have so far received comparatively little attention. In addition, the techniques for providing internal variables during training are not satisfactory in every case, as either a high computational effort is required or flexibility and accuracy are not sufficient. Thus, within this contribution, we present a physics-augmented NN model for viscoelastic materials and an efficient training method which only requires stress and strain paths for training. The model is built on the concept of generalized standard materials and is therefore thermodynamically consistent by construction. It consists of a free energy and a dissipation potential, which can be either expressed by the coordinates of their tensor arguments or by a suitable set of invariants. The two potentials are expressed by FICNNs/PICNNs [49]. For training of the NN model by paths of stress and strain, an efficient and flexible training method based on an LSTM cell is developed to automatically provide the internal variable(s) during the training process. The focus of this work is on a comprehensive benchmark test of the proposed method based on existing approaches. These include a method that obtains the internal variable by integrating the evolution equation over the entire sequence [37], while the other method uses an auxiliary FNN for the internal variable(s) [24]. Databases for training are generated by using a conventional nonlinear viscoelastic reference model. These training data sets include either multiaxial, biaxial, uniaxial stress states with either ideal or noisy data points. The coordinate-based and the invariant-based formulation are compared and the three training methods are applied. We show that the presented framework yields complete and accurate 3D constitutive models with all of these datasets, using only a single short training sequence. An overview about this work and a classification in a larger context is schematically illustrated in Fig. 1.

The article is organized as follows: After a short summary of the concept of generalized standard materials in Sect. 2, the NN-based model is described in Sect. 3. In Sect. 4, three methods to calibrate the NN model are explained. Subsequently, the presented model and the training methods are tested and compared within numerical examples in Sect. 5 using different data sets with stress states of decreasing complexity and either ideal or noisy stress data. After a discussion of the results, some closing remarks are given in Sect. 6.

Notation

Throughout this work, several important quantities are tensors of different ranks. The space of tensors of rank

$n \geq 1$ is denoted as $\mathcal{L}_n$. Especially, the cases of $n = 2$ and $n = 4$, i.e., tensors of rank two and four, are of importance herein. A tensor of rank two is denoted with upright bold symbols as $\mathbf{T} = T_{ij}\boldsymbol{e}_i \otimes \boldsymbol{e}_j \in \mathcal{L}_2$ and a tensor of rank four with blackboard bold symbols as $\mathbb{C} = C_{ijkl}\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \in \mathcal{L}_4$. Therein, the Einstein summation convention is used, $\otimes$ is the dyadic product and $\boldsymbol{e}_i \in \mathcal{L}_1$ is the i-th Cartesian basis vector. For tensors of rank two, only symmetric second order tensors $\mathbf{S} \in \mathcal{S}ym_2 \subset \mathcal{L}_2$ are relevant in the scope of this work, with $\mathcal{S}ym_2 = \left\{ \mathbf{S} \in \mathcal{L}_2 \mid \mathbf{S} = \mathbf{S}^\top \right\}$ and $\mathbf{S}^\top = S_{ji}\boldsymbol{e}_i \otimes \boldsymbol{e}_j$ denoting the transpose of $\mathbf{S}$. For tensors of rank four, the subset $\mathcal{S}ym_4 \subset \mathcal{L}_4$ of tensors with major and minor symmetries is defined as $\mathcal{S}ym_4 = \left\{ \mathbb{C} \in \mathcal{L}_4 \mid C_{ijkl} = C_{ijlk} = C_{jikl} = C_{klij} \right\}$. Some special fourth order tensors required here are the fully symmetric fourth order identity tensor $\mathbb{I}^{\mathrm{S}} = \frac{1}{2}\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right) \boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l \in \mathcal{S}ym_4$, the spherical projector $\mathbb{I}^{\mathrm{K}} = \frac{1}{3}\mathbf{1} \otimes \mathbf{1} \in \mathcal{S}ym_4$ with $\mathbf{1} = \delta_{ij}\boldsymbol{e}_i \otimes \boldsymbol{e}_j$ and the deviatoric projector given by $\mathbb{I}^{\mathrm{D}} = \mathbb{I}^{\mathrm{S}} - \mathbb{I}^{\mathrm{K}}$. In the above definitions, $:$ denotes double contraction of adjacent indices. Furthermore, $\mathrm{tr}(\mathbf{T}) = T_{kk}$ is the trace of $\mathbf{T} \in \mathcal{L}_2$, the square of a tensor is $\mathbf{T}^2 = T_{ik}T_{kj}\boldsymbol{e}_i \otimes \boldsymbol{e}_j$ and the Frobenius norm is denoted by $\|\mathbf{T}\|$ in this work and is defined as $\|\mathbf{T}\| = \sqrt{\mathbf{T} : \mathbf{T}^\top}$.

2 Generalized standard materials

The concept of *generalized standard materials (GSMs)* [58–65] allows the formulation of thermodynamically consistent constitutive models that are entirely described by two scalar valued functions. In the context of this work, thermodynamic consistency is equivalent to satisfying the Clausius-Duhem inequality

$$\mathcal{D} = \boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}} - \dot{\psi} \geq 0, \tag{1}$$

where $\mathcal{D}$ is the dissipation rate, $\boldsymbol{\sigma} \in \mathcal{S}ym_2$ is the stress, $\dot{\boldsymbol{\varepsilon}} \in \mathcal{S}ym_2$ is the strain rate, and ψ is the Helmholtz free energy density, or *free energy* for short. This free energy $\psi(\boldsymbol{\varepsilon}, \mathbf{q}_\alpha)$ depends on the current strain $\boldsymbol{\varepsilon} \in \mathcal{S}ym_2$ and a set of internal variables $\mathbf{q}_\alpha \in \mathcal{S}ym_2$. The stress $\boldsymbol{\sigma}$ and the internal forces $\boldsymbol{\tau}_\alpha \in \mathcal{S}ym_2$ are obtained by differentiating the free energy with respect to the strain or the internal variables, respectively, i.e.,

$$\boldsymbol{\sigma} = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}} \quad \text{and} \quad \boldsymbol{\tau}_\alpha = -\frac{\partial \psi}{\partial \mathbf{q}_\alpha}. \tag{2}$$

In the context of GSMs, there exists another potential besides the free energy, the so-called *dissipation potential* $\phi(\dot{\mathbf{q}}_\alpha, \mathbf{q}_\alpha, \boldsymbol{\varepsilon})$, which depends on the rates of the internal variables $\dot{\mathbf{q}}_\alpha$ and possibly on the strain and the internal variables themselves. Differentiating the dissipation potential with respect to the rate of the internal variables again yields the internal forces

$$\hat{\boldsymbol{\tau}}_\alpha = \frac{\partial \phi}{\partial \dot{\mathbf{q}}_\alpha} \quad , \tag{3}$$

which are denoted with an additional $\hat{(\cdot)}$ to distinguish them from the internal forces $\boldsymbol{\tau}_\alpha$ obtained with the free energy. To construct a constitutive model that complies with in Eq. (1), the dissipation potential must satisfy all of the following three requirements:

(i) $\phi(\dot{\mathbf{q}}_\alpha, \mathbf{q}_\alpha, \boldsymbol{\varepsilon})$ must be convex in all $\dot{\mathbf{q}}_\alpha$,
(ii) $\phi(\dot{\mathbf{q}}_\alpha = \mathbf{0}, \mathbf{q}_\alpha, \boldsymbol{\varepsilon}) = 0$ and
(iii) $\phi(\dot{\mathbf{q}}_\alpha, \mathbf{q}_\alpha, \boldsymbol{\varepsilon}) \geq 0$.

The material law is formulated with the *Biot equation* [66], that relates both potentials via

$$\frac{\partial \psi}{\partial \mathbf{q}_\alpha} + \frac{\partial \phi}{\partial \dot{\mathbf{q}}_\alpha} = \mathbf{0} \quad \text{or equivalently} \quad \boldsymbol{\tau}_\alpha = \hat{\boldsymbol{\tau}}_\alpha. \tag{4}$$

Evaluating Eq. (4) gives rise to the evolution laws [67] and is thus the *evolution equation* for the internal variables given in an *implicit form*.[1] The stated conditions on ϕ automatically construct these evolution laws such that inequality Eq. (1) is satisfied. This framework of GSMs contains various classical constitutive models, including the viscoelastic reference model in Sect. 5.1.1 that is used to generate the data for the numerical experiments.

3 Formulation of the physics-augmented neural network-based model

Based on the theoretical background from Sect. 2, an NN-based constitutive model for viscoelastic materials is presented. It uses a single internal variable $\mathbf{q} \in \mathcal{S}ym_2$ of strain type to model the inelastic behavior. The model adapts the concept of generalized standard materials, where the two potentials are expressed as FNNs with incorporated *convexity* constraints [20, 25, 37, 39, 43, 49, 50], see App. A. First, the modeling approaches for free energy and dissipation potential are described, followed by the prediction process using the adapted free energy and dissipation potential. The training methods to find the weights and biases of the FNNs without requiring the internal variable in the training data set are explained in Sect. 4.

[1] In Haupt [1], it is postulated that the internal variables' evolution is described by a system of ordinary differential equations (ODEs) of the form $\dot{\mathbf{q}}_\alpha = \boldsymbol{f}_\alpha(\boldsymbol{\varepsilon}, \mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n)$, referred to as evolution equations. We refer to Eq. (4) as an implicitly given evolution equation, since the given form of the ODEs cannot be explicitly determined for every permissible potential ϕ. The Biot equation is nevertheless equivalent.

3.1 Modeling of the potentials

The potentials ψ and ϕ can be either expressed in terms of tensor coordinates ($\psi(\boldsymbol{\varepsilon}, \mathbf{q})$ and $\phi(\dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\varepsilon})$) or with a set of invariants of those tensors ($\psi(\mathcal{I}^{\psi})$ and $\phi(\mathcal{I}^{\phi})$) [20, 57, 68]. Since some details must be taken into account in the invariant formulation, this formulation is explained in more detail below. The coordinate formulation is obtained analogously with the corresponding simplifications.

3.1.1 Free energy

The free energy is additively decomposed into an equilibrium part $\psi^{\mathrm{eq}}(\boldsymbol{\varepsilon})$ and a non-equilibrium part $\psi^{\mathrm{ov}}(\boldsymbol{\varepsilon}, \mathbf{q})$, i.e., $\psi(\boldsymbol{\varepsilon}, \mathbf{q}) = \psi^{\mathrm{eq}}(\boldsymbol{\varepsilon}) + \psi^{\mathrm{ov}}(\boldsymbol{\varepsilon}, \mathbf{q})$. For the non-equilibrium part it is assumed, that the dependency on $(\boldsymbol{\varepsilon}, \mathbf{q})$ can be expressed as $\psi^{\mathrm{ov}}(\boldsymbol{\varepsilon}, \mathbf{q}) = \psi^{\mathrm{ov}}(\mathbf{p}(\boldsymbol{\varepsilon}, \mathbf{q}))$, where $\mathbf{p} = \boldsymbol{\varepsilon} - \mathbf{q}$ [39]. The free energy is formulated with the following assumptions:

(i) ψ is an isotropic tensor function.
(ii) In the initial state $\boldsymbol{\varepsilon} = \mathbf{q} = \mathbf{0}$, the free energy vanishes, i.e., $\psi(\mathbf{0}, \mathbf{0}) = 0$.
(iii) In the initial state, the stress vanishes, i.e., $\partial_{\boldsymbol{\varepsilon}} \psi(\mathbf{0}, \mathbf{0}) = \mathbf{0}$.
(iv) In the initial state, the internal force vanishes, i.e., $-\partial_{\mathbf{q}} \psi(\mathbf{0}, \mathbf{0}) = \mathbf{0}$.
(v) $\psi(\boldsymbol{\varepsilon}, \mathbf{q})$ is assumed to be convex in $\boldsymbol{\varepsilon}$ and $\mathbf{q}$.

Condition (v) is used for several reasons. On the one hand, the convexity of ψ results in the positive definiteness of the material tangent $\partial_{\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}} \psi$, which offers numerical advantages in the application [69]. On the other hand, the resulting restrictions simplify the training of the model. Furthermore, (v) is fulfilled for the used reference model from Sect. 5.1.1. Furterhmore, it should be noted that other works assume the convexity of ψ as well [39, 70].

These constraints have to be incorporated into the FNN representation of ψ, where ψ^{eq} and ψ^{ov} are each described by an FNN. These FNNs are denoted as $\tilde{\psi}^{\mathrm{eq}}$ and $\tilde{\psi}^{\mathrm{ov}}$, respectively. Requirements (i)–(v) can be satisfied as follows:

(i) In order to ensure an isotropic free energy, ψ^{eq} and ψ^{ov} are expressed in terms of invariants of their tensor arguments [20], summarized in the invariant set $\mathcal{I}^{\psi^{\mathrm{eq}}} = (I_\alpha^{\psi^{\mathrm{eq}}})_{\alpha=1}^{3}$ and $\mathcal{I}^{\psi^{\mathrm{ov}}} = (I_\alpha^{\psi^{\mathrm{ov}}})_{\alpha=1}^{3}$, resp. That is, $\psi^{\mathrm{eq}}(\boldsymbol{\varepsilon})$ becomes $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$ and $\psi^{\mathrm{ov}}(\mathbf{p})$ becomes $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}})$ with

$$\mathcal{I}^{\psi^{\mathrm{eq}}}(\boldsymbol{\varepsilon}) = (\operatorname{tr}\boldsymbol{\varepsilon},\ \operatorname{tr}\boldsymbol{\varepsilon}^2,\ \operatorname{tr}\boldsymbol{\varepsilon}^4) \tag{5}$$

$$\mathcal{I}^{\psi^{\mathrm{ov}}}(\mathbf{p}) = (\operatorname{tr}\mathbf{p},\ \operatorname{tr}\mathbf{p}^2,\ \operatorname{tr}\mathbf{p}^4)\ . \tag{6}$$

The specific choice of the invariant sets is explained in (v), where the convexity properties are discussed.

(ii) In order to ensure $\psi(\mathbf{0}, \mathbf{0}) = 0$, each part $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}(\boldsymbol{\varepsilon}))$ and $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}(\mathbf{p}))$ is set to zero in the initial state, i.e., $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}(\mathbf{0})) = 0$ and $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}(\mathbf{0})) = 0$. These requirements are not incorporated into the network functions themselves. Instead, a correction term is appended to the definitions of $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$ and $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}})$, that is, $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) = \tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) - \psi_0^{\mathrm{eq}}$ and $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) = \tilde{\psi}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) - \psi_0^{\mathrm{ov}}$, where $\psi_0^{\mathrm{eq}} = \tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}(\boldsymbol{\varepsilon} = \mathbf{0}))$ and $\psi_0^{\mathrm{eq}} = \tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{ov}}}(\mathbf{p} = \mathbf{0}))$.

(iii) For the stress to vanish in the initial state, the equilibrium stress $\partial_{\boldsymbol{\varepsilon}} \psi^{\mathrm{eq}}$ and the overstress $\partial_{\boldsymbol{\varepsilon}} \psi^{\mathrm{ov}}$ must vanish. This is enforced with another set of correction terms $\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}$ and $\psi_{\boldsymbol{\sigma}}^{\mathrm{ov}}$, such that $\psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) = \tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) - \psi_0^{\mathrm{eq}} - \psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$ and $\psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) = \tilde{\psi}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) - \psi_0^{\mathrm{ov}} - \psi_{\boldsymbol{\sigma}}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}})$ [20, 24, 56]. These correction terms are defined such that, when differentiated with respect to $\boldsymbol{\varepsilon}$, they compensate the error in the initial stress prediction by $\tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$ or $\tilde{\psi}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}})$, respectively. A possible, but inconvenient way to achieve this is to set

$$\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}, \boldsymbol{\varepsilon}) = \sum_{\alpha=1}^{3} \left[\left. \frac{\partial \tilde{\psi}^{\mathrm{eq}}}{\partial I_\alpha^{\psi^{\mathrm{eq}}}} \right|_{\boldsymbol{\varepsilon}=\mathbf{0}} \left. \frac{\partial I_\alpha^{\psi^{\mathrm{eq}}}}{\partial \boldsymbol{\varepsilon}} \right|_{\boldsymbol{\varepsilon}=\mathbf{0}} \right] : \boldsymbol{\varepsilon} \tag{7}$$

and analogous for ψ^{ov}. However, this leads to a free energy that depends on not only the invariants, but also on the strain tensor itself, which may violate (i). Therefore, $\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$ and $\psi_{\boldsymbol{\sigma}}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}})$ may only depend on the invariants. This is achieved with

$$\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) = \left. \frac{\partial \tilde{\psi}^{\mathrm{eq}}}{\partial I_1^{\psi^{\mathrm{eq}}}} \right|_{\boldsymbol{\varepsilon}=\mathbf{0}} I_1^{\psi^{\mathrm{eq}}} \quad \text{and} \tag{8}$$

$$\psi_{\boldsymbol{\sigma}}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) = \left. \frac{\partial \tilde{\psi}^{\mathrm{ov}}}{\partial I_1^{\psi^{\mathrm{ov}}}} \right|_{\mathbf{p}=\mathbf{0}} I_1^{\psi^{\mathrm{ov}}}\ . \tag{9}$$

Since $I_1^{\psi^{\mathrm{eq}}} = \operatorname{tr}\boldsymbol{\varepsilon}$ and $I_1^{\psi^{\mathrm{ov}}} = \operatorname{tr}\mathbf{p}$ are linear functions in $\boldsymbol{\varepsilon}$ and $\mathbf{p}$, this does not effect convexity of ψ.

(iv) The internal force vanishes, if $-\partial_{\mathbf{q}} \psi^{\mathrm{ov}}\big|_{\mathbf{p}=\mathbf{0}} = \mathbf{0}$. Since $-\partial_{\mathbf{q}} \psi^{\mathrm{ov}} = \partial_{\boldsymbol{\varepsilon}} \psi^{\mathrm{ov}}$ holds and $\partial_{\boldsymbol{\varepsilon}} \psi^{\mathrm{ov}}\big|_{\mathbf{p}=\mathbf{0}} = \mathbf{0}$ is already assured with (iii), this requirement is already secured.

(v) Convexity in $\boldsymbol{\varepsilon}$ and $\mathbf{q}$ can be achieved if two conditions are met: (a) The chosen invariants are convex with respect to $\boldsymbol{\varepsilon}$ and $\mathbf{q}$ and (b) $\tilde{\psi}^{\mathrm{eq}}$ and $\tilde{\psi}^{\mathrm{ov}}$ are convex and non-decreasing in those invariants [29]. Since both $\psi^{\mathrm{eq}}(\boldsymbol{\varepsilon})$ and $\psi^{\mathrm{ov}}(\mathbf{p})$ depend on a single symmetric tensor of rank two, a complete set of invariants composes, e.g., the invariants $(\operatorname{tr}\mathbf{S},\ \operatorname{tr}\mathbf{S}^2,\ \operatorname{tr}\mathbf{S}^3)$, where

$\mathbf{S} \in \mathcal{S}ym_2$. The third invariant $\operatorname{tr}\mathbf{S}^3$, however, is not convex in $\mathbf{S}$, as shown in App. B. Therefore, the functional basis is adjusted suitably by replacing $\operatorname{tr}\mathbf{S}^3$ by the convex invariant $\operatorname{tr}\mathbf{S}^4$. This is allowed, since $\operatorname{tr}\mathbf{S}^3$ can be expressed by $\operatorname{tr}\mathbf{S}$, $\operatorname{tr}\mathbf{S}^2$ and $\operatorname{tr}\mathbf{S}^4$ using the Cayley-Hamilton theorem and the original basis can be recovered. A comment on the convexity of the used invariants can be found in Appendix B. Consequently, the invariant bases in Eqs. (5) and (6) allow for the construction of convex functions using *non-decreasing fully input convex neural networks(FICNNs)*[2] [20, 29, 49, 71] as described in App. A.1.1.

Summarizing, the free energy is modeled with an additive decomposition, where each part is a non-decreasing FICNN with additional correction terms and is expressed in a set of three convex invariants, such that

$$\psi(\boldsymbol{\varepsilon}, \mathbf{q}) = \psi^{\mathrm{eq}}(\boldsymbol{\varepsilon}) + \psi^{\mathrm{ov}}(\mathbf{p}(\boldsymbol{\varepsilon}, \mathbf{q})), \quad \text{where} \tag{10}$$

$$\psi^{\mathrm{eq}}(\boldsymbol{\varepsilon}) = \psi^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}(\boldsymbol{\varepsilon})) = \tilde{\psi}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}}) - \psi_0^{\mathrm{eq}} - \psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}(\mathcal{I}^{\psi^{\mathrm{eq}}})$$

and (11)

$$\psi^{\mathrm{ov}}(\mathbf{p}) = \psi^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}(\mathbf{p})) = \tilde{\psi}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}) - \psi_0^{\mathrm{ov}} - \psi_{\boldsymbol{\sigma}}^{\mathrm{ov}}(\mathcal{I}^{\psi^{\mathrm{ov}}}). \tag{12}$$

With that, the free energy representation is physically meaningful, even in the untrained state of the NN model. Details on the hyperparameters of the FICNNs can be found in Sec. A.3 To simplify notation, the weights and biases of the two FICNNs are summarized in a single parameter set w^{ψ}.

The dissipation potential is constructed using similar techniques.

3.1.2 Dissipation potential

The NN-based dissipation potential depends on $(\dot{\mathbf{q}}, \boldsymbol{\varepsilon}, \mathbf{q})$ [17]. Similar to the non-equilibrium part of the free energy, the dependency of $\boldsymbol{\varepsilon}$ and $\mathbf{q}$ is expressed in terms of $\mathbf{p} = \boldsymbol{\varepsilon} - \mathbf{q}$. Consequently, $\phi = \phi(\dot{\mathbf{q}}, \mathbf{p}(\boldsymbol{\varepsilon}, \mathbf{q}))$. As discussed in Sect. 2, the dissipation potential must obey some constraints:

(a) ϕ is an isotropic tensor function.
(b) ϕ is convex with respect to its first argument, the rate of the internal variable $\dot{\mathbf{q}}$.
(c) If the internal variable does not evolve, the dissipation potential vanishes, i.e., $\phi(\mathbf{0}, \mathbf{p}) = 0$.
(d) If the inelastic strain does not evolve, the dissipation potential is in a minimum, i.e., $\partial_{\dot{\mathbf{q}}}\phi(\mathbf{0}, \mathbf{p}) = \mathbf{0}$.

The respective network to model this function is denoted as $\tilde{\phi}$. The stated requirements are enforced with techniques similar to those used to construct the free energy.

(a) Formulating ϕ in invariants enforces isotropy of the resulting function. The network $\tilde{\phi}(\mathcal{I}^{\phi}(\dot{\mathbf{q}}, \mathbf{p}))$ depends on the six invariants

$$\mathcal{I}^{\phi}(\dot{\mathbf{q}}, \mathbf{p}) = (\operatorname{tr}\dot{\mathbf{q}},\ \operatorname{tr}\dot{\mathbf{q}}^2,\ \operatorname{tr}\dot{\mathbf{q}}^4,\ \operatorname{tr}\mathbf{p},\ \operatorname{tr}\mathbf{p}^2,\ \operatorname{tr}\mathbf{p}^3). \tag{13}$$

Here, no mixed invariants between $\dot{\mathbf{q}}$ and $\mathbf{p}$ are used.
(b) To enforce convexity with respect to the rate of the internal variable, a *non-decreasing partially input convex neural network (PICNN)* [49, 50] is used, which is convex with respect to the three invariants $(\operatorname{tr}\dot{\mathbf{q}},\ \operatorname{tr}\dot{\mathbf{q}}^2,\ \operatorname{tr}\dot{\mathbf{q}}^4)$ and not necessarily convex with respect to $(\operatorname{tr}\mathbf{p},\ \operatorname{tr}\mathbf{p}^2,\ \operatorname{tr}\mathbf{p}^3)$.
(c) Analogous to the free energy, a correction term ϕ_0 is appended to the definition of ϕ, such that $\phi = \tilde{\phi} - \phi_0$. ϕ_0 compensates the offset in $\tilde{\phi}$ and is defined as $\phi_0 = \tilde{\phi}(\mathcal{I}^{\phi}(\dot{\mathbf{q}} = \mathbf{0}, \mathbf{p}))$.
(d) By adapting the stress correction term for the free energy, a correction term $\phi_{\boldsymbol{\tau}}$ is appended, such that $\phi = \tilde{\phi} - \phi_0 - \phi_{\boldsymbol{\tau}}$, where $\phi_{\boldsymbol{\tau}}$ is

$$\phi_{\boldsymbol{\tau}} = \left.\frac{\partial\tilde{\phi}}{\partial I_1^{\phi}}\right|_{\dot{\mathbf{q}}=\mathbf{0}} I_1^{\phi}. \tag{14}$$

Summarizing, the dissipation potential is modeled with a non-decreasing PICNN and additional correction terms. The dependency on $\dot{\mathbf{q}}$ is expressed with convex invariants of $\dot{\mathbf{q}}$. That is,

$$\phi(\dot{\mathbf{q}}, \boldsymbol{\varepsilon}, \mathbf{q}) = \tilde{\phi}(\mathcal{I}^{\phi}(\dot{\mathbf{q}}, \mathbf{p}(\boldsymbol{\varepsilon}, \mathbf{q}))) - \phi_0 - \phi_{\boldsymbol{\tau}} \quad . \tag{15}$$

The NN model is thus *thermodynamically consistent* and *isotropic* by construction. All weights and biases of the PICNN with the hyperparamters from Sec. A.3 are summarized in the parameter set w^{ϕ}.

3.1.3 Coordinate formulation

The previous explanations refer to the invariant formulation. For the formulation with coordinates [39], requirements (i) and (a), i.e., the isotropy of the potentials, cannot be fulfilled. In principle, the same techniques are used to comply with the remaining requirements, but instead of convex and non-decreasing NNs, only convex NNs are used, as the tensors themselves are inputs of the networks. The coordinate formulations of the additional terms for zero energy and zero

[2] Since the invariants $\operatorname{tr}\boldsymbol{\varepsilon}$ and $\operatorname{tr}\mathbf{p}$ are linear functions of $\boldsymbol{\varepsilon}$ and $\mathbf{q}$, the constraints on the corresponding weights in the passthrough layers of the non-decreasing FICNN as described in Sec. A are in general too restrictive. However, for the here shown NN-based model and the used reference material, this does not effect the prediction quality and is henceforth ignored.

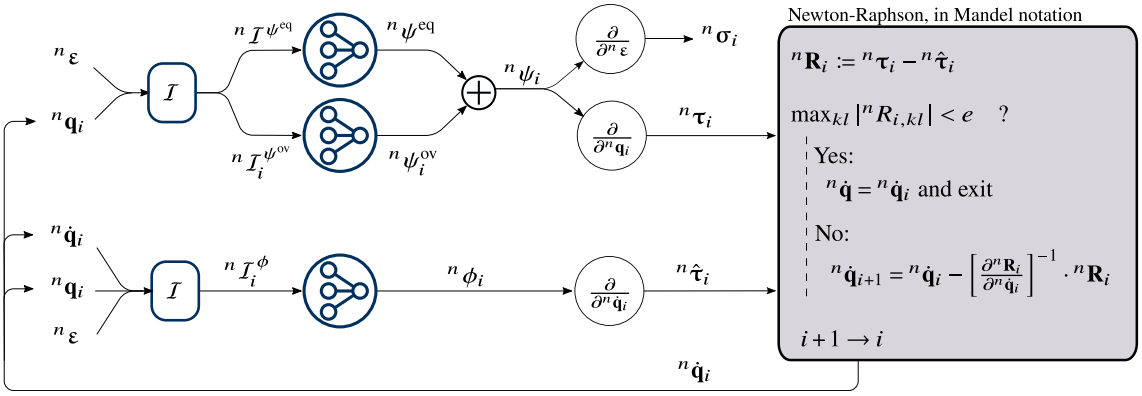


Fig. 2 Structure of the invariant-based two-potential model and prediction process. The rate of the internal variables is determined iteratively so that the internal stress $\boldsymbol{\tau}$ calculated from the free energy and the internal stress $\hat{\boldsymbol{\tau}}$ calculated from the dissipation potential are equal within a specified tolerance e. Note that the invariant set ${}^n\mathcal{I}^{\psi^{\mathrm{eq}}} = (\mathrm{tr}\,({}^n\boldsymbol{\varepsilon}), \mathrm{tr}\,({}^n\boldsymbol{\varepsilon}^2), \mathrm{tr}\,({}^n\boldsymbol{\varepsilon}^4))$ as input for the ψ^{eq} network has no subscript i, since it only depends on ${}^n\boldsymbol{\varepsilon}$

stress in the initial state are briefly explained using the example of ψ^{eq}: The term ψ_0^{eq} for zero energy in the initial state is calculated as

$$\psi_0^{\mathrm{eq}} = \tilde{\psi}^{\mathrm{eq}}(\boldsymbol{\varepsilon} = \mathbf{0}) \tag{16}$$

and the term $\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}}$ for the stress free initial state becomes

$$\psi_{\boldsymbol{\sigma}}^{\mathrm{eq}} = \left.\frac{\partial \tilde{\psi}^{\mathrm{eq}}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}=\mathbf{0}} : \boldsymbol{\varepsilon}. \tag{17}$$

The other additional terms for ψ^{ov} and ϕ are calculated analogously.

3.2 Incremental predictions with the trained models

Once the weights and biases of the networks for free energy and dissipation potential have been adapted, the constitutive model can be used to predict the constitutive response to a prescribed strain path consisting of multiple discrete time steps. In order to solve the evolution equation numerically, an *implicit Euler* scheme is applied for the temporal discretization, in which the rate of the internal variable is approximated as ${}^n\dot{\mathbf{q}} \approx ({}^n\mathbf{q} - {}^{n-1}\mathbf{q})/{}^n\Delta t$, where the superscript n corresponds to the n-th time step of a sequence and ${}^n\Delta t = {}^nt - {}^{n-1}t$ is the time increment. This ensures stability of the evolution of $\mathbf{q}$. In each time step, the rate of the internal variable is adapted iteratively, such that Eq. (4) is fulfilled up to a prescribed tolerance. That is, the resulting internal force, which is calculated from the free energy, i.e., $\boldsymbol{\tau} = -\partial_{\mathbf{q}}\psi$, and the internal force calculated from the dissipation potential, i.e., $\hat{\boldsymbol{\tau}} = \partial_{\dot{\mathbf{q}}}\phi$, are supposed to be identical within a given tolerance. This iterative solution of Eq. (4) is done using a *Newton–Raphson scheme*, as depicted in Fig. 2.

4 Training methods

Since the internal variable and its rate are arguments of the free energy and the dissipation potential, they must be provided in the training process to render training possible. In the following, however, it is assumed, that they are not present in the training data set. For data obtained through homogenization, a few approaches have been developed to enable the determination of internal variables using autoencoders [54, 55]. The training data set $\mathscr{D} = (\boldsymbol{\varepsilon}(t), \boldsymbol{\sigma}(t), \mathbf{q}(t))$ then comprises sequences of stresses, strains and internal variable(s). In real experiments, however, the determination of internal variables is in general not possible, shrinking the data set $\mathscr{D} = (\boldsymbol{\varepsilon}(t), \boldsymbol{\sigma}(t))$ to sequences of only stresses and strains. This section introduces three methods to address this issue. Two methods from the literature determine the internal variable by integrating the evolution equation over the entire sequence or by means of an additional auxiliary network in the form of an FNN. In the newly developed third method, the auxiliary network is replaced by an RNN.

4.1 Integration

The training approach denoted as integration in the following is probably the most intuitive among the three presented and has been applied in [37, 57]. In every training epoch, the stress response to individual sequences is determined following the scheme provided in Figs. 2 and 3. Starting from the initial state ${}^0\boldsymbol{\varepsilon} = {}^0\boldsymbol{\sigma} = {}^0\mathbf{q} = \mathbf{0}$, the stress ${}^1\boldsymbol{\sigma}$ for the new strain ${}^1\boldsymbol{\varepsilon}$ and the corresponding time step ${}^1\Delta t$ is calculated. The obtained internal variable is then passed on to the next time step and so forth until the last time step of the sequence is reached. The stresses $\boldsymbol{\sigma}$ determined this way are compared to the expected stresses $\bar{\boldsymbol{\sigma}}$ from the training data set using the

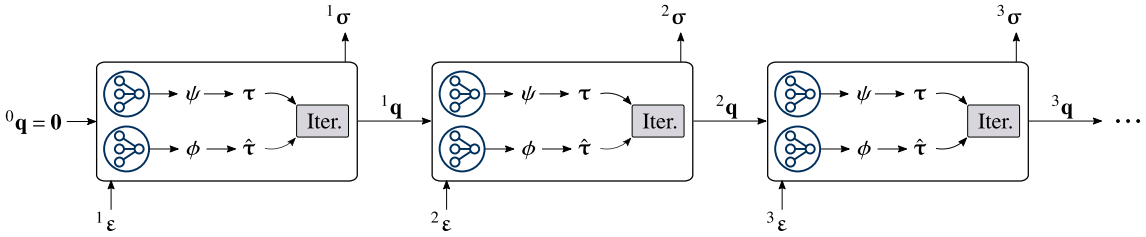


Fig. 3 Schematic representation of the training process using the integration strategy. In each time step, the new material state is obtained iteratively with the procedure given in Fig. 2. The internal variable is passed on to every time step as starting point for the next integration step until the last time step of the sequence is reached. Consequently, the calculation of the stress for time step n requires the evaluation of all time steps $1, 2, \ldots, n-1$ in advance

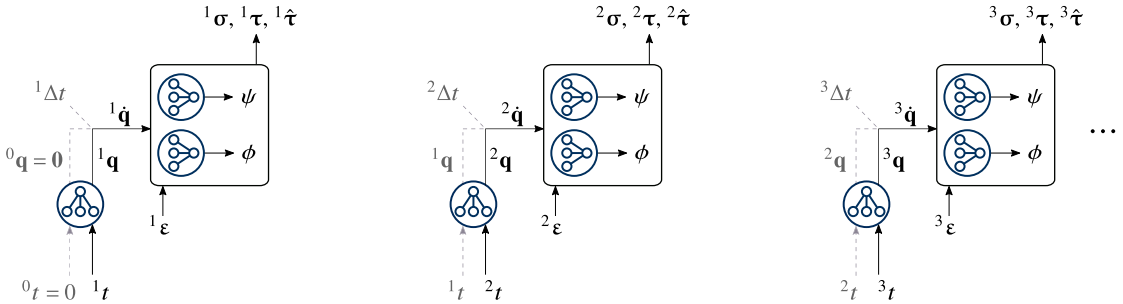


Fig. 4 Schematic representation of the training process using an FNN as auxiliary network for the internal variable. This FNN receives a single input, the time ${}^{n}t$, and outputs the six independent entries of ${}^{n}\mathbf{q}$. To calculate the rate ${}^{n}\dot{\mathbf{q}} \approx {}^{n}\mathbf{q} - {}^{n-1}\mathbf{q})/{}^{n}\Delta t$, ${}^{n-1}\mathbf{q}$ is obtained by evaluating this FNN with ${}^{n-1}t$ as input. Note that each time step can be evaluated detached from the others, which allows fast training and the creation of minibatches within a sequence, if required

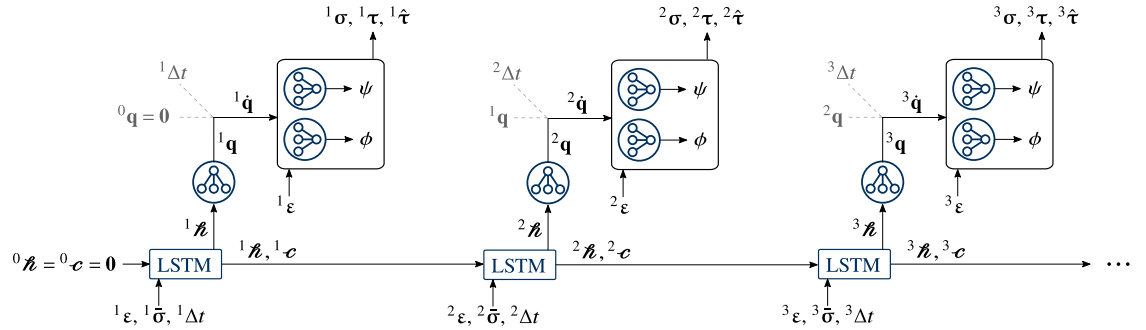


Fig. 5 Schematic representation of the training process using an RNN as auxiliary network for the internal variable. In each time step n, this RNN receives three inputs: the strain ${}^{n}\boldsymbol{\varepsilon}$, the stress ${}^{n}\boldsymbol{\sigma}$ and the time increment ${}^{n}\Delta t$. Together with the hidden state ${}^{n-1}\boldsymbol{h}$ and cell state ${}^{n-1}\boldsymbol{c}$ from the previous time step, the RNN-cell processes these data into a new hidden state, that contains the information about the new internal variable. This new hidden state is forwarded to an FNN, that reduces the dimensionality to six for the six independent entries of ${}^{n}\mathbf{q}$. The new hidden state and cell state are passed on to the next time step, until the final time step of the sequence is reached

loss function

$$\mathcal{L} = \mathcal{L}^{\sigma} \quad \text{with} \quad \mathcal{L}^{\sigma} = \mathrm{MAE}(\boldsymbol{\sigma}, \bar{\boldsymbol{\sigma}})/s_{\boldsymbol{\sigma}} \quad , \tag{18}$$

where $\mathrm{MAE}(\boldsymbol{\sigma}, \bar{\boldsymbol{\sigma}})$ is the mean absolute error between predicted stress $\boldsymbol{\sigma}$ and expected stress $\bar{\boldsymbol{\sigma}}$. Compared to the mean squared error, the mean absolute error performed best for the problems in this article and is therefore used within the scope of this work. Furthermore, $s_{\boldsymbol{\sigma}}$ is the normalization factor for the stress, that normalizes the loss to magnitude 1. The calculation of $s_{\boldsymbol{\sigma}}$ is explained in A.2. To adapt the weights and biases of the networks, the loss function Eq. (18) is minimized in the optimization problem

$$(\hat{w}^{\psi}, \hat{w}^{\phi}) = \underset{w^{\psi} \in \mathcal{C}^{\psi}, w^{\phi} \in \mathcal{C}^{\phi}}{\arg\min} \mathcal{L} \, , \tag{19}$$

where the parameters w^ψ and w^ϕ correspond to the weights and biases of the two FICNNs of the free energy and the PICNN of the dissipation potential. They are constrained by the sets $\mathcal{C}^\psi$ and $\mathcal{C}^\phi$ containing restrictions for weights and biases of these ICNNs as described in App. A. Due to the iterative solution scheme, the evaluation of the loss function is very expensive [37]. Hence, an optimizer with fast convergence is favorable. The optimizer Sequential Least Squares Programming (SLSQP) has shown to perform well for this application [17, 20, 26, 72] and is therefore used in the numerical examples in Sect. 5.

4.2 Auxiliary feedforward network

To avoid the challenges of the integration method, another method has been proposed by As'ad and Farhat [39] and is adopted with slight modifications herein [17]. This method introduces an additional FNN, that is supposed to learn the temporal course of the internal variable. Let $\tilde{\mathbf{q}}(t)$ denote this additional FNN. It depends on only a single input, the time t, and has six outputs corresponding to the six independent entries of the symmetric tensor $\mathbf{q}$, as shown in Fig. 4. For the training of the networks for free energy and dissipation potential, values for the internal variable and its rate are taken from this network, where the rate is approximated as ${}^n\dot{\mathbf{q}} \approx ({}^n\mathbf{q} - {}^{n-1}\mathbf{q})/{}^n\Delta t$ to be consistent with the prediction process. The weights and biases of $\tilde{\mathbf{q}}$ are adapted such that the predicted temporal course of $\tilde{\mathbf{q}}(t)$ allows the loss function

$$\mathcal{L} = \mathcal{L}^{\boldsymbol{\sigma}} + \mathcal{L}^{\mathrm{Biot}} \quad \text{with} \tag{20}$$

$$\mathcal{L}^{\boldsymbol{\sigma}} = \mathrm{MAE}(\boldsymbol{\sigma}, \bar{\boldsymbol{\sigma}})/s_{\boldsymbol{\sigma}} \quad \text{and} \quad \mathcal{L}^{\mathrm{Biot}} = \mathrm{MAE}(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}})/s_{\boldsymbol{\sigma}} \tag{21}$$

to become as small as possible. The loss function consists of a term $\mathcal{L}^{\boldsymbol{\sigma}}$ for the accurate stress prediction and another $\mathcal{L}^{\mathrm{Biot}}$ term to comply with the Biot relation Eq. (4) [17]. Thus, the reformulated optimization problem reads

$$(\hat{w}^\psi, \hat{w}^\phi, \hat{w}^q) = \underset{w^\psi \in \mathcal{C}^\psi, w^\phi \in \mathcal{C}^\phi, w^q}{\arg\min} (\mathcal{L}) \ , \tag{22}$$

where the parameter set w^q contains weights and biases of the auxiliary FNN as specified in Sec. A.3. This leads to a reasonable representation of the internal variable without explicitly specifying its value in advance or having to integrate every sequence in every iteration of training. However, we have found that starting the optimization with randomly initialized weights for $\tilde{\mathbf{q}}$ makes the problem too complex for common optimizers and does not lead to the desired results. In order to facilitate the training, the auxiliary network is pre-trained using the strain $\boldsymbol{\varepsilon}(t)$ as an initial guess for $\tilde{\mathbf{q}}(t)$. Once the pre-training and subsequent actual training is finished, the auxiliary network is no longer necessary and the prediction process can be carried out using only free energy and dissipation potential.

Remark 1 With this architecture, the temporal course of the internal variable for a single sequence can be modeled via a single FNN. However, if data from several sequences are available in the data set, another FNN must be added for each additional sequence, making the optimization problem increasingly difficult.

4.3 Auxiliary recurrent network

RNNs have proven to be particularly suitable for describing the behavior of path-dependent systems [10, 12, 14, 51]. Therefore, the auxiliary FNN is now to be replaced by an RNN. Using an RNN cell to generate the internal variable allows to use multiple sequences for training without requiring a new auxiliary network for every training sequence. We use an LSTM cell [18] as RNN cell. This LSTM cell has two different state vectors, that allow the cell to store information from past time steps. These state vectors are the hidden state $\hbar$ and the cell state c, where $\hbar$ is the output of the cell for every time step. The training with the auxiliary RNN works as shown in Fig. 5: In each time step, the RNN cell receives the current strain ${}^n\boldsymbol{\varepsilon}$, the associated stress ${}^n\boldsymbol{\sigma}$ from the data set as well as the time step ${}^n\Delta t$. An FNN reduces the output of the RNN cell, the hidden state ${}^n\hbar$, to 6 entries corresponding to the 6 independent entries of ${}^n\mathbf{q}$. The rate of the internal variable ${}^n\dot{\mathbf{q}}$ is calculated using ${}^n\Delta t$ and ${}^{n-1}\mathbf{q}$ from the previous time step. Through differentiation of $\psi({}^n\boldsymbol{\varepsilon}, {}^n\mathbf{q})$ and $\phi({}^n\dot{\mathbf{q}}, {}^n\boldsymbol{\varepsilon}, {}^n\mathbf{q})$, the stress ${}^n\boldsymbol{\sigma}$ and the internal forces ${}^n\boldsymbol{\tau}$ and ${}^n\hat{\boldsymbol{\tau}}$ are obtained. The status of the LSTM cell, i.e., ${}^n\hbar$ and ${}^nc$, is passed on to the next time step, to obtain the next material state and so on. Using the calculated stresses and internal forces, the loss function

$$\mathcal{L} = \mathcal{L}^{\boldsymbol{\sigma}} + \mathcal{L}^{\mathrm{Biot}} \quad \text{with} \tag{23}$$

$$\mathcal{L}^{\boldsymbol{\sigma}} = \mathrm{MAE}(\boldsymbol{\sigma}, \bar{\boldsymbol{\sigma}}) \quad \text{and} \quad \mathcal{L}^{\mathrm{Biot}} = \mathrm{MAE}(\boldsymbol{\tau}, \hat{\boldsymbol{\tau}}) \tag{24}$$

is evaluated. This loss function again contains a term for the error in the stress prediction and a term for compliance with the Biot equation. The optimization problem thus reads

$$(\hat{w}^\psi, \hat{w}^\phi, \hat{w}^q) = \underset{w^\psi \in \mathcal{C}^\psi, w^\phi \in \mathcal{C}^\phi, w^q}{\arg\min} (\mathcal{L}) \ , \tag{25}$$

where the weights and biases of the auxiliary LSTM and the connected FNN are summarized in the parameter set w^q. The concrete architecture of the LSTM and FNN can be found in Sec. A.3. Finally, after training, the RNN and the attached FNN can be discarded, leaving ψ and ϕ as constitutive model.

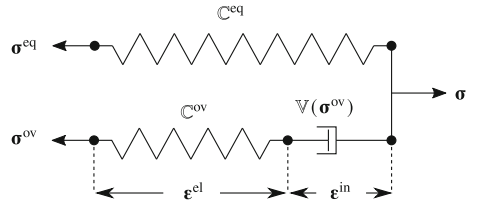


Fig. 6 Rheological model of the viscoelastic reference solid with a single internal variable $\mathbf{q} = \boldsymbol{\varepsilon}^{\mathrm{in}}$

5 Numerical examples

In the following, the presented framework is examined comprehensively. First, the reference material is introduced and several data sets are generated using this reference model. These data sets differ in terms of the material states they contain, for example multiaxial stress state or uniaxial stress states, and the presence of noise in the stress data. The data sets are then used as the basis for calibrating the model in different scenarios. An overview of the conducted experiments is given in Table 1.

5.1 Data base

5.1.1 Reference material

To generate the training data, a reference model consisting of a spring and a parallel Maxwell element as depicted in Fig. 6 is used. The free energy and the dissipation potential for such a model are defined as

$$\psi(\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}^{\mathrm{in}}) = \frac{1}{2}\boldsymbol{\varepsilon} : \mathbb{C}^{\mathrm{eq}} : \boldsymbol{\varepsilon} + \frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{el}} : \mathbb{C}^{\mathrm{ov}} : \boldsymbol{\varepsilon}^{\mathrm{el}} \quad \text{and} \tag{26}$$

$$\phi(\dot{\boldsymbol{\varepsilon}}^{\mathrm{in}}, \boldsymbol{\varepsilon}^{\mathrm{in}}, \boldsymbol{\varepsilon}) = \frac{1}{2}\dot{\boldsymbol{\varepsilon}}^{\mathrm{in}} : \mathbb{V}(\boldsymbol{\varepsilon}^{\mathrm{in}}, \boldsymbol{\varepsilon}) : \dot{\boldsymbol{\varepsilon}}^{\mathrm{in}}, \tag{27}$$

where $\dot{\boldsymbol{\varepsilon}}^{\mathrm{in}}$ is defined to be the internal variable, $\boldsymbol{\varepsilon}^{\mathrm{el}} = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{in}}$,

$$\mathbb{C}^{\mathrm{eq}} = 3K^{\mathrm{eq}}\mathbb{I}^{\mathrm{K}} + 2G^{\mathrm{eq}}\mathbb{I}^{\mathrm{D}} \quad \text{and} \quad \mathbb{C}^{\mathrm{ov}} = 3K^{\mathrm{ov}}\mathbb{I}^{\mathrm{K}} + 2G^{\mathrm{ov}}\mathbb{I}^{\mathrm{D}} \tag{28}$$

are the equilibrium and non-equilibrium stiffness tensors, respectively, and $\mathbb{V}(\boldsymbol{\varepsilon}^{\mathrm{in}}, \boldsymbol{\varepsilon})$ is a positive semidefinite fourth order tensor describing the viscous properties of the material. This tensor is not constant, but depends on the overstress $\boldsymbol{\sigma}^{\mathrm{ov}} = \mathbb{C}^{\mathrm{ov}} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{in}})$ via

$$\mathbb{V} = (1 - o)\mathbb{V}_0 \exp(-\|\frac{1}{a}\boldsymbol{\sigma}^{\mathrm{ov}}\|^b) + o \quad \text{with} \tag{29}$$

$$\mathbb{V}_0 = 3\eta^{\mathrm{K}}\mathbb{I}^{\mathrm{K}} + 2\eta^{\mathrm{D}}\mathbb{I}^{\mathrm{D}}. \tag{30}$$

This ansatz is motivated from [73]. The specific material parameters are given in Table 2.

5.1.2 Generation of the data base

The data base $\mathscr{D} = \{\, \boldsymbol{\varepsilon}(t), \boldsymbol{\sigma}(t) \,\}$ contains information about the temporal course of strain and stress for a single or multiple sequences. The generation of such sequences is explained in the subsequent paragraphs, where a distinction is made between training data with multi-, bi- and uniaxial stress states and test data to evaluate the prediction quality of trained models. To indicate the different data sets, a superscript and a subscript is appended to the symbol for the data set that describe the type of data and the number of sequences and timesteps. For example, the data set $\mathscr{D}^{\mathrm{multiax}}_{1\times 200}$ contains ideal multiaxial data in a single sequence of 200 timesteps.

Multiaxial training data Multiaxial training data sequences are obtained by prescribing a randomized strain path $\boldsymbol{\varepsilon}(t)$ and evaluating the respective stress response of the reference constitutive model from Sect. 5.1.1. A possible randomized strain path $\boldsymbol{\varepsilon}(t)$ is shown in Fig. 7. This strain path is created with cubic splines that connect a set of randomly sampled knots [17, 39]. Starting from $\varepsilon_{ij}^{\mathrm{knot}} = 0$ and $t^{\mathrm{knot}} = 0$, a time increment Δt^{knot} is sampled from a uniform distribution between $\Delta t^{\mathrm{knot}}_{\mathrm{min}} = 0.2\mathrm{s}$ and $\Delta t^{\mathrm{knot}}_{\mathrm{max}} = 1\mathrm{s}$. The increments of the six independent coordinates of the strain tensor are sampled from a normal distribution with standard deviation $s^{\mathrm{knot}}_{\Delta\varepsilon} = 0.5\mathrm{s}$ around mean 0. If the resulting absolute value of the strain $|\varepsilon_{ij}^{\mathrm{knot}} + \Delta\varepsilon_{ij}^{\mathrm{knot}}|$ exceeds 2s, the strain increment $\Delta\varepsilon^{\mathrm{knot}}$ is sampled again. Once this strain path $\boldsymbol{\varepsilon}(t)$ is generated, it is applied to the reference material with random time increments Δt from a uniform distribution between $\Delta t_{\mathrm{min}} = 0.03\mathrm{s}$ and $\Delta t_{\mathrm{max}} = 0.07\mathrm{s}$ to obtain the corresponding stresses $\boldsymbol{\sigma}(t)$. The symbol $\mathscr{D}^{\mathrm{multiax}}_{1\times 200}$ indicates such a data set with ideal multiaxial data.

Multiaxial training data with noise To investigate the performance of the training methods with non-ideal data, another training data set with noisy stress data is generated. This data set reuses the ideal multiaxial training data and modifies the stress data therein. To receive the noisy stress data, Gaussian noise is added to the ideal stress, such that $\tilde{\sigma}_{ij} = \sigma_{ij} + \Delta\sigma_{ij}$, where $\Delta\sigma_{ij}$ is sampled from a normal distribution with standard deviation $s^{\Delta\sigma} = 1.5\mathrm{MPa}$ and mean 0. The ideal and noisy data are shown in Fig. 7. To indicate the noise in the stress data, the symbol $\tilde{\mathscr{D}}^{\mathrm{multiax}}_{1\times 200}$ for this data set has an additional tilde.

Plane strain training data To generate the plane strain data set, the strain paths that were used to generate the multiaxial data set are reused. These strain paths are modified so that the corresponding coordinates are set to zero, i.e. $\varepsilon_{13}(t) = \varepsilon_{23}(t) = \varepsilon_{33}(t) = 0$. The stress response to these strain paths is calculated and the data set is assembled. Plane strain data sets are denoted as $\mathscr{D}^{\mathrm{plane}}_{1\times 200}$, for example.

Biaxial training data In order to analyze the applicability of the training methods with potential experimental data, a data

Table 1 Overview about the numerical examples in this section

Training data set							Methods		Figure
Name	Sequences	Timesteps per sequence	Noise	Material states	Driven by	**q** given	Model formulation	Training method	
$\mathscr{D}^{\text{multiax}}_{5\times200}$	5	200	No	Multiaxial	Strain	Yes	Invar./Coord	–	8
$\mathscr{D}^{\text{plane}}_{5\times200}$	5	200	No	Plane strain	Strain	Yes	Invar./Coord	–	8
$\mathscr{D}^{\text{multiax}}_{1\times200}$	1	200	No	Multiaxial	Strain	No	Invariants	Int./FNN/RNN	9/11
$\tilde{\mathscr{D}}^{\text{multiax}}_{1\times200}$	1	200	Yes	Multiaxial	Strain	No	Invariants	Int./FNN/RNN	9/10
$\mathscr{D}^{\text{multiax}}_{100\times100}$	100	100	No	Multiaxial	Strain	No	Invariants	RNN	12
$\mathscr{D}^{\text{biax}}_{1\times200}$	1	200	No	Biaxial	Stress	No	Invariants	RNN	13
$\mathscr{D}^{\text{uniax}}_{1\times200}$	1	200	No	Uniaxial	Stress	No	Invariants	RNN	13

Table 2 Material parameters for the viscoelastic reference material that is used to generate the training data

(K^{eq}, G^{eq}) in MPa	(K^{ov}, G^{ov}) in MPa	(η^{K}, η^{D}) in MPa. S^{-1}	(a, b, o) in MPa, -, -)
(500, 300)	(1000, 700)	(400, 200)	(10, 2, 0.1)

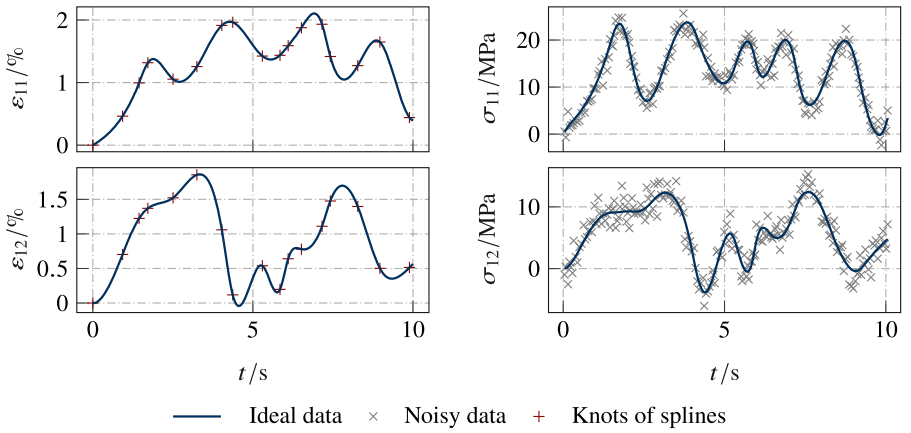


Fig. 7 A strain path $\boldsymbol{\varepsilon}(t)$ from the training data set with respective ideal stress response $\boldsymbol{\sigma}(t)$ for the data sets $\mathscr{D}^{\text{multiax}}$ and noisy data for the data set $\tilde{\mathscr{D}}^{\text{multiax}}$. The curve $\boldsymbol{\varepsilon}(t)$ is generated using cubic splines connecting a set of randomly sampled points, indicated as red plus signs

set with data from a simulated stress driven biaxial tension test is generated. In this biaxial tension test, the temporal course of the stress coordinates $\sigma_{11}(t) \neq \sigma_{22}(t)$ is prescribed and in general non-zero, while all other coordinates are $\sigma_{ij}(t) = 0$. The path of the non-zero stress coordinates is generated randomly in the same way as for the strain components in the multiaxial training data set. That is, cubic splines are used that connect randomly sampled knots with positions $(t, \sigma_{ij}^{\text{knot}})$. These random knots are generated with the time increments from a uniform distribution with $\Delta t_{\text{min}}^{\text{knot}} = 0.2\text{s}$ and $\Delta t_{\text{max}}^{\text{knot}} = 1\text{s}$, a standard deviation of the stress increments of $s_{\Delta\sigma} = 12.5\text{MPa}$ and a maximum absolute stress value for the knots of 25MPa. The stress path described by this cubic spline is evaluated with time increments Δt from a uniform distribution within $\Delta t_{\text{min}} = 0.03\text{s}$ and $\Delta t_{\text{max}} = 0.07\text{s}$ and the corresponding strain $\boldsymbol{\varepsilon}(t)$ is determined iteratively, such that the strain $\boldsymbol{\varepsilon}(t)$ results in the prescribed stress path $\boldsymbol{\sigma}(t)$. A biaxial data set is indicated with $\mathscr{D}^{\text{biax}}_{1\times200}$.

Uniaxial training data To simulate an uniaxial tension test, the course of the stress coordinate $\sigma_{11}(t)$ is prescribed and non-zero, while all other coordinates are $\sigma_{ij}(t) = 0$. The same technique as for the biaxial data set is used, i.e., cubic splines are generated for $\sigma_{11}(t)$. Here, the parameters $\Delta t_{\text{min}}^{\text{knot}} = 0.2\text{s}$ and $\Delta t_{\text{max}}^{\text{knot}} = 1\text{s}$ for the limits of Δt^{knot}, the standard deviation $s_{\Delta\sigma} = 25\text{MPa}$ and the maximum absolute value 50MPa for the stress are used. Again, the spline is scanned with $\Delta t_{\text{min}} = 0.03\text{s}$ and $\Delta t_{\text{max}} = 0.07\text{s}$ to obtain the strain $\boldsymbol{\varepsilon}(t)$. Uniaxial training data is denoted with $\mathscr{D}^{\text{uniax}}_{1\times200}$.

Test data Following the training, the models are tested with an unseen path, that is not part of the training data set. For this purpose, a strain path $\boldsymbol{\varepsilon}(t)$ is generated using the method for the generation of multiaxial training data. The corresponding stress response of the reference material is calculated. That is, the trained models are tested on an arbitrary multiaxial test path where all stress and strain coordinates are generally non-zero, i.e., $\varepsilon_{ij}(t) \neq 0$ and $\sigma_{ij}(t) \neq 0$. This test sequence is used as a benchmark for all trained models, regardless of the data set used.

5.2 Training with multiaxial data

The presented NN-based constitutive model and training methods are now to be tested in different scenarios. In the first part of the subsection, it is exploited that the used data is generated synthetically and it is assumed that the internal variable is given in the data set to compare the invariant formulation and the coordinate formulation of the potentials in a simple test case, i.e., $\mathscr{D} = (\boldsymbol{\varepsilon}(t), \boldsymbol{\sigma}(t), \mathbf{q}(t))$ additionally contains information about the internal variable. Afterwards, the internal variable is removed from the data set and the three training methods are compared for the invariant formulation, using both ideal and noisy stress data.

5.2.1 Comparison of invariant vs. coordinate formulation

To compare the model formulation using invariants and the formulation using tensor coordinates as inputs of the potentials, a data set $\mathscr{D}^{\text{multiax}}_{5\times200}$ with 5 sequences à 200 time steps and the corresponding plane strain data set $\mathscr{D}^{\text{plane}}_{5\times200}$ are used. These data sets are sufficiently large to show the full potential of both models and contain the internal variable. The training is carried out using the Adam optimizer with $20,000$ epochs, an initial learning rate of 0.01 and an exponential learning rate decay such that the learning rate is multiplied with 0.1 every $4,000$ epochs. To reduce the effect of the random weights initialization, both models are trained 25 times for each data set. The models with the lowest final value of the loss function are then tested on the unseen test strain path. The results are shown as scatter diagrams in Fig. 8a.

For the coordinate formulation, it can be seen that the model is able to make fairly accurate predictions for the test path based on data set $\mathscr{D}^{\text{multiax}}_{5\times200}$. Using data set $\mathscr{D}^{\text{plane}}_{5\times200}$, however, the model fails to extrapolate from the plane strain data to the full strain states. To illustrate this behavior, Fig. 8b shows the actual course of predicted stresses σ_{11}, σ_{12} and σ_{13}. It becomes evident that the large deviations from the expected values mainly concern the out-of-plane coordinates for which the training data set lacks sufficient data.

However, looking at the invariant formulation, it can be seen that it produces very accurate results regardless of the used data set, even exceeding the accuracy of the coordinate formulation with data set $\mathscr{D}^{\text{uniax}}_{5\times200}$. Thus, the invariant formulation enables extrapolation from plane strain data to full strain states without notable loss of accuracy. Such a well-developed extrapolation behavior also occurs with elastic NN models based on invariants [20, 26]. This benefit arises from the choice of a specific set of invariants that provides additional information about the anisotropy class of the underlying material law to the model. In fact, the change from plane strain to full strain states may not correspond to an extrapolation in the invariant space at all [28].

Table 3 Calculation time comparison of different training methods for the model for one epoch

	Integration	FNN	RNN	RNN
Data set	$\mathscr{D}^{\text{multiax}}_{1\times200}$	$\mathscr{D}^{\text{multiax}}_{1\times200}$	$\mathscr{D}^{\text{multiax}}_{1\times200}$	$\mathscr{D}^{\text{multiax}}_{100\times100}$
Time per epoch	35.0s	0.021s	0.059s	0.112s

Moreover, the invariant formulation requires significantly less training data. The decision to use 5 sequences of 200 time steps was made due to the larger amount of data needed for the coordinate formulation, whereas the invariant formulation can operate on a single sequence of 200 steps with similar prediction accuracy.

For this reason, only the invariant formulation is used in the following studies and the data set is reduced to one sequence of length 200.

5.2.2 Comparison of the three training methods

The internal variable is now removed from the data set and has to be generated during the training process. To do so, the methods described in Sect. 4 are applied, where the data sets $\mathscr{D}^{\text{multiax}}_{1\times200}$ and its noisy equivalent $\tilde{\mathscr{D}}^{\text{multiax}}_{1\times200}$ are used to compare the methods. Again, all models are trained 25 times and the weigths of the training run with the lowest final value of the loss function are chosen. The result graphs show scatter diagrams of the stress predictions for the unseen test path (Fig. 9), the stress predictions over time for the data set $\tilde{\mathscr{D}}^{\text{multiax}}_{1\times200}$ (Fig. 10) and the learned course of the internal variable for the training sequence as used in the last epoch of training (Fig. 11). The history of the losses over iterations for the different methods are shown in Appendix C, exemplarily on the example of the data set $\mathscr{D}^{\text{multiax}}_{1\times200}$.

Integration

The integration training method yields highly accurate results for the ideal data set and outperforms the other methods in terms of accuracy. However, it should be noted that this method uses the SLSQP optimizer for computational reasons, which has been shown to provide better results for smaller networks than Adam [72]. Since no additional network is required for this method, SLSQP can be used in a computationally efficient manner with fewer function evaluations compared to Adam. Nevertheless, the evaluation of the loss function consumes the majority of computational time for training, since an iterative solution is required at each time step, regardless of the optimizer, see Fig. 3. Adding another sequence to the data set or doubling its length roughly doubles the cost for the evaluation of the loss function, making it more and more inefficient. However, very good results can be achieved with both the ideal training data as well as the noisy training data set as Figs. 9 and 10 show.

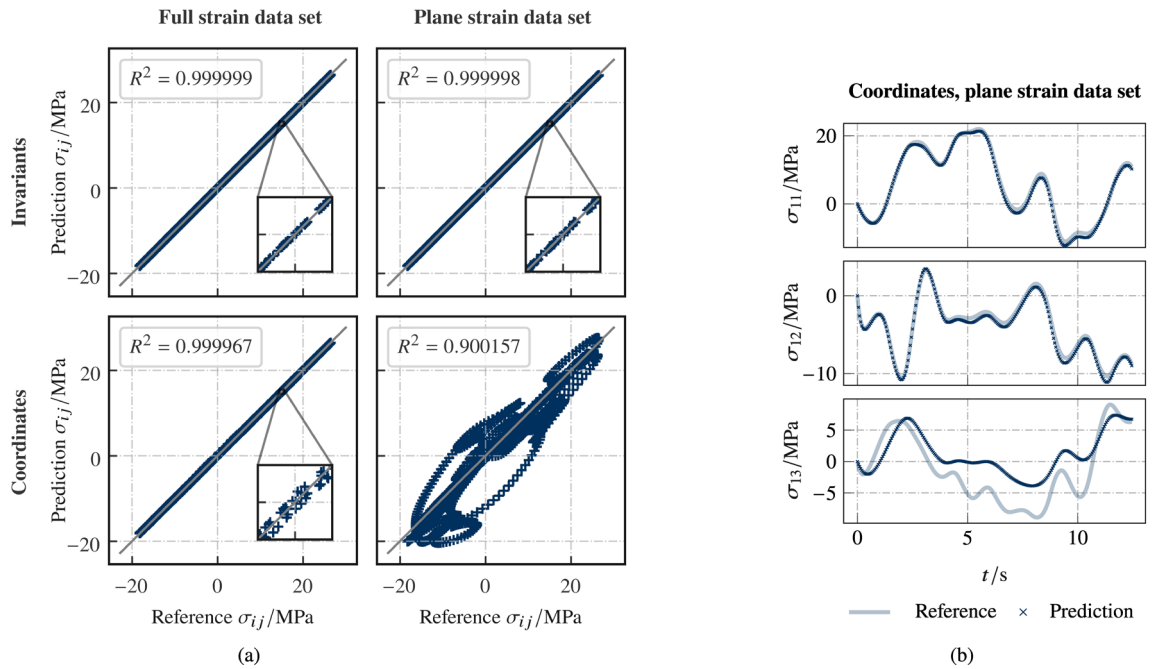


Fig. 8 Comparison of the formulation with invariants and coordinates as inputs to the networks: **a** shows scatter diagrams for the prediction results for both formulations using the data sets $\mathscr{D}^{\text{multiax}}_{5\times200}$ (full strain states) and $\mathscr{D}^{\text{plane}}_{5\times200}$ (plane strain states) with given internal variable. **b** shows the stress response $\boldsymbol{\sigma}(t)$ for the coordinate formulation with the data set $\mathscr{D}^{\text{plane}}_{5\times200}$ to emphasize the lacking extrapolation capability for the out-of-plane coordinates on the example of σ_{13}

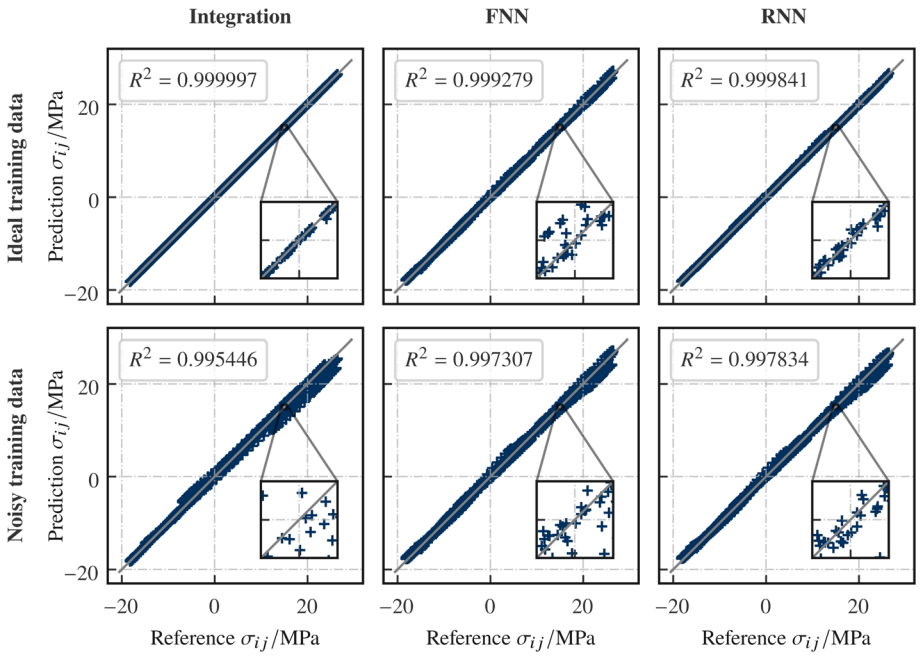


Fig. 9 Comparison of the training methods with scatter diagrams for the results of the stress prediction for the unseen test path. The models were trained with the data sets $\mathscr{D}^{\text{multiax}}_{1\times200}$ and $\tilde{\mathscr{D}}^{\text{multiax}}_{1\times200}$, respectively. For the noisy data, the reference stress refers to the underlying ground truth, i.e., the noiseless data

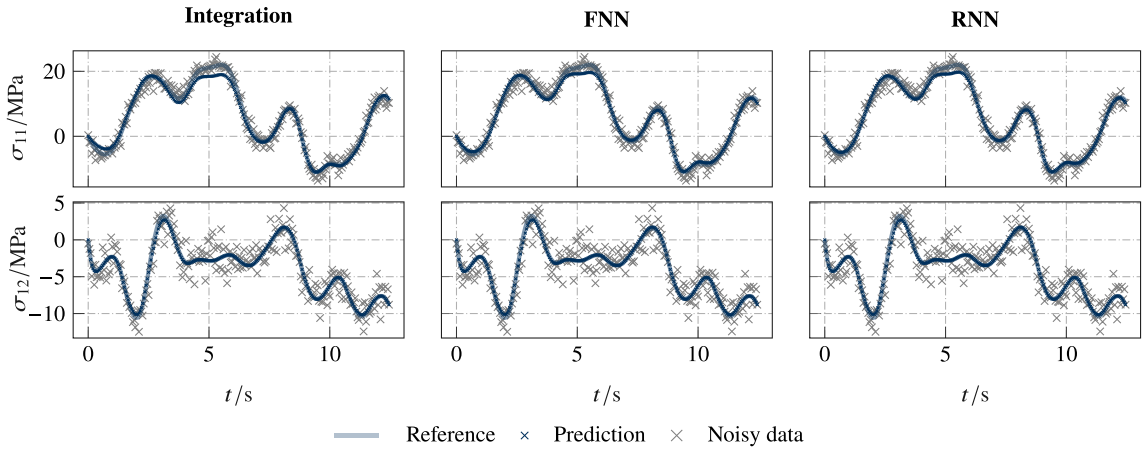


Fig. 10 Stress responses $\boldsymbol{\sigma}(t)$ of the trained models for the unseen test path $\boldsymbol{\varepsilon}(t)$ on the example of the coordinates σ_{11} and σ_{12}. The models were trained with the data set $\tilde{\mathscr{D}}^{\text{multiax}}_{1\times200}$. The grey marks indicate noisy stress data to illustrate the amount of noise in the training data set

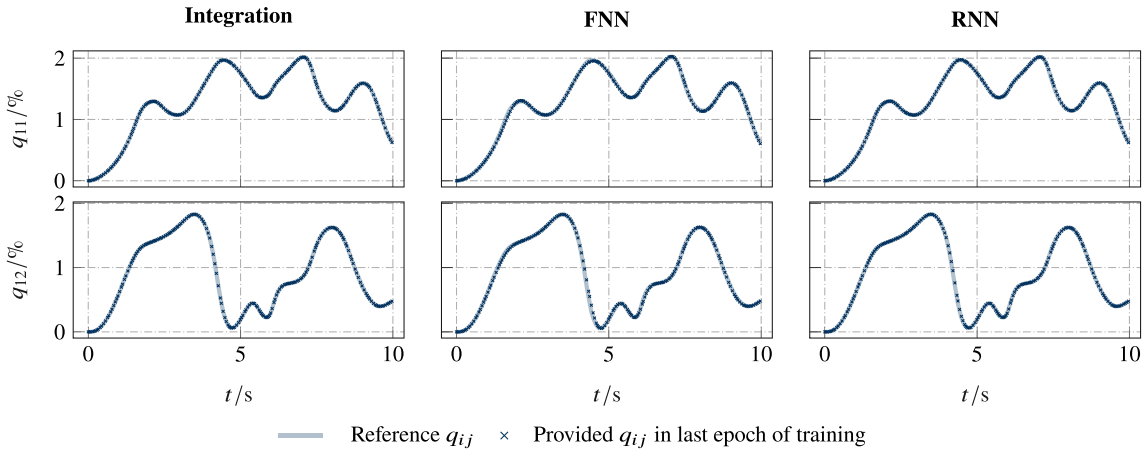


Fig. 11 Predicted temporal course of the internal variable $\mathbf{q}(t)$ of the training sequence for the three training methods using the example of the coordinates q_{11} and q_{12}. The models were trained with the data set $\mathscr{D}^{\text{multiax}}_{1\times200}$

Auxiliary feedforward network

Although this training method shows the least accurate predictions for the training with ideal data, the results are still sufficient for both ideal as well as noisy data. In addition to that, the computational time per iteration is very low. However, before the actual training starts, the auxiliary FNN must be pre-trained to a reasonable initial guess for $\mathbf{q}(t)$ to have good starting values for the weights and biases. Using randomly initialized parameters at the beginning of the actual training did not yield useful results in the numerical experiments shown here. We used $\mathbf{q}(t) = \boldsymbol{\varepsilon}(t)$ as initial guess to pre-train the FNN. Another major drawback of this method is the increasing demand of trainable parameters for a longer sequence or additional sequences in the training data set. Since a single FNN only predicts the temporal course of one sequence, every new sequence requires a new FNN, making the optimization more and more complex. For a single sequence of moderate length, however, the method can yield useful results, see Figs. 9 and 10.

Auxiliary recurrent network

The RNN as auxiliary network for the generation of the internal variable also provides precise results with both data sets. Compared to the integration method, the training is very fast and in contrast to the FNN it does not need a pre-training and the final value of the loss function is smaller, see Fig. 18 in the appendix.

Moreover, using a training data set with several sequences instead of only one sequence does not increase the number of trainable parameters, as the RNN implicitly learns how the internal variable evolves instead of memorizing its temporal course. Thus, we consider a test case which is more complex compared to the works [37, 39, 57], where only one path has been used for training. To do so, several sequences with multiaxial states are added to the training data set to obtain $\mathscr{D}^{\text{multiax}}_{100\times100}$ with 100 sequences. The prediction results

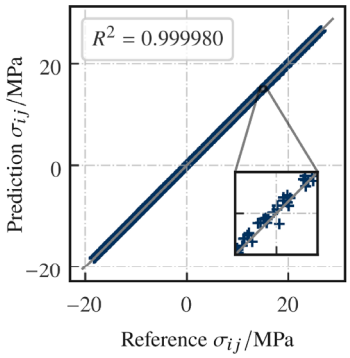


Fig. 12 Scatter diagram for the prediction results for the unseen test path using the RNN training method with the data set $\mathscr{D}^{\text{multiax}}_{100\times100}$, i.e., with 100 sequences

Table 4 Advantages and disadvantages of the different training methods

	Integration	FNN	RNN
Good results for ideal data	✓	✓	✓
Good results for noisy data	✓	✓	✓
Fast training	✗	✓	✓
No pre-training	✓	✗	✓
Data set with many sequences	(✓)	✗	✓

for the architecture trained with the set $\mathscr{D}^{\text{multiax}}_{100\times100}$ are shown in Fig. 12. As can be seen, the stress response for the unseen strain path is even more precise, while the required time per training iteration remains short. Although the data set contains 50-times the number of time steps, the time per iteration increases from 0.059s to only 0.112s. This shows that the RNN method allows to use multiple sequences efficiently without increasing amount of trainable parameters and within reasonable training time.

Summary

Each of the analyzed techniques exhibits certain advantages compared to the others. In order to provide a comprehensive summary of this comparison, the advantages and disadvantages of each approach are listed in Table 4. The integration training is very accurate. However, the biggest disadvantage of this method is the training time, which also affects the applicability for large data sets with many sequences, such that this combination is not practically manageable due to cost reasons. Using an FNN as an auxiliary network is extremely fast, but requires a pre-training of the FNN. It is also limited to one or very few sequences. In contrast, the RNN is fast, does not require pre-training, and can be applied to large data sets with multiple sequences without further adjustments. This feature is particularly relevant for real-world applications with experimental or homogenization data.

5.3 Training with biaxial data

So far, data sets were used that are difficult to obtain experimentally. In the following, data sets will be used for which experimental approaches are available. The models are trained with the RNN method. First, the data set $\mathscr{D}^{\text{biax}}_{1\times200}$ is used, which contains data from a virtual biaxial test in a single sequence with 200 time steps with ideal stress data. The model is again calibrated 25 times with the RNN method and the model with the lowest final loss value is kept and tested with the multiaxial test path. The results of the predictions for this test path are shown in Fig. 13. These show that the model is able to predict arbitrary multiaxial stress states despite the training data set being limited to biaxial data. The slight loss of accuracy compared to the multiaxial data set is due to the lacking coverage of the full invariant range in the training. However, with a suitable choice of load path for the training data, this problem can be minimized and the model can also be trained and applied with biaxial data using a suitable training method—in this case the RNN.

5.4 Training with uniaxial data

The previous paragraph shows that good prediction results can also be achieved with biaxial stress training data. We will proceed by assuming that only data from uniaxial stress testing is available. For this purpose, the data set $\mathscr{D}^{\text{uniax}}_{1\times200}$ is used, which contains synthetic data from a uniaxial test. This data set again contains 1 sequence with 200 time steps and ideal stress data. Compared to the data sets used so far, however, the maximum achievable stress and the rate of the stress are increased for this data set as described in Sect. 5.1.2 in order to compensate the unideal coverage of the invariant space. The model is calibrated 25 times using the RNN method. The prediction results of the best model are shown in the right plot in Fig. 13. This shows that the model can predict arbitrary multiaxial states without suffering from substantial limitations in accuracy, despite being restricted to uniaxial training data. However, it can also be seen that there are systematic deviations from the reference for large σ_{ij}, which indicate that the model extrapolates strongly in the invariant space within this region.[3] Nevertheless, it can be stated that by restricting the anisotropy class and incorporating basic physical principles, both the number of data points and the complexity of stress states in the training data can be greatly reduced, which allows to use experimental methods to generate training data for the presented model.

[3] With the uniaxial data set, only a small area in the invariant space can be scanned. In the case of isotropic elasticity, it is even only a single path, since ε_{11} and the transverse strains $\varepsilon_{22} = \varepsilon_{33}$ can be uniquely assigned to each uniaxial stress state [28, 42].

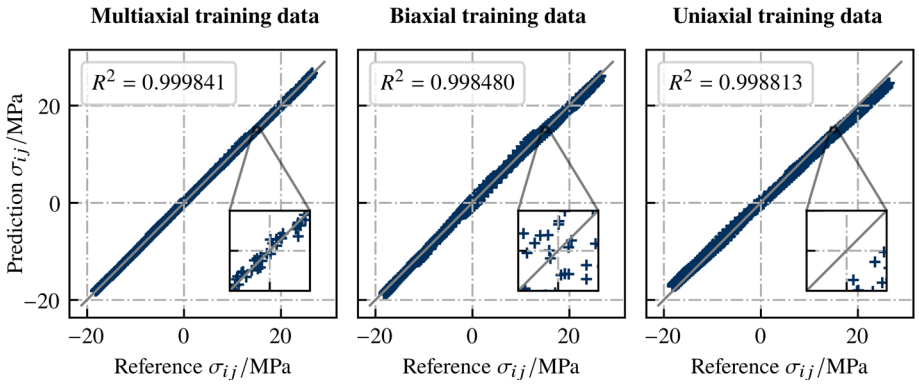


Fig. 13 Comparison of stress prediction quality for a given multiaxial test strain path using different training data sets and the RNN training method. All data sets consist of 1 sequence of 200 time steps and contain stress states with different complexity: (1) complete multiaxial states, strain driven (see Fig. 7) (2) biaxial stress states with different magnitude in the two non-zero directions, stress driven, and (3) uniaxial stress states, stress driven

6 Conclusions

This paper proposes a fast and widely applicable method to calibrate inelastic constitutive models without prior knowledge of the internal variables. This method is compared comprehensively with two existing methods. For this comparison, we propose and use a physics-augmented NN-based model for viscoelastic materials based on the concept of GSMs.

In the beginning of this work, the concept of GSMs is briefly described. Subsequently, the NN-based model is presented, which consists of two potentials: the free energy and the dissipation potential. These potentials are constrained in a physically meaningful way so that thermodynamic consistency is ensured in advance. The potentials can be expressed using either tensor coordinates or invariants of the tensor arguments. Following the presentation of the NN-based constitutive model, three training methods are introduced. These comprise a method that integrates entire sequences and two methods that use an additional NN, either an FNN or an RNN, to provide the internal variable. Based on these techniques, numerical experiments are conducted using synthetically generated data from classical constitutive models. First, it is assumed that the internal variable is present in the multiaxial and plane strain training data sets. Using these data sets, the invariant formulation is compared to the coordinate formulation. It is shown, that the invariant formulation requires less data, yields more accurate results and exhibits far better extrapolation capabilities when predicting stresses for arbitrary strain paths with plane strain training data. Henceforth, only the invariant formulation is used in the subsequent studies. In order to compare the three training methods, the internal variable is erased from the multiaxial data set and the performance of the methods is examined using only stress and strain data. It shows, that all three methods are able to provide models that yield accurate predictions for unseen data, even for a small data set with noisy stress data. However, only the proposed RNN model allows to achieve fast and accurate results with large data sets comprising many sequences and is therefore the method with the broadest applicability.

Finally, the applicability of the framework with an invariant-based NN model and the RNN training method for biaxial and uniaxial stress training data is examined. We show, that a complete 3D model can be calibrated, that is capable of predicting arbitrary multiaxial stress states, although only bi- or uniaxial stress data is used for training. Thus, this paper proposes a flexible constitutive model for viscoelastic materials and an efficient method for calibrating such models without prior knowledge of the internal variable. The presented training method exceeds the application possibilities of existing approaches and can form the basis for future extensions in the context of NN-based data driven constitutive modeling.

However, this study also has limitations. Specifically, the presented algorithms assume that constitutive behavior can be modeled with a single internal variable expressed by a symmetric second-order tensor. While this assumption is valid for the synthetically generated data used herein, it may not be sufficient to generalize for more complex behavior. In this case, the number of internal variables can be increased stepwise until the desired accuracy is achieved. Therefore, possible future studies include the implementation and testing of such algorithms, the application to elastoplastic materials [27, 32], extension to viscoelasticity at finite strains [39] as well as the use of experimental [38] or homogenization data [26].

A neural networks

This section describes the concept of input convex feed-forward neural networks, the normalization of inputs and outputs and gives an overview about the chosen hyperparameters of the networks, that are used in the numerical examples. In order to provide a compact explanation, some symbols are introduced: The network function is denoted as $\mathscr{N}(\mathfrak{i})$ and depends on an input vector $\mathfrak{i}$. The output $\boldsymbol{u}_l$ of the layer l is calculated using the activation function $\mathscr{A}_l$, the biases $\boldsymbol{b}_l$ and the weights $\boldsymbol{W}_l$ connecting this layer to the previous layer $l-1$. The following concepts are mainly based on Amos, Xu, and Kolter [49], with extensions to functions that are not

only convex with respect to the inputs i, but also with respect to some x in a network $\mathscr{N}(\mathrm{i}(x))$ by making the network non-decreasing in i [20, 50].

A.1 Input convex neural networks

A.1.1 Fully input convex neural networks

A fully input convex neural network (FICNN) is an FNN, that is convex in all of its arguments. There are several ways to construct such an FNN, but not all variants are equivalent. Consider for example a normal FNN with non-negative weights across all layers as well as convex and non-decreasing activation functions. Such an FNN is convex, but is also extremely restrictive. Therefore, the more flexible approach of Amos, Xu, and Kolter [49], shown in Fig. 14, is adopted. It is characterized by two different sets of weights: the weights in $\boldsymbol{W}_l^{\boldsymbol{u}}$ connect the layer $l-1$ with the layer l and the weights $\boldsymbol{W}_l^{\mathrm{i}}$ connect the input i with the layer l. The outputs of the layers $l \in (1, \dots, L)$ is now calculated as

$$\boldsymbol{u}_1 = \mathscr{A}_1\left(\boldsymbol{W}_1^{\mathrm{i}}\mathrm{i} + \boldsymbol{b}_1\right) \quad \text{for layer } l = 1 \text{ and} \tag{31}$$

$$\boldsymbol{u}_l = \mathscr{A}_l\left(\boldsymbol{W}_l^{\boldsymbol{u}}\boldsymbol{u}_{l-1} + \boldsymbol{W}_l^{\mathrm{i}}\mathrm{i} + \boldsymbol{b}_l\right) \; \forall l \in (2, \dots, L). \tag{32}$$

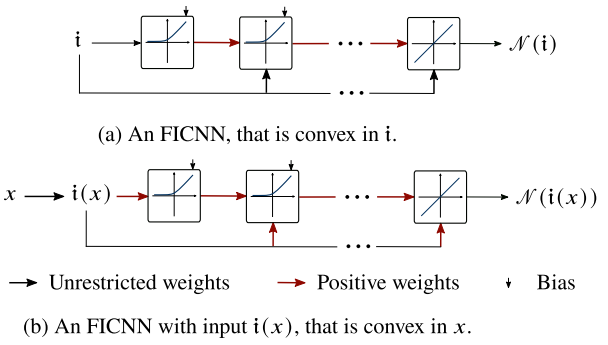
The network $\mathscr{N}(\mathrm{i}) = \boldsymbol{u}_L$ is convex if the following three conditions are fulfilled:

(i) all weights in $\boldsymbol{W}_l^{\boldsymbol{u}}$ are non-negative,
(ii) all $\mathscr{A}_l$ are convex, and
(iii) all $\mathscr{A}_l$ are non-decreasing.

This approach offers greater flexibility since the weights $\boldsymbol{W}_l^{\mathrm{i}}$ remain unconstrained as shown in Fig. 14a. A common choice for a convex and non-decreasing activation function is the softplus activation function, which is defined by

$$\mathscr{SP} : \mathbb{R} \to \mathbb{R}_{>0},\; x \mapsto \mathscr{SP}(x) = \ln(1 + \exp x). \tag{33}$$

So far, the network has been constructed in such a way that it is convex with respect to its direct inputs. However, this must not always be sufficient. For example, in the context of this work, networks are to be implemented, that are convex with respect to a tensor, although the network inputs are invariants. The arbitrariness of the $\boldsymbol{W}_l^{\mathrm{i}}$ follows from the fact that the second derivatives of $\boldsymbol{W}_l^{\mathrm{i}}\mathrm{i}$ with respect to i vanish and consequently do not have any influence on the convexity in i of the network function $\mathscr{N}(\mathrm{i})$. This changes if $\mathscr{N}$ is not supposed to be convex in its inputs, but in another variable x, which the inputs depend on. That is, $\mathscr{N}(\mathrm{i}(x))$ is supposed to be convex in x. Then, the second derivatives of $\boldsymbol{W}_l^{\mathrm{i}}\mathrm{i}(x)$ with respect to x do not vanish if $\mathrm{i}(x)$ is a nonlinear function of x.



(a) An FICNN, that is convex in i.

(b) An FICNN with input $\mathrm{i}(x)$, that is convex in x.

Fig. 14 Two types of FICNN architectures: the architecture in (**a**) is convex with respect to all entries in the input vector i with unrestricted weights in the passthrough layers. The architecture in (**b**) is constructed, such that it is convex *and* non-decreasing in $\mathrm{i}(x)$, making it convex in x. This affects the weights in passthrough layers, which may no longer take negative values. This significantly limits the effectiveness of the passthrough layers, but enforces convexity in x. The illustrations are based on Klein et al. [50]

Consequently, another restriction on the weights arises, see Fig. 14b: besides the conditions (i)–(iii), it must also hold that

(iv) all entries in $\mathrm{i}(x)$ are convex functions of x, and
(v) all weights in $\boldsymbol{W}_l^{\mathrm{i}}$ are non-negative [29].

A.1.2 Partially input convex neural networks

A partially input convex neural network (PICNN) is, in contrast to an FICNN, only convex in some of its inputs. Let $\mathscr{N}(\mathrm{i}^{\cup}, \mathrm{i}^{\frown})$ denote the network, that is convex in $\mathrm{i}^{\cup}$, but not necessarily convex in $\mathrm{i}^{\frown}$. The architecture of such a network is shown in Fig. 15. The figure shows the two different paths corresponding to the convex (bottom) and non-convex part (top). The non-convex part with the layer outputs $\boldsymbol{v}_l$ is a regular FNN, i.e., each layer is connected to only the previous layer of the non-convex path. A layer in the convex path with layer outputs $\boldsymbol{u}_l$, however, depends on not only the previous layer of the convex path, but also on the output of previous layer in the non-convex path and the convex inputs. The outputs of the layers in the non-convex path are defined as

$$\begin{aligned}\boldsymbol{v}_1 &= \mathscr{A}_1^{\frown}\left(\boldsymbol{W}_1^{\boldsymbol{vv}}\mathrm{i}^{\frown} + \boldsymbol{b}_1^{\boldsymbol{vv}}\right) \quad \text{for the first layer } l = 1 \text{ and}\\ \boldsymbol{v}_l &= \mathscr{A}_l^{\frown}\left(\boldsymbol{W}_l^{\boldsymbol{vv}}\boldsymbol{v}_{l-1} + \boldsymbol{b}_l^{\boldsymbol{vv}}\right) \; \forall l \in (2, \dots, L-1)\end{aligned} \tag{34}$$

and for the convex path as

$$\begin{aligned}\boldsymbol{u}_1 = \mathscr{A}_1^{\cup}\Big(\boldsymbol{W}_1^{\boldsymbol{u}\mathrm{i}^{\cup}}\Big[\mathrm{i}^{\cup} \odot \mathscr{A}_1^{\mathrm{i}^{\cup}\boldsymbol{v}}(\tilde{\boldsymbol{W}}_1^{\mathrm{i}^{\cup}\boldsymbol{v}}\mathrm{i}^{\frown} + \tilde{\boldsymbol{b}}_1^{\mathrm{i}^{\cup}\boldsymbol{v}})\Big] +\\ \boldsymbol{W}_1^{\boldsymbol{uv}}\mathrm{i}^{\frown} + \boldsymbol{b}_1^{\boldsymbol{u}}\Big) \quad \text{for the first layer } l = 1 \text{ and}\end{aligned} \tag{35}$$

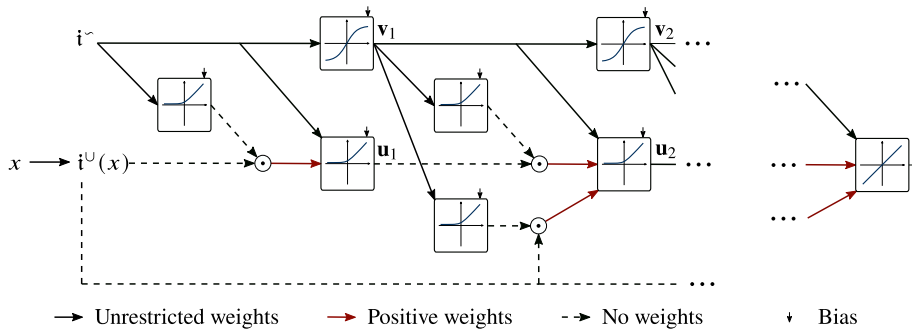


Fig. 15 The PICNN architecture is both convex and non-decreasing in $\mathrm{i}^{\cup}$, but allows arbitrary functional relations in $\mathrm{i}^{\frown}$. For this purpose, the PICNN is divided into two paths, one for the convex part ($\boldsymbol{u}_l$) and one for the non-convex part ($\boldsymbol{v}_l$). The non-convex path $\boldsymbol{v}_l$ is independent of the convex path $\boldsymbol{u}_l$, while the layers in the convex path take into account both the output of the non-convex path as well as multiplications between $\boldsymbol{v}_l$, $\boldsymbol{u}_l$ and the convex input $\mathrm{i}^{\cup}$. The final output is the last layer in the convex path. The illustration is based on Klein et al. [50]

$$\begin{aligned}\boldsymbol{u}_l = \mathscr{A}_l^{\cup}\Big(& \boldsymbol{W}_l^{\boldsymbol{uu}}\left[\boldsymbol{u}_{l-1} \odot \mathscr{A}_l^{\boldsymbol{uv}}(\tilde{\boldsymbol{W}}_l^{\boldsymbol{vv}}\boldsymbol{v}_{l-1} + \tilde{\boldsymbol{b}}_l^{\boldsymbol{uv}})\right] + \\ & \boldsymbol{W}_l^{\boldsymbol{u}\mathrm{i}^{\cup}}\left[\ \mathrm{i}^{\cup} \odot \mathscr{A}_l^{\mathrm{i}^{\cup}\boldsymbol{v}}(\tilde{\boldsymbol{W}}_l^{\mathrm{i}^{\cup}\boldsymbol{v}}\boldsymbol{v}_{l-1} + \tilde{\boldsymbol{b}}_l^{\mathrm{i}^{\cup}\boldsymbol{v}})\right] + \\ & \boldsymbol{W}_l^{\boldsymbol{uv}}\boldsymbol{v}_{l-1} + \boldsymbol{b}_l^{\boldsymbol{u}}\Big) \quad \forall l \in (2, \ldots, L) \quad . \end{aligned} \tag{36}$$

The final output of the network is the last layer of the convex path, i.e., $\mathscr{N}(\mathrm{i}^{\cup}, \mathrm{i}^{\frown}) = \boldsymbol{u}_L$. Instead of two sets of weights, a PICNN comprises six different sets of weights ($\boldsymbol{W}_l^{\boldsymbol{vv}}$, $\boldsymbol{W}_l^{\boldsymbol{uu}}$, $\boldsymbol{W}_l^{\boldsymbol{u}\mathrm{i}^{\cup}}$, $\boldsymbol{W}_l^{\boldsymbol{uv}}$, $\tilde{\boldsymbol{W}}_l^{\boldsymbol{vv}}$, $\tilde{\boldsymbol{W}}_l^{\mathrm{i}^{\cup}\boldsymbol{v}}$) to take into account possible products between the two paths and the convex input. The network is convex in $\mathrm{i}^{\cup}$ if the following four conditions are fulfilled:

(i) all weights in $\boldsymbol{W}_l^{\boldsymbol{uu}}$ are non-negative,
(ii) all activations $\mathscr{A}_l^{\cup}$ are convex,
(iii) all activations $\mathscr{A}_l^{\cup}$ are non-decreasing, and
(iv) all activations $\mathscr{A}_l^{\boldsymbol{uv}}$ map to non-negative values.

All other weights may as well take negative values and the activations $\mathscr{A}_l^{\frown}$ and $\mathscr{A}_l^{\mathrm{i}^{\cup}\boldsymbol{v}}$ can be chosen arbitrarily. A valid choice for $\mathscr{A}_l^{\cup}$ is, like for the FICNN, the softplus function $\mathscr{SP}$, which additionally is non-negative and thus can be used as $\mathscr{A}_l^{\boldsymbol{uv}}$ as well. The mentioned conditions yield a network $\mathscr{N}(\mathrm{i}^{\cup}, \mathrm{i}^{\frown})$, that is convex in all entries of $\mathrm{i}^{\cup}$. As already discussed for the FICNN, this is not always sufficient. If a network $\mathscr{N}(\mathrm{i}^{\cup}(x), \mathrm{i}^{\frown})$ is supposed to be convex in x, it is not the first function acting on x, which leads to further restrictions on the weights and activations. $\mathscr{N}(\mathrm{i}^{\cup}(x), \mathrm{i}^{\frown})$ is convex in x, if in addition to (i)–(iv) it holds:

(v) all entries of $\mathrm{i}^{\cup}$ are convex functions of x,
(vi) all weights in $\boldsymbol{W}_l^{\boldsymbol{u}\mathrm{i}^{\cup}}$ are non-negative, and
(vii) all $\mathscr{A}_l^{\mathrm{i}^{\cup}\boldsymbol{v}}$ map to non-negative values.

These additional restrictions are only necessary if the entries of $\mathrm{i}^{\cup}$ are non-linear function in x, i.e., the second derivatives do not vanish. Otherwise, the standard architecture is also valid.

A.2 Normalization

The training process of neural networks is typically more stable and efficient if the inputs and outputs of the network are values of magnitude 1. In particular, since the free energy and the dissipation potential take on large numerical values in the range $> 10^6$, normalization is essential for a successful training. Normalization of a known quantity $(\circ)$ to the range $(-1, 1) \ni (\tilde{\circ})$ can be carried out with

$$(\tilde{\circ}) = \frac{(\circ) - m_{(\circ)}}{s_{(\circ)}} \quad \text{with} \tag{37}$$

$$s_{(\circ)} = \frac{1}{2}((\circ)_{\max} - (\circ)_{\min}) \quad \text{and} \tag{38}$$

$$m_{(\circ)} = \frac{1}{2}((\circ)_{\max} + (\circ)_{\min}) \quad , \tag{39}$$

where $(\tilde{\circ})$ represents the normalized quantity and $(\circ)_{\max}$ and $(\circ)_{\min}$ are the maximum and minimum value of $(\circ)$ across the whole training data set. For the problem described here, the inputs $\boldsymbol{\varepsilon}, \mathbf{q}, \dot{\mathbf{q}}, t$ and Δt or the respective invariants $\mathcal{I}^{\psi^{\mathrm{eq}}}$, $\mathcal{I}^{\psi^{\mathrm{ov}}}$ and $\mathcal{I}^{\phi}$ have to be normalized, as well as the outputs ψ^{eq}, ψ^{ov} and ϕ. The difficulty for the proposed model arises from the fact that neither $\mathbf{q}$ and $\dot{\mathbf{q}}$ as inputs nor ψ^{eq}, ψ^{ov} or ϕ as outputs are known before the training, but only $\boldsymbol{\varepsilon}(t)$, $\boldsymbol{\sigma}(t)$ and t are known in advance. For this reason, the order of magnitude of the expected values for the unknown variables is estimated based on the available variables. Therefore, the normalization parameters s and m are determined on the basis of the following assumptions: the internal variable and the variable $\mathbf{p} = \boldsymbol{\varepsilon} - \mathbf{q}$ are normalized with the values of $\boldsymbol{\varepsilon}$, i.e.,

$s_{\mathbf{q}} = s_{\mathbf{p}} = s_{\boldsymbol{\varepsilon}}$, while the rate $\dot{\mathbf{q}}$ is normalized with $\dot{\boldsymbol{\varepsilon}}$. The normalization parameters for the potentials are defined to be $s_{\psi^{\mathrm{eq}}} = s_{\psi^{\mathrm{ov}}} = m_{\psi^{\mathrm{eq}}} = m_{\psi^{\mathrm{ov}}} = s_{\boldsymbol{\varepsilon}} \cdot s_{\boldsymbol{\sigma}}$, where the choice of the ms exploits the fact that ψ^{eq}, ψ^{ov}, $\phi \geq 0$. Using these normalization parameters, the network itself maps only from values of magnitudes around 1 to values of magnitude 1, which enables an efficient training. Note, that due to the linear nature of the transformation Eq. (37), this type of normalization does not effect the convexity properties of the resulting network function.

A.3 Architecture details

A.3.1 Potentials

The free energy and the dissipation potential are implemented as two FICNNs and a PICNN. The details about the size of the networks, i.e., number of hidden layers, number of neurons or used activation functions can be found in Table 5.

A.3.2 Auxiliary networks

Integration No auxiliary network is necessary for this training method.
FNN The auxiliary network for the internal variable is an ordinary FNN with a single input (the time t), 2 hidden layers with 50 neurons each, tanh activations in the hidden layers and 6 output neurons with linear activation.
RNN The auxiliary network comprises an RNN cell and a subsequent FNN. The RNN cell is an LSTM cell with 13 inputs (6 for the strain tensor $\boldsymbol{\varepsilon}$, 6 for the stress tensor $\boldsymbol{\sigma}$ and one for the time increment Δt), 50 entries in the hidden state and tanh activations. The FNN takes the hidden state as input, has no hidden layers, i.e., it consists of only the input and output layer, and 6 has output neurons with linear activations.

B Convexity of the invariant basis

To enforce convexity of a network with respect to a symmetric second order tensor $\mathbf{S} \in \mathcal{S}ym_2$, the invariant basis $\mathcal{I} = (\operatorname{tr}\mathbf{S}, \operatorname{tr}\mathbf{S}^2, \operatorname{tr}\mathbf{S}^4)$ is used throughout this work. To motivate this choice, the convexity of

(i) $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}$,
(ii) $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^2$,
(iii) $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^3$ and
(iv) $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^4$

is examined. Two equivalent definitions of convexity are used here. A function $f(\mathbf{S})$ is convex, if

$$f(\mathbf{A} + \lambda[\mathbf{B} - \mathbf{A}]) \leq f(\mathbf{A}) + \lambda[f(\mathbf{B}) - f(\mathbf{A})] \quad \forall \mathbf{A}, \mathbf{B} \in \mathcal{S}ym_2,\ \forall \lambda \in [0,1] \quad . \tag{40}$$

or, under the assumption, that f is twice continuously differentiable,

$$\mathbf{A} : \frac{\partial^2 f}{\partial \mathbf{S} \partial \mathbf{S}} : \mathbf{A} \geq 0 \,\forall \mathbf{A} \in \mathcal{S}ym_2. \tag{41}$$

Note that for twice differentiable f, Eq. (41) follows from Eq. (40) and vice versa.

(i) From Eq. (40) follows directly

$$\operatorname{tr}(\mathbf{A} + \lambda[\mathbf{B} - \mathbf{A}]) = \operatorname{tr}(\mathbf{A}) + \lambda[\operatorname{tr}(\mathbf{B}) - \operatorname{tr}(\mathbf{A})]. \tag{42}$$

As a linear function, the invariant $\operatorname{tr}(\mathbf{S})$ is thus (weakly) convex.

(ii) Applying Eq. (40) to $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^2$ yields

$$\begin{aligned}\operatorname{tr}\left((\mathbf{A} + \lambda[\mathbf{B} - \mathbf{A}])^2\right) = \operatorname{tr}\big(&\mathbf{A}^2 + \lambda\mathbf{A}\cdot[\mathbf{B} - \mathbf{A}] + \\ &\lambda[\mathbf{B} - \mathbf{A}]\cdot\mathbf{A} + \\ &\lambda^2\left[\mathbf{B}^2 - \mathbf{A}\cdot\mathbf{B} - \mathbf{B}\cdot\mathbf{A} + \mathbf{A}^2\right]\big) .\end{aligned} \tag{43}$$

With $\operatorname{tr}(\mathbf{A}\cdot\mathbf{B}) = \operatorname{tr}(\mathbf{B}\cdot\mathbf{A})$, this expression simplifies to

$$\begin{aligned}&\operatorname{tr}(\mathbf{A}^2) + 2\lambda\operatorname{tr}(\mathbf{A}\cdot\mathbf{B} - \mathbf{A}^2) + \\ &\quad\lambda^2\operatorname{tr}(\mathbf{B} - 2\mathbf{A}\cdot\mathbf{B} + \mathbf{A}^2) \leq \operatorname{tr}(\mathbf{A}^2) + \\ &\qquad\qquad\lambda\left[\operatorname{tr}(\mathbf{B}^2) - \operatorname{tr}(\mathbf{A}^2)\right] \\ &\lambda^2\operatorname{tr}(\mathbf{B}^2 - 2\mathbf{A}\cdot\mathbf{B} + \mathbf{A}^2) \leq \lambda\operatorname{tr}(\mathbf{B}^2 - 2\mathbf{A}\cdot\mathbf{B} + \mathbf{A}^2) \\ &\qquad\lambda^2\operatorname{tr}((\mathbf{B} - \mathbf{A})^2) \leq \lambda\operatorname{tr}((\mathbf{B} - \mathbf{A})^2) \\ &\qquad\qquad\lambda \leq 1 \quad .\end{aligned} \tag{44}$$

(iii) For $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^3$, the tensors $\mathbf{A} = \mathbf{0}$ and $\mathbf{B} = -\mathbf{1}$ are considered. With Eq. (40) it follows

$$\begin{aligned}\operatorname{tr}\left((\mathbf{0} + \lambda[-\mathbf{1} - \mathbf{0}])^3\right) = -3\lambda^3 &\leq \operatorname{tr}(\mathbf{0}^3) + \\ &\lambda\left[\operatorname{tr}((-\mathbf{1})^3) - \operatorname{tr}(\mathbf{0}^3)\right] \\ -3\lambda^3 &\leq -3\lambda \\ \lambda^3 &\geq \lambda,\end{aligned} \tag{45}$$

which is a contradiction for $\lambda \in [0, 1]$. The invariant $\operatorname{tr}\mathbf{S}^3$ is thus not convex in $\mathbf{S}$.

(iv) To show the convexity of $f(\mathbf{S}) = \operatorname{tr}\mathbf{S}^4$, Eq. (41) is used. Evaluating Eq. (41) and introducing the abbreviation $A_{ik}S_{kj} = r_{ij}$ yields

Table 5 Hyperparameters, that were used for the NNs of the potentials

Network Input type	$\tilde{\psi}^{\mathrm{eq}}$ Invariants	$\tilde{\psi}^{\mathrm{eq}}$ Coordinates	$\tilde{\psi}^{\mathrm{ov}}$ Invariants	$\tilde{\psi}^{\mathrm{ov}}$ Coordinates	$\tilde{\phi}$ Invariants	$\tilde{\phi}$ Coordinates
Architecture	FICNN	FICNN	FICNN	FICNN	PICNN	PICNN
Inputs (convex path)	3	6	3	6	3	6
Inputs (non-convex path)	–	–	–	–	3	6
Hidden layers	1	1	1	1	1	1
Neurons in hidden layer (convex path)	10	20	10	20	10	20
Neurons in hidden layer (non-convex path)	–	–	–	–	10	20
Outputs	1	1	1	1	1	1
Activation (convex path)	$\mathscr{SP}$	$\mathscr{SP}$	$\mathscr{SP}$	$\mathscr{SP}$	–	–
Activation (non-convex path)	–	–	–	–	tanh	tanh

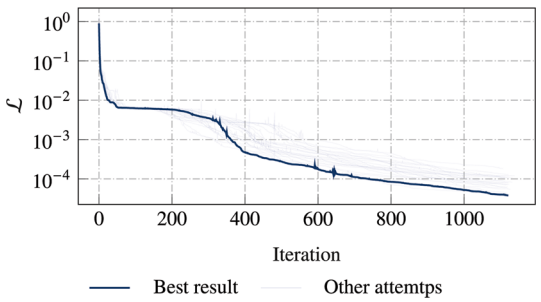


Fig. 16 Value of the loss function over iterations for the integration training method. The model was trained 25 times using the optimizer SLSQP, the best attempt is highlighted as thick blue curve

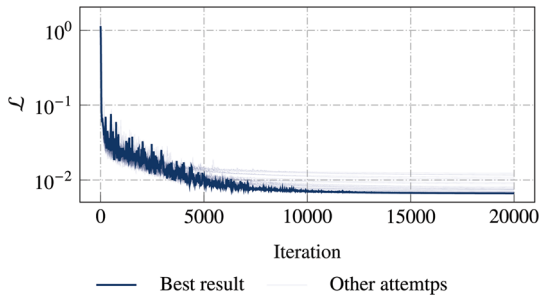


Fig. 17 Value of the loss function over iterations for the training method with an FNN as auxiliary network for the internal variable. The model was trained 25 times using the optimizer Adam

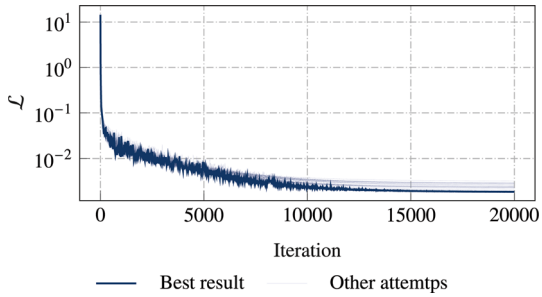


Fig. 18 Value of the loss function over iterations for the training method with an RNN as auxiliary network for the internal variable. The model was trained 25 times using the optimizer Adam

$$\frac{\partial f}{\partial S_{ij}\partial S_{kl}} A_{ij} A_{kl} = 4\left[2 r_{mn} r_{mn} + r_{mn} r_{nm}\right]. \tag{46}$$

Decomposing $\boldsymbol{r}$ into the symmetric part $s_{ij} = s_{ji}$ and antisymmetric part $a_{ij} = -a_{ji}$, such that $r_{mn} = s_{mn} + a_{mn}$ and using $s_{mn} a_{mn} = 0$ finally leads to

$$\frac{\partial f}{\partial S_{ij}\partial S_{kl}} A_{ij} A_{kl} = 4\left[3 s_{mn} s_{mn} + a_{kn} a_{kn}\right] \geq 0. \tag{47}$$

Consequently, $\mathcal{I} = (\operatorname{tr}\mathbf{S}, \operatorname{tr}\mathbf{S}^2, \operatorname{tr}\mathbf{S}^4)$ forms a complete and convex set of invariants.

C Loss over iterations

In 5.2.2, the three presented training methods are compared. Therefor, the data sets $\mathscr{D}^{\mathrm{multiax}}_{1\times 200}$ and $\tilde{\mathscr{D}}^{\mathrm{multiax}}_{1\times 200}$ are used for training. The value of the respective loss functions for each method over the number of training iterations using the data set with ideal stress data $\mathscr{D}^{\mathrm{multiax}}_{1\times 200}$ is shown in this section. Recall, that every training method was used 25 times. To show how reliable each method is, all 25 curves with different random initializations are shown and the best run is highlighted with a thicker curve.

Acknowledgements All presented computations were performed on a HPC-Cluster at the Center for Information Services and High Performance Computing (ZIH) at TU Dresden. The authors thus thank the ZIH for generous allocations of computer time. MR and MK thank the German Research Foundation (DFG) for the support within the Research Training Group GRK2868 D^3—Project Number 493401063. The work of KK was supported by a postdoc fellowship of the German Academic Exchange Service (DAAD) and by the Graduate Academy (GA) of TU Dresden. All supports are gratefully acknowledged.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Peter H (2000) Continuum mechanics and theory of materials. Springer, Berlin. https://doi.org/10.1007/978-3-662-04775-0
2. de Souza NEA, Peri D, Owen DRJ (2008) Computational methods for plasticity. Wiley, Chichester. https://doi.org/10.1002/9780470694626
3. Bock Frederic E et al (2019) A review of the application of machine learning and data mining approaches in continuum materials mechanics. Front Mater 6:110. https://doi.org/10.3389/fmats.2019.00110
4. Dornheim J et al. (2023) Neural networks for constitutive modeling: from universal function approximators to advanced models and the integration of physics. In: Archives of computational methods in engineering. https://doi.org/10.1007/s11831-023-10009-y
5. Ghaboussi J, Garrett JH, Wu X (1991) Knowledge-based modeling of material behavior with neural networks. J Eng Mech 117(1):132–153. https://doi.org/10.1061/(ASCE)0733-9399(1991)117:1(132)
6. Furukawa T, Yagawa G (1998) Implicit constitutive modelling for viscoplasticity using neural networks. Int J Numer Methods Eng 43(2):195–219
7. Hambli R, Katerchi H, Benhamou C-L (2011) Multiscale methodology for bone remodelling simulation using coupled finite element and neural network computation. Biomech Model Mechanobiol 10(1):133–145. https://doi.org/10.1007/s10237-010-0222-x
8. Yao CG et al (2014) Artificial neural network modelling to predict hot deformation behaviour of as HIPed FGH4169 superalloy. Mater Sci Technol 30(10):1170–1176. https://doi.org/10.1179/1743284713Y.0000000411
9. Xiaoxin L et al (2019) A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites. Comput Mech 64(2):307–321. https://doi.org/10.1007/s00466-018-1643-0
10. Heider Y, Wang K, Sun WC (2020) SO(3)- invariance of informed-graph-based deepneuralnetwork for anisotropic elastoplastic materials. Comput Methods Appl Mech Eng 363:112875. https://doi.org/10.1016/j.cma.2020.112875
11. Fuchs A et al (2021) DNN2: a hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions. Comput Struct 249:106505. https://doi.org/10.1016/j.compstruc.2021.106505
12. Ghavamian F, Simone A (2019) Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. Comput Methods Appl Mech Eng 357:112594. https://doi.org/10.1016/j.cma.2019.112594
13. Mei H et al (2020) Study on constitutive relation of nickel-base superalloy inconel 718 based on long short term memory recurrent neural network. Metals 10(12):1588. https://doi.org/10.3390/met10121588
14. Bonatti C, Mohr D (2022) On the importance of self-consistency in recurrent neural network models representing elasto-plastic solids. J Mech Phys Solids 158:104697. https://doi.org/10.1016/j.jmps.2021.104697
15. Rumelhart DE, Hinton GE, Williams RJ (1988) Learning internal representations by error propagation. In: Readings in cognitive science. Elsevier, pp 399–421. https://doi.org/10.1016/B978-1-4832-1446-7.50035-2
16. Leygue A et al (2018) Data-based derivation of material response. Comput Methods Appl Mech Eng 331:184–196. https://doi.org/10.1016/j.cma.2017.11.013
17. Rosenkranz M et al (2023) A comparative study on different neural network architectures to model inelasticity. Int J Numer Meth Eng 124(21):4802–4840. https://doi.org/10.1002/nme.7319
18. Hochreiter S, Schmidhuber J (1997) Long short- term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
19. Cho K et al. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078 [cs.CL]
20. Linden L et al (2023) Neural networks meet hyperelasticity: a guide to enforcing physics. J Mech Phys Solids 179:105363. https://doi.org/10.1016/j.jmps.2023.105363
21. Masi F et al (2021) Thermodynamics-based artificial neural networks for constitutive modeling. J Mech Phys Solids 147:104277. https://doi.org/10.1016/j.jmps.2020.104277
22. Raissi M, Perdikaris P, Karniadakis GE (2019) Physicsinformed neural networks: adeep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045
23. Henkes A, Wessels H, Mahnken R (2022) Physics informed neural networks for continuum micromechanics. Comput Methods Appl Mech Eng 393:114790. https://doi.org/10.1016/j.cma.2022.114790
24. As'ad F, Avery P, Farhat C (2022) A mechanics-informed artificial neural network approach in data-driven constitutive modeling. Int J Numer Meth Eng 123(12):2738–2759. https://doi.org/10.1002/nme.6957
25. Klein DK et al (2022) Finite electro-elasticity with physics-augmented neural networks. Comput Methods Appl Mech Eng 400:115501. https://doi.org/10.1016/j.cma.2022.115501
26. Kalina KA et al (2023) FEANN: an efficient data- driven multiscale approach based on physics-constrained neural networks and automated data mining. Comput Mech 71:827–851. https://doi.org/10.1016/j.cma.2022.115501
27. Vlassis NN, Sun WC (2021) Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. Comput Methods Appl Mech Eng 377:113695. https://doi.org/10.1016/j.cma.2021.113695
28. Kalina KA et al (2022) Automated constitutive modeling of isotropic hyperelasticity based on artificial neural networks. Comput Mech 69(1):213–232. https://doi.org/10.1007/s00466-021-02090-6
29. Klein DK et al (2022) Polyconvex anisotropic hyperelasticity with neural networks. J Mech Phys Solids 159:104703. https://doi.org/10.1016/j.jmps.2021.104703
30. Eivazi H et al (2023) FE2 computations with deep neural networks: algorithmic structure, data generation, and implementation. Mathematical and Computational Applications 28(4):91. https://doi.org/10.3390/mca28040091

31. Vlassis NN, Sun W (2021) Component- based machine learning paradigm for discovering rate-dependent and pressure-sensitive level-set plasticity models. J Appl Mech 10(1115/1):4052684
32. Meyer KA, Ekre F (2023) Thermodynamically consistent neural network plasticity modeling and discovery of evolution laws. J Mech Phys Solids 180:105416. https://doi.org/10.1016/j.jmps.2023.105416
33. Fuhg JN et al (2023) Modular machine learning- based elasto-plasticity: generalization in the context of limited data. Comput Methods Appl Mech Eng 407:115930. https://doi.org/10.1016/j.cma.2023.115930
34. Rezaei S, Moeineddin A, Harandi A (2024) Learning solutions of thermodynamics-based nonlinear constitutive material models using physics-informed neural networks. Comput Mech. https://doi.org/10.1007/s00466-023-02435-3
35. Fuhg JN, Jones RE, Bouklas N (2023) Extreme sparsification of physics-augmented neural networks for interpretable model discovery in mechanics. arXiv: 2310.03652 [cs.CE]
36. Benady A, Baranger E, Chamoin L (2024) Unsupervised learning of history-dependent constitutive material laws with thermodynamically consistent neural networks in the modified constitutive relation error framework. Working paper or preprint. https://hal.science/hal-04368755
37. Taç V et al (2023) Data-driven anisotropic finite viscoelasticity using neural ordinary differential equations. Comput Methods Appl Mech Eng 411:116046. https://doi.org/10.1016/j.cma.2023.116046
38. Abdolazizi KP, Linka K, Cyron CJ (2023) Viscoelastic constitutive artificial neural networks (vCANNs)—a framework for data-driven an isotropic nonlinear finite viscoelasticity. J Comput Phys 499:112704. https://doi.org/10.1016/j.jcp.2023.112704
39. As'ad F, Farhat C (2023) A mechanics-informed neural network framework for data-driven nonlinear viscoelasticity. In: AIAA SCITECH 2023 Forum. https://doi.org/10.2514/6.2023-0949
40. Upadhyay K et al. (2023) Physics-informed data-driven discovery of constitutive models with application to strain-rate-sensitive soft materials. arXiv: 2304.13897 [cs.CE]
41. Cai C et al (2023) Equivariant geometric learning for digital rock physics: estimating formation factor and effective permeability tensors from Morse graph. Int J Multiscale Comput Eng 21(5):1–24. https://doi.org/10.1615/IntJMultCompEng.2022042266
42. Linka K et al (2021) Constitutive artificial neural networks: a fast and general approach to predictive data driven constitutive modeling by deep learning. J Comput Phys 429:110010. https://doi.org/10.1016/j.jcp.2020.110010
43. Fuhg JN, Bouklas N, Jones RE (2022) Learning hyperelastic anisotropy from data via a tensor basis neural network. J Mech Phys Solids 168:105022. https://doi.org/10.1016/j.jmps.2022.105022
44. Tac V, Costabal FS, Tepole AB (2022) Data-driven tissue mechanics with polyconvex neural ordinary differential equations. Comput Methods Appl Mech Eng 398:115248. https://doi.org/10.1016/j.cma.2022.115248
45. Vlassis NN, Ma R, Sun WC (2020) Geometric deep learning for computational mechanics Part I: anisotropic hyperelasticity. Comput Methods Appl Mech Eng 371:113299. https://doi.org/10.1016/j.cma.2020.113299
46. Baydin AG et al. (2015) Automatic differentiation in machine learning: a survey. arXiv: 1502.05767 [cs.SC]
47. Daw A et al. (2021) Physics-guided neural networks (PGNN): an application in lake temperature modeling. arXiv: 1710.11431 [cs.LG]
48. Thakolkaran P et al (2022) NN-EUCLID: deep-learning hyperelasticity without stress data. J Mech Phys Solids 169:105076. https://doi.org/10.1016/j.jmps.2022.105076
49. Amos B, Xu L, Zico KJ (2016) Input convex neural networks. arXiv: 1609.07152 [cs.LG]
50. Klein Dominik K et al (2023) Parametrized polyconvex hyperelasticity with physics-augmented neural networks. Data-Centric Eng 4:e25. https://doi.org/10.1017/dce.2023.21
51. He X, Chen J-S (2022) Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials. IComput Methods Appl Mech Eng 402:115348. https://doi.org/10.1016/j.cma.2022.115348
52. Weber P, Wagner W, Freitag S (2023) Physically enhanced training for modeling rate- independent plasticity with feedforward neural networks. Comput Mech. https://doi.org/10.1007/s00466-023-02316-9
53. Malik A et al (2021) A hybrid approach employing neural networks to simulate the elasto-plastic deformation behavior of 3d-foam structures. Adv Eng Mater 24(2):2100641. https://doi.org/10.1002/adem.202100641
54. Masi F, Stefanou I (2022) Multiscale modeling of inelastic materials with thermodynamicsbased artificial neural networks (TANN). Comput Methods Appl Mech Eng 398:115190. https://doi.org/10.1016/j.cma.2022.115190
55. Vlassis NN, Sun WC (2023) Geometric deep learning for computational mechanics part II: graph embedding for interpretable multiscale plasticity. Comput Methods Appl Mech Eng 404:115768. https://doi.org/10.1016/j.cma.2022.115768
56. Huang S et al (2022) Variational onsager neural networks (VONNs): a thermodynamics-based variational learning strategy for non-equilibrium PDEs. J Mech Phys Solids 163:104856. https://doi.org/10.1016/j.jmps.2022.104856
57. Holthusen H et al. (2023) Theory and implementation of inelastic constitutive artificial neural networks. arXiv: 2311.06380 [cs.LG]
58. Miehe C, Kiefer B, Rosato D (2011) An incremental variational formulation of dissipative magnetostriction at the macroscopic continuum level. Int J Solids Struct 48(13):1846–1866. https://doi.org/10.1016/j.ijsolstr.2011.02.011
59. Miehe C, Schotte J, Lambrecht M (2002) Homogenization of inelastic solid materials at ÿnite strains based on incremental minimization principles. Application to the texture analysis of polycrystals. J Mech Phys Solids 50(10):2123–2167. https://doi.org/10.1016/S0022-5096(02)00016-9
60. Miehe C (2002) Strain-driven homogenization of inelastic microstructures and composites based on an incremental variational formulation. Int J Numer Meth Eng 55(11):1285–1322. https://doi.org/10.1002/nme.515
61. Mielke A (2006) A mathematical framework for generalized standard materials in the rate-independent case. In: Multifield problems in solid and fluid mechanics, vol 28. Springer, Berlin, pp 399–428. https://doi.org/10.1007/978-3-540-34961-7_12
62. Ziegler H, Wehrli C (1987) The derivation of constitutive relations from the free energy and the dissipation function. In: Advances in applied mechanics, vol 25. Elsevier, pp 183–238.https://doi.org/10.1016/S0065-2156(08)70278-3
63. Rice JR (1971) Inelastic constitutive relations for solids: an internal-variable theory and its application to metal plasticity. J Mech Phys Solids 19(6):433–455. https://doi.org/10.1016/0022-5096(71)90010-X
64. Moreau JJ (2011) On unilateral constraints, friction and plasticity. In: Capriz G, Stampacchia G (eds) New variational techniques in mathematical physics. Springer, Berlin, pp 171–322. https://doi.org/10.1007/978-3-642-10960-7_7
65. Halphen B, Nguyen QS (1975) Sur les matériaux standard généralisés. Journal deMécanique 14:39–63
66. Biot MA (1965) Mechanics of incremental deformations. https://hal.science/hal-01352219
67. Kumar A, Lopez-Pamies O (2016) On the twopotential constitutive modeling of rubber viscoelastic materials. Comptes Rendus Mécanique 344(2):102–112. https://doi.org/10.1016/j.crme.2015.11.004

68. Fuhg JN, Bouklas N, Jones RE (2023) Stress representations for tensor basis neural networks: alternative formulations to Finger–Rivlin–Ericksen. arXiv:2308.11080 [cond-mat.soft]
69. Liu M, Liang L, Sun W (2020) A generic physics-informed neural network-based constitutive model for soft biological tissues. Comput Methods Appl Mech Eng 372:113402. https://doi.org/10.1016/j.cma.2020.113402
70. Flaschel M, Kumar S, De Lorenzis L (2023) Automated discovery of generalized standard material models with EUCLID. Comput Methods Appl Mech Eng 405:115867. https://doi.org/10.1016/j.cma.2022.115867
71. Bahador B, WaiChing S (2023) Physics-constrained symbolic model discovery for polyconvex incompressible incompressible hyperelastic materials. arXiv: 2310.04286 [cs.CE]
72. Kalina KA et al (2024) Neural network-based multiscale modeling of finite strain magneto-elasticity with relaxed convexity criteria. Comput Methods Appl Mech Eng 421:116739. https://doi.org/10.1016/j.cma.2023.116739
73. Kästner M et al (2012) Inelastic material behavior of polymers - experimental characterization, formulation and implementation of a material model. Mech Mater 52:40–57. https://doi.org/10.1016/j.mechmat.2012.04.011