J. Mech. Phys. Solids 180 (2023) 105416

Contents lists available at ScienceDirect

Journal of the Mechanics and Physics of Solids

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

Thermodynamically consistent neural network plasticity modeling and discovery of evolution laws

Knut Andreas Meyer *, Fredrik Ekre

Institute of Applied Mechanics, TU Braunschweig, Germany

ARTICLE INFO

Keywords:
Plasticity
Machine learning
Neural network
Data-driven modeling
Equation discovery

ABSTRACT

Over the past decade, advancements in computational frameworks and processing power have made deep neural networks increasingly viable for material modeling. However, purely data-driven models can yield non-physical predictions due to the lack of physical constraints. While recent works that incorporate physics into the training process partially address this issue, they offer no guarantees beyond the scope of the training data. To tackle this challenge, we propose a novel approach that embeds the neural network within the material model, inherently fulfilling thermodynamic laws. The network represents only the unknown physics allowing us to integrate knowledge accumulated from decades of constitutive modeling research into the data-driven methodology. By analyzing the trained, embedded networks, we recover existing evolution laws from artificial training data and discover new evolution laws from experimental data. The discovered evolution laws for isotropic and kinematic hardening can qualitatively predict an experimentally observed yield strength evolution, which conventional evolution laws cannot describe.

1. Introduction

Traditional material modeling requires researchers to propose various model equations and evaluate their ability to describe observed material behavior. In recent years, various alternatives to this approach have been suggested. The model-free approach, introduced by Kirchdoerfer and Ortiz (2016), aims to circumvent the modeling altogether and instead rely directly on an extensive database of experimental data. Another alternative is using a generic model to encode all material behaviors. A neural network is such a generic model according to the universal function approximation theorem (Hornik et al., 1989). With the rising availability of easy-to-use implementations and computational power, this approach has received increasing attention in the last decade. However, the idea is not new, cf. e.g. Ghaboussi and Sidarta (1998), but it also requires an extensive database of stress–strain history data. Unfortunately, such databases are not readily available due to the high cost and time resources needed to run experiments.

Physics Informed Neural Networks (PINNs) combine physical constraints with available training data by including known physical laws in the loss functions. For example, Borkowski et al. (2022) added positive plastic dissipation as a regularization term to improve their model. Masi et al. (2021) introduces the "Thermodynamics-based ANN", intrinsically fulfilling the first law of thermodynamics and making it easy to regularize wrt. positive dissipation (second law of thermodynamics). Heider et al. (2020) used spectral decomposition to reduce the required training data from experiments and enforce material objectivity (frame invariance), and later used this approach with recurrent neural networks and transfer learning in Fuchs et al. (2021). Malik et al. (2022) also use invariants as inputs to a neural network embedded in their model to describe the yield surface, which was trained on multiscale

* Corresponding author.
E-mail address: k.a.meyer@tu-bs.de (K.A. Meyer).

https://doi.org/10.1016/j.jmps.2023.105416
Received 17 April 2023; Received in revised form 26 July 2023; Accepted 8 August 2023
Available online 18 August 2023

simulations of foam structures. Wen et al. (2021) trained a tree-based regressor for plastic slip rate and isotropic hardening modulus before embedding this in a viscoplastic model to evaluate lithium's temperature and rate-dependent behavior.

Although the abovementioned approaches show impressive abilities to describe constitutive behaviors, they lead to black-box models. Using a completely different idea, Flaschel et al. (2021) introduced a novel method to identify hyperelastic material behavior based on full-field data and sparse regression. The concept was recently extended to linear viscoelasticity, Marino et al. (2023), plasticity, Flaschel et al. (2022), and general dissipative materials, Flaschel et al. (2023). For neural networks, Koeppe et al. (2022) identified model behavior using a Recurrent Neural Network, and Haghighat et al. (2023) used Feed-Forward PINNs to re-identify material parameters from noisy data.

The recent review paper by Rosenkranz et al. (2023) concluded that approaches for material modeling that enforce thermodynamics intrinsically, i.e. without relying on training data, show the best extrapolation behavior. Current approaches achieve this by formulating two potentials, the free energy and a dissipation potential, using so-called "input convex neural networks", Amos et al. (2017). See e.g. Linden et al. (2023) for hyperelasticity, Huang et al. (2022), Taç et al. (2023), Abdolazizi et al. (2023) for viscoelasticity, and Fuhg et al. (2023) for kinematically hardening plasticity.

In this paper, we also propose to embed neural networks into the standard dissipative constitutive model structure. Our formulation is inherently thermodynamic consistent and frame invariant, even without training. Instead of using the potentials, we directly formulate a set of evolution laws for the internal state variables, s, using neural networks, NN, written in a simplified manner as $\dot{s} = \mathrm{NN}(\boldsymbol{\epsilon}, s)$ (see Eqs. (11) and (14) for the exact descriptions). Such continuous formulations decouple the physics from the time integration, permitting the usage of existing time integration schemes. As opposed to the approaches discussed in the previous paragraph, the neural networks do not have to be input convex; hence their parameter values are not restricted. After training the model with an embedded neural network with experimental data, we analyze the neural networks to discover interpretable analytical evolution equations. Finally, we automatically generate new material models that are efficient, accurate, and thermodynamically consistent.

The paper is organized as follows: In Section 2, we derive and motivate the consistent modeling framework that embeds a neural network. Section 3.1 describes how the material models are trained to experimental data, while Section 3.2 outlines the method to discover analytical equations approximating the trained neural network. For the results in Section 4, we first show that the proposed approach works for artificial data generated by a standard J2-plasticity model with Ohno–Wang kinematic hardening in Section 4.1. Thereafter, we apply it to experimental data from Meyer and Ahlström (2023), showing that we can significantly improve existing models in Section 4.2. Sections 5 and 6 discuss and summarize our findings. Finally, in Appendix, we describe key implementation details for obtaining the parameter sensitivity of material models with embedded neural networks and implicit time integration.

2. Model formulation

Strain-driven material models calculate the stress, $\boldsymbol{\sigma}$, for a given time history of strains, $\boldsymbol{\epsilon}$. In addition to fitting experimental data, it is widely accepted that material models respect the following physical constraints:

- A convex Helmholtz' free energy.
- A non-negative dissipation.
- Frame invariance.

The Helmholtz's free energy, Ψ, for a plasticity formulation considering linear elasticity with isotropic and kinematic hardening, can typically be formulated as

$$\Psi(\boldsymbol{\epsilon}, \boldsymbol{\epsilon}_\mathrm{p}, \boldsymbol{b}, k) := \Psi_\mathrm{e} + \Psi_\mathrm{p}, \quad \Psi_\mathrm{e} := \frac{1}{2}[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\mathrm{p}] : \mathbf{E} : [\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\mathrm{p}], \quad \Psi_\mathrm{p} := \frac{1}{3} H_\mathrm{kin} \boldsymbol{b} : \boldsymbol{b} + \frac{1}{2} |H_\mathrm{iso}| k^2 \tag{1}$$

where $\boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}_\mathrm{p}$ are the total and plastic strains, respectively. $\mathbf{E}$ is the 4th order elastic stiffness tensor, while H_kin and H_iso are the kinematic and isotropic hardening moduli. $\boldsymbol{b}$ and k are internal variables associated with kinematic and isotropic hardening. Thermodynamic stability requires that Ψ is a convex function, as discussed in Maugin (1992), Lemaitre and Chaboche (1990) and Lubarda (2008). For the specific form in Eq. (1), it is sufficient that $\mathbf{E}$ is positive definite and that H_kin is positive to obtain a convex function. The absolute value of H_iso enables isotropic softening while maintaining the convexity. This choice is further discussed in Section 2.1.

In this work, the isotropic von Mises effective stress is used to define the yield criterion as

$$\Phi := f_\mathrm{vM}(\boldsymbol{\sigma} - \boldsymbol{\beta}) - [Y_0 + \kappa], \quad f_\mathrm{vM}(\boldsymbol{x}) := \sqrt{\frac{3}{2} \boldsymbol{x} : \boldsymbol{x}^\mathrm{dev}} \tag{2}$$

where Y_0 is the initial yield limit. $\bullet^\mathrm{dev}$ is the deviatoric part of the tensor $\bullet$, $\boldsymbol{\beta}$ is the back-stress conjugated to $\boldsymbol{b}$, and κ is the isotropic hardening stress conjugated to k. The stresses are then defined by the free energy as

$$\boldsymbol{\sigma} := \frac{\partial \Psi}{\partial \boldsymbol{\epsilon}} = \mathbf{E} : [\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\mathrm{p}] \tag{3a}$$

$$\boldsymbol{\beta} := -\frac{\partial \Psi}{\partial \boldsymbol{b}} = -\frac{2}{3} H_\mathrm{kin} \boldsymbol{b} \tag{3b}$$

$$\kappa := -\frac{\partial \Psi}{\partial k} = -|H_\mathrm{iso}| k \tag{3c}$$

Following these definitions, the dissipation is given as

$$\mathcal{D} = \boldsymbol{\sigma} : \dot{\boldsymbol{\epsilon}} - \dot{\Psi} = \boldsymbol{\sigma} : \dot{\boldsymbol{\epsilon}}_{\mathrm{p}} + \boldsymbol{\beta} : \dot{\boldsymbol{b}} + \kappa \dot{k} \tag{4}$$

where the Clausius–Duhem dissipation inequality requires that $\mathcal{D} \geq 0$. Associative evolution laws maximize $\mathcal{D}$, and is typically employed for the plastic strains. For $\dot{\boldsymbol{b}}$ and $\dot{k}$, this leads to linear hardening laws that do not accurately capture experimentally observed material behaviors, motivating the use of non-associative evolution laws. As an example, the evolution laws for Armstrong–Frederick kinematic hardening (Frederick and Armstrong, 2007) and Voce isotropic hardening, together with the associative evolution of plastic strains, are given by

$$\dot{\boldsymbol{\epsilon}}_{\mathrm{p}} = \dot{\lambda} \boldsymbol{\nu} \tag{5a}$$

$$\dot{\boldsymbol{b}} = -\dot{\lambda} \boldsymbol{g}_{\mathrm{kin}}(\cdot) = -\dot{\lambda} \left[\boldsymbol{\nu} - \frac{3}{2} \frac{\boldsymbol{\beta}}{\beta_\infty} \right] \tag{5b}$$

$$\dot{k} = -\dot{\lambda} g_{\mathrm{iso}}(\cdot) = -\dot{\lambda} \left[1 - \frac{\kappa}{\kappa_\infty} \right] \tag{5c}$$

where

$$\boldsymbol{\nu} := \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{\sigma}} = \frac{3}{2} \frac{\boldsymbol{\sigma}^{\mathrm{dev}} - \boldsymbol{\beta}^{\mathrm{dev}}}{f_{\mathrm{vM}}(\boldsymbol{\sigma} - \boldsymbol{\beta})} \tag{6}$$

The functions $\boldsymbol{g}_{\mathrm{kin}}(\cdot)$ and $g_{\mathrm{iso}}(\cdot)$ denote arbitrary evolution laws that are discussed later, while the right hand side shows the specific Armstrong–Frederick and Voce hardening laws.

The plastic multiplier, $\dot{\lambda}$, is given by the KKT loading/unloading conditions,

$$\boldsymbol{\Phi} \leq 0, \quad \dot{\lambda} \geq 0, \quad \boldsymbol{\Phi} \dot{\lambda} = 0 \tag{7}$$

for rate independent behavior. For viscoplastic behavior, $\dot{\lambda}$ is defined by an overstress function, η,

$$\dot{\lambda} = \frac{1}{t^*} \eta(\boldsymbol{\sigma}, \boldsymbol{\beta}, \kappa) \tag{8}$$

where $\eta \geq 0$ is unitless and $t^* > 0$ is a characteristic time. In this work, we adopt the Norton overstress function,

$$\eta = \left[\frac{\langle \boldsymbol{\Phi} \rangle}{Y_0} \right]^n \tag{9}$$

where n is a material parameter and $\langle \bullet \rangle := \max(\bullet, 0)$ is the Macaulay bracket. For the evolution laws in Eq. (5), the dissipation inequality is fulfilled,

$$\mathcal{D} = \dot{\lambda} \left[\boldsymbol{\Phi} + Y_0 + \frac{3}{2} \frac{\boldsymbol{\beta} : \boldsymbol{\beta}}{\beta_\infty} + \frac{\kappa^2}{\kappa_\infty} \right] \geq 0 \tag{10}$$

if $\beta_\infty > 0$ and $\kappa_\infty > 0$. We will denote this complete model, with the Norton overstress function, as the "Chaboche"-model. It is well known that this model cannot accurately describe the behavior of many materials. A critical modeling choice is the form and parameterization of the hardening evolution laws (5b) and (5c). Especially for kinematic hardening, many different equations have been proposed; see Table 1 in Xiao et al. (2012) for an overview. As these functions are unknown, a universal function approximator, such as a neural network, is suitable to model these. However, such a material model must fulfill the previously mentioned criteria.

Convexity is already assured by not modifying the free energy. To fulfill the dissipation inequality, we propose the following novel evolution equations

$$\dot{\boldsymbol{b}} = -\dot{\lambda} \left[\left[1 - [\boldsymbol{\nu} : \boldsymbol{\beta}] \, \mathrm{NN}_{\mathrm{k},\nu} \right] \boldsymbol{\nu} - \mathrm{NN}_{\mathrm{k},\beta} \boldsymbol{\beta} \right] \tag{11a}$$

$$\dot{k} = -\dot{\lambda} \left[1 - \kappa \mathrm{NN}_{\mathrm{iso}} \right] \tag{11b}$$

where $\mathrm{NN}_{\mathrm{k},\nu}$, $\mathrm{NN}_{\mathrm{k},\beta}$, and $\mathrm{NN}_{\mathrm{iso}}$ are scalar outputs from a neural network. The scalar factors, $\boldsymbol{\nu} : \boldsymbol{\beta}$ and κ, in front of $\mathrm{NN}_{\mathrm{k},\nu}$ and $\mathrm{NN}_{\mathrm{iso}}$, makes it possible to inherently fulfill the Clausius–Duhem dissipation inequality, $\mathcal{D} \geq 0$,

$$\mathcal{D} = \dot{\lambda} \left[\boldsymbol{\Phi} + Y_0 + [\boldsymbol{\nu} : \boldsymbol{\beta}]^2 \mathrm{NN}_{\mathrm{k},\nu} + [\boldsymbol{\beta} : \boldsymbol{\beta}] \mathrm{NN}_{\mathrm{k},\beta} + \kappa^2 \mathrm{NN}_{\mathrm{iso}} \right] \tag{12}$$

by requiring positive neural network outputs. A non-negative activation function in the last layer fulfills this requirement; see Section 2.3. Furthermore, those scalar factors cause H_{kin} and H_{iso} to retain their physical interpretation wrt. the initial plastic stiffness. We note that a straightforward choice of evolution law, e.g. $\dot{k} = -\dot{\lambda} \mathrm{NN}_{\mathrm{iso}}$, requires more complicated restrictions on the neural network to ensure a positive dissipation.

Finally, we require a frame invariant model formulation, implying that the model response is independent of the chosen coordinate system. In particular, tensor components are neither inputs nor outputs of the neural network; inputs are independent invariants of the current material state, and outputs are the scalar factors in Eq. (11). In the most general case for the proposed

model, we can then use the following 19 invariant inputs (see e.g. Boehler, 1977)

$$\begin{array}{cccc} \kappa & \mathrm{tr}(\boldsymbol{\epsilon}_\mathrm{p}^2) & \mathrm{tr}(\boldsymbol{\beta}^2) & \\ \mathrm{tr}(\boldsymbol{\nu}\boldsymbol{\beta}) & \mathrm{tr}(\boldsymbol{\nu}\boldsymbol{\epsilon}_\mathrm{p}) & \mathrm{tr}(\boldsymbol{\beta}\boldsymbol{\epsilon}_\mathrm{p}) & \mathrm{tr}(\boldsymbol{\beta}^2\boldsymbol{\epsilon}_\mathrm{p}^2) \\ \mathrm{tr}(\boldsymbol{\nu}^3) & \mathrm{tr}(\boldsymbol{\epsilon}_\mathrm{p}^3) & \mathrm{tr}(\boldsymbol{\beta}^3) & \mathrm{tr}(\boldsymbol{\beta}^2\boldsymbol{\epsilon}_\mathrm{p}) \\ \mathrm{tr}(\boldsymbol{\nu}\boldsymbol{\beta}^2) & \mathrm{tr}(\boldsymbol{\nu}\boldsymbol{\epsilon}_\mathrm{p}^2) & \mathrm{tr}(\boldsymbol{\nu}\boldsymbol{\beta}\boldsymbol{\epsilon}_\mathrm{p}) & \mathrm{tr}(\boldsymbol{\beta}\boldsymbol{\epsilon}_\mathrm{p}^2) \\ \mathrm{tr}(\boldsymbol{\nu}^2\boldsymbol{\epsilon}_\mathrm{p}) & \mathrm{tr}(\boldsymbol{\nu}^2\boldsymbol{\beta}) & \mathrm{tr}(\boldsymbol{\nu}^2\boldsymbol{\beta}^2) & \mathrm{tr}(\boldsymbol{\nu}^2\boldsymbol{\epsilon}_\mathrm{p}^2) \end{array} \tag{13}$$

noting that $\mathrm{tr}(\boldsymbol{\nu}^2) = 3/2$ is not used as it is constant. Furthermore, the von Mises yield criterion implies that $\mathrm{tr}(\boldsymbol{\nu}) = \mathrm{tr}(\boldsymbol{\epsilon}_\mathrm{p}) = \mathrm{tr}(\boldsymbol{\beta}) = 0$ and that these tensors are symmetric. The choice of using $\boldsymbol{\nu}$ instead of, e.g., $\boldsymbol{\sigma}^\mathrm{dev}$, is motivated by the dependence between κ, other invariants, and the effective stress for rate-independent loading. The final model uses a reduced set of the above invariants, as discussed in Section 2.3.

While the presented model is initially isotropic, structure tensors, cf. e.g. Holzapfel and Ogden (2010), can extend the model to anisotropic responses. For example, a constant anisotropic elastic stiffness, $\mathbf{E}$, would add to the invariants by considering, e.g., $\boldsymbol{\beta} : \mathbf{E} : \boldsymbol{\beta}$. However, in the case of isotropy, this reduces to $2G\mathrm{tr}(\boldsymbol{\beta}^2)$ (where G is the elastic shear modulus) and does not add an independent invariant. Similarly, an evolving 4th order anisotropy tensor, $\mathbf{C}$, can be used in the yield criterion (cf. Meyer and Menzel, 2021). This tensor would be another state variable giving rise to additional invariants, such as $\boldsymbol{\beta} : \mathbf{C} : \boldsymbol{\beta}$.

2.1. Isotropic softening

The discussion so far has considered an initially hardening behavior. However, many metallic materials, such as e.g. carbon steel, may exhibit initial isotropic softening. A negative hardening modulus, $H_\mathrm{iso} < 0$, results in isotropic softening, neglecting the absolute value in Eq. (1). However, without the absolute value, the Helmholtz free energy becomes concave for a negative hardening modulus. Therefore, for $H_\mathrm{iso} < 0$, we use the following evolution law instead:

$$\dot{k} = \dot{\lambda}\left[1 + \kappa \mathrm{NN}_\mathrm{iso} + \frac{\kappa}{Y_0}\right] \tag{14}$$

In this case, $\kappa \leq 0$, and the evolution of k has the opposite sign in comparison to in Eq. (11b). The additional term, κ/Y_0 ensures that the dissipation,

$$D = \dot{\lambda}\left[\Phi + \left[\sqrt{Y_0} + \frac{\kappa}{\sqrt{Y_0}}\right]^2 + \mathrm{NN}_{\mathrm{k},\nu}[\boldsymbol{\nu} : \boldsymbol{\beta}]^2 + \mathrm{NN}_{\mathrm{k},\beta}\boldsymbol{\beta} : \boldsymbol{\beta} + \kappa^2 \mathrm{NN}_\mathrm{iso}\right] \tag{15}$$

is non-negative for non-negative outputs from the neural network. For a zero-valued network, this corresponds to a isotropic saturation stress of $-Y_0$.

2.2. Convergence requirements

The neural networks are chosen for their ability to approximate any function. However, during training, this implies that the predicted material response can become non-convergent. In practice, we have found it necessary to (i) avoid a zero-sized yield surface and (ii) avoid too rapid softening. Both these lead to lack of solutions to the local plasticity residual equations. The constraints described below aid convergence during training but should typically not affect the response of trained models.

2.2.1. Minimum yield strength

An isotropic yield limit $Y_0 + \kappa < 0$ yields $\Phi > 0$. This result contradicts the KKT conditions and prevents solutions to the local plasticity problem. To ensure a positive yield limit, we constrain the isotropic hardening stress, κ, via the smooth bounds function h defined as

$$h(x, a, b) := \begin{cases} x & \frac{x-b}{a-b} \geq 1 \\ [a-b]\exp\left(\frac{x-a}{a-b}\right) + b & \frac{x-b}{a-b} < 1 \end{cases} \tag{16}$$

For $a > b$, h provides a smooth lower bound of x such that

$$h(x, a, b) = \begin{cases} x & x \geq a \\ b & x \to -\infty \end{cases} \tag{17}$$

where we note that $\mathrm{d}h/\mathrm{d}x \to 1$ as $x \to a$. Specifically, we constrain κ as

$$\hat{\kappa} = h(\kappa + Y_0, Y_\mathrm{low}, Y_\mathrm{min}) - Y_0 \tag{18}$$

and use $\hat{\kappa}$ in the yield criterion such that $Y_0 + \hat{\kappa} \geq Y_\mathrm{min}$. In all cases, $Y_\mathrm{low} = 2Y_\mathrm{min} = 40\,\mathrm{MPa}$.

2.2.2. Maximum softening

In a more detailed analysis compared to Meyer (2020), we analyze the existence of a solution for a rate-independent material. Specifically, we consider a loading with $\dot{\epsilon} = \dot{\epsilon}\boldsymbol{n}$, where $\dot{\epsilon} > 0$ and $\boldsymbol{n}$ is a 2nd order, normalized tensor, describing the loading direction. The KKT conditions in Eq. (7), implies that $\dot{\Phi} = 0$ during plastic loading, resulting in

$$\dot{\boldsymbol{\sigma}}^{\text{dev}} = 2G\dot{\epsilon}\boldsymbol{n}^{\text{dev}} - 2G\dot{\lambda}\boldsymbol{\nu} \tag{19}$$

$$\dot{\Phi} = 0 = 2G\dot{\epsilon}\boldsymbol{\nu} : \boldsymbol{n}^{\text{dev}} - \dot{\lambda}\left[3G + \frac{2}{3}H_{\text{kin}}\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot) + |H_{\text{iso}}|g_{\text{iso}}(\cdot)\right] \tag{20}$$

where we have isotropic elasticity with shear modulus G, and $\boldsymbol{g}_{\text{kin}}(\cdot)$ and $g_{\text{iso}}(\cdot)$ are defined in Eq. (5), with specific forms in Eqs. (11) and (14). During plastic loading, $\boldsymbol{\nu} : \boldsymbol{n}^{\text{dev}} > 0$, implying that

$$3G + [2H_{\text{kin}}/3]\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot) + |H_{\text{iso}}|g_{\text{iso}}(\cdot) > 0 \tag{21}$$

is required as $\dot{\lambda} \geq 0$. To fulfill this requirement, we constrain the evolution laws by using the smooth bounding function h from Eq. (16) as

$$\hat{g}_{\text{iso}}(\cdot) = \frac{h(|H_{\text{iso}}|g_{\text{iso}}(\cdot), -1.8G, -2.0G)}{|H_{\text{iso}}|} \tag{22}$$

$$\boldsymbol{\nu} : \hat{\boldsymbol{g}}_{\text{kin}}(\cdot) = \frac{h\left(\frac{2H_{\text{kin}}}{3}\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot), \frac{-G}{5}, \frac{-G}{2}\right)}{2H_{\text{kin}}/3} \tag{23}$$

These bounds ensure that

$$3G + [2H_{\text{kin}}/3]\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot) + |H_{\text{iso}}|g_{\text{iso}}(\cdot) > 0.5G \tag{24}$$

As most materials only exhibit isotropic softening, more isotropic than kinematic softening is allowed. While Eq. (22) directly modifies the isotropic hardening law, the limit for the kinematic evolution is implicit. A scaling factor, s, for the neural network outputs yields,

$$\hat{\boldsymbol{g}}_{\text{kin}}(\cdot) = \boldsymbol{\nu} - s\,[\boldsymbol{\nu} : \boldsymbol{\beta}]\,\text{NN}_{\text{k},\nu}\boldsymbol{\nu} - s\text{NN}_{\text{k},\beta}\boldsymbol{\beta} \tag{25}$$

$$s = \frac{H_{\text{kin}} - h\left(\frac{2}{3}H_{\text{kin}}\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot), \frac{-G}{5}, \frac{-G}{2}\right)}{H_{\text{kin}} - \frac{2}{3}H_{\text{kin}}\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot)} \tag{26}$$

such that $s = 1$ if $[2/3]H_{\text{kin}}\boldsymbol{\nu} : \boldsymbol{g}_{\text{kin}}(\cdot) > -G/5$. Using the modified functions, $\hat{g}_{\text{iso}}$ and $\hat{\boldsymbol{g}}_{\text{kin}}$ in the evolution of κ and $\boldsymbol{\beta}$, ensures that a solution can be found.

2.3. Network architecture and scaling

Recurrent neural networks are typically used to model history-dependent behavior. The material model presented herein may, in fact, be interpreted as a recurrent network since the current step depends on the previous state. Directly using such network architectures to model the stress–strain relationship has been done, see e.g. Mozaffar et al. (2019) who employs "Long Short Term Memory" (LSTM) networks. However, for the proposed model formulation, the network is only a component of the complete model that produces the output variables for the next time step. A recurrent network is not required for our formulation since the outer model structure implicitly considers recurrence. Consequently, we use a standard feed-forward neural network.

Considered as a function, a feed-forward neural network accepts a vector of inputs, $\underline{\boldsymbol{x}}_0$, and gives a vector of outputs, $\underline{\boldsymbol{x}}_N$, by passing through its N layers. The layer i is described as

$$\underline{\boldsymbol{x}}_i = a_i.(\underline{\underline{\boldsymbol{A}}}_i\,\underline{\boldsymbol{x}}_{i-1} + \underline{\boldsymbol{b}}_i) \tag{27}$$

where the notation $a.(\underline{\boldsymbol{x}})$ implies application of the activation function a on each component of the vector $\underline{\boldsymbol{x}}$ individually. The trainable parameters of the network are then the weights in the matrices $\underline{\underline{\boldsymbol{A}}}_i$ and the bias vectors $\underline{\boldsymbol{b}}_i$ for $i \in [1, N]$.

In this study, we reduce the number of invariants from Eq. (13) and consider the following six:

$$\begin{aligned} &I_1 = \kappa, \quad && I_2 = \boldsymbol{\beta} : \boldsymbol{\beta}, \quad && I_3 = \boldsymbol{\nu} : \boldsymbol{\beta}, \\ &I_4 = \boldsymbol{\beta} : \boldsymbol{\epsilon}_{\text{p}}, \quad && I_5 = \boldsymbol{\nu} : \boldsymbol{\epsilon}_{\text{p}}, \quad && I_6 = \boldsymbol{\epsilon}_{\text{p}} : \boldsymbol{\epsilon}_{\text{p}} \end{aligned} \tag{28}$$

as network inputs. These choices are motivated by the von Mises yield surface. During the development of the model we tested a few different network dimensions and activation functions, and, while many gave good results, a feed-forward network consisting of five fully connected layers with widths 6-6-6-6-5-3 produced the most stable results. The activation function for the first four layers was chosen as $a_i(x) = \tanh(x)$, while the last activation function, $a_5(x) = x^2$ ensured the previously discussed non-negativity of the output.

In the evolution laws, Eqs. (11) and (14), the unit of the network outputs is the inverse of stress. To make the networks unit independent and to ensure weights with reasonable magnitudes, the network outputs are divided by a constant, $\text{NN}_0 = 1000\,\text{MPa}$. Similarly, many of the inputs also have the unit of stress. These are divided by NN_0^n, where n is an exponent depending on the input's unit (e.g. $n = 2$ for $\text{tr}(\boldsymbol{\beta}^2)$).

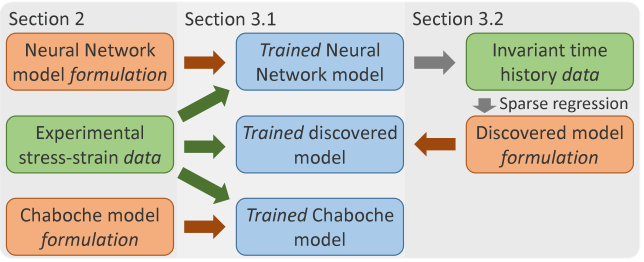


Fig. 1. Training methodology workflow.

3. Training

The overall training procedure in this paper is described in Fig. 1, along with the relevant sections. The starting points are the Neural Network and Chaboche models, formulated in Section 2. Independent of the material formulation, we use the same training process as described in Section 3.1. Section 3.2 outlines the methodology for discovering new analytical evolution equations based on the trained Neural Network model. After equation discovery, a new model is automatically generated and trained with experimental data following the process in Section 3.1.

3.1. Training of material models

Parameter identification for classical material modeling is closely related to training in machine learning: the end goal is to minimize some objective or loss function wrt. a set of parameters. A difference between regular material models and neural networks is that neural networks are overparameterized such that multiple equivalent minima may exist. Because our models may include overparameterized neural networks, we use training algorithms and procedures from machine learning. It turns out that these approaches perform well for regular material models, and we use the same training procedure irrespective of the material model type.

3.1.1. Parameter bounds and penalization

The elastic parameters are fixed to Young's modulus, $E = 210\,\mathrm{GPa}$, and Poisson's ratio, $\nu = 0.3$. For some parameters participating in the parameter identification, reasonable bounds may be given a-priori. One example is the initial yield strength, Y_0. We use gradient-based optimization in the present work, which requires differentiable constraints. Using the smooth bounding function in Eq. (16) for both upper and lower bounds ensures differentiability. One issue with this function is that the gradient vanishes when the value of a parameter approaches the hard bounds. To remedy this, we first scale the data such that values within the smooth bounds are between 0 and 1, and use the so-called "epsilon insensitive loss", cf. Lee et al. (2005), to apply L2 regularization on the distance of scaled values to this range (i.e. both -0.1 and 1.1 are penalized by 0.1^2).

Another common issue with neural networks is unstable training due to exploding gradients, which L2 regularization also remedies (Pascanu et al., 2013). In the present study, however, the regular L2 penalization of the neural network's weights and biases reduced the training performance. Therefore, all network parameters were treated as regular material parameters, with soft bounds -1 and 1. The only exceptions were the biases in the last layer, where bounds yielding reasonable saturation levels were used (cf. parameters κ_∞ and β_∞ in the Chaboche model).

In summary, given a design variable x, the actual parameter p used in the model is calculated as

$$g_p(p) = \frac{p - p_{\min}}{p_{\max} - p_{\min}}, \qquad \hat{p} = \begin{cases} h\left(x, 1, g\left(p^{\mathrm{h}}_{\max}\right)\right) & x > 1 \\ x & 0 \le x \le 1 \\ h\left(x, 0, g\left(p^{\mathrm{h}}_{\min}\right)\right) & x < 0 \end{cases}, \qquad p = \hat{p}\left[p_{\max} - p_{\min}\right] + p_{\min} \tag{29}$$

where $p_{\min}$ and $p_{\max}$ are the soft bounds (cf. a in Eq. (16)) and $p^{\mathrm{h}}_{\min}$ and $p^{\mathrm{h}}_{\max}$ the hard bounds (cf. b in Eq. (16)). Consequently, each parameter p_i has a corresponding design variable x_i, which is between 0 and 1 if inside the soft bounds. Values outside these bounds are added to the regularization loss, L^{r},

$$L^{\mathrm{r}} := \sum_{i=1}^{N_{\mathrm{p}}} g_x(x_i)^2, \qquad g_x(x) = \begin{cases} x & x < 0 \\ 0 & 0 \le x \le 1 \\ x - 1 & x > 1 \end{cases} \tag{30}$$

3.1.2. Loss function

To have comparable values for the errors between each experiment, simulated stress components, σ_{ij}^{s}, and experimentally measured stress components, σ_{ij}^{e} are normalized by the number of simulation steps, N_{stp} and the stress range in the experiment.

$$\hat{\sigma}_{ij}^{\bullet} = \frac{\sigma_{ij}^{\bullet}}{\sqrt{N_{\mathrm{stp}}}\left[\max\left(\sigma_{ij}^{\mathrm{e}}\right) - \min\left(\sigma_{ij}^{\mathrm{e}}\right)\right]} \tag{31}$$

for $\bullet$ either s (simulated) or e (experiment). Using these normalized stresses, the following loss, L_k is calculated for each simulation, k,

$$L_k := \sum_{ij}\left[\sum_{l=1}^{N}\left[\hat{\sigma}_{ij}^{\mathrm{s}}(t_l) - \hat{\sigma}_{ij}^{\mathrm{e}}(t_l)\right]^2 + w_{\mathrm{g}}\sum_{l=1}^{N-10}\left[\left[\left[\hat{\sigma}_{ij}^{\mathrm{s}}\right]_{l_{10}} - \left[\hat{\sigma}_{ij}^{\mathrm{s}}\right]_{l}\right] - \left[\left[\hat{\sigma}_{ij}^{\mathrm{e}}\right]_{l_{10}} - \left[\hat{\sigma}_{ij}^{\mathrm{e}}\right]_{l}\right]\right]^2\right] \tag{32}$$

where $\left[\hat{\sigma}_{ij}^{\bullet}\right]_l$ denotes the stress in time step l , $l_{10} = l + 10$, and w_{g} is a weighting factor. The first term in Eq. (32) is the standard L2-loss function.

While Vlassis and Sun (2021) showed that Sobolev training is advantageous for learning accurate models, the experimental training data in the present study does not have the required derivative information. However, finite differences can approximate the directional derivative along the loading direction, which the second term in Eq. (32) uses to regularize the loss function. This addition serves two purposes: Firstly, solutions with oscillation stress responses may occur due to the neural networks' high flexibility. The regularization penalizes such oscillations. Secondly, most experimental scatter in stress–strain curves is variation between test samples. While the curves for repeated experiments have similar shapes with low noise, the stress levels may vary slightly. Penalizing the difference in stress changes gives more importance to the shape of the curve, making the loss function less sensitive to sample variations. During the development and testing, we found that a weighting factor $w_{\mathrm{g}} = 1$ worked well with using data ten points apart to calculate the numerical gradient.

For a given batch of M experiments, the total loss is then calculated as

$$L := w_{\mathrm{r}} L^{\mathrm{r}} + \frac{1}{\sqrt{M}}\sum_{k=1}^{M} L_k \tag{33}$$

where L^{r} is defined in Eq. (30) and w_{r} is a weighting factor. For normalized losses in L_k, $w_{\mathrm{r}} = 1$ worked well, and we found that the epsilon-insensitive regularization in L^{r} made the choice of w_{r} less critical than for standard L2-regularization.

3.1.3. Optimizer and training procedure

Three different gradient-based optimizers, available in `Flux.jl` (Innes, 2018), were tested: `RMSProp` (Tieleman and Hinton, 2012), `Adam` (Kingma and Ba, 2015), and `RAdam` (Liu et al., 2020). For the specific problem, the `RAdam` optimizer gave the most reliable results. While `RMSProp` and `Adam` performed similarly, the final losses were somewhat higher for some initial guesses. A coarse sweep revealed that while fine-tuning could improve the convergence rate, the default values worked sufficiently well for the present study. While efficient implementations of parameter sensitivity for neural networks are abundant, this is not true for nonlinear material models with implicit time integration. Appendix outlines our efficient method for obtaining this sensitivity that the gradient-based optimizers require.

The expressiveness of the embedded neural network implies that multiple local minima exist. Several strategies for avoiding getting stuck in such local minima exist in the literature, such as "Adaptive activation functions" (Jagtap et al., 2020) and mini-batch training (Keskar et al., 2017). In this study, we experienced that mini-batch training significantly improved the performance when we used one experiment in each batch. In addition, one experiment is used as a validation to prevent overfitting. The overall training procedure consists of the following steps to learn the scaled parameters x:

- For $N_{minibatch}$ updates:
  - Simulate and update x for each mini-batch set.
  - Simulate the validation set: Save loss.
  - Simulate the full set: Save loss and update x.
- For $N_{fullbatch}$ updates:
  - Simulate the validation set: Save loss.
  - Simulate the full set: Save loss and update x.

where the full set typically is the collection of all mini-batches. The experiment in the validation set is never used to update the parameters.

3.2. Equation discovery

The equation discovery aims to identify analytical expressions that describe the evolution laws, i.e. the neural network outputs, $\mathrm{NN}_{\mathrm{iso}}$, $\mathrm{NN}_{\mathrm{k},\beta}$, and $\mathrm{NN}_{\mathrm{k},v}$. Various techniques exist, such as symbolic regression (cf. Holland, 1992 (originally published 1975)) and

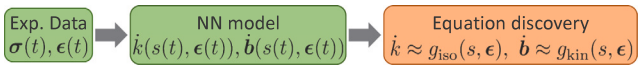


Fig. 2. Illustration of the discovery of equation from data generated by the trained neural network.

sparse regression (cf. e.g. Brunton et al., 2016). Herein, we adopted a modified sparse regression technique based on (Flaschel et al., 2021), described in more detail below.

Fig. 2 illustrates the overall procedure for discovering evolution equations for k and $\boldsymbol{b}$, as functions of the state, $s = \{\epsilon_{\text{p}}, \boldsymbol{\beta}, \kappa\}$, and the current strain, $\boldsymbol{\epsilon}$. Specifically we note that the mapping between current states, s, and evolution laws, $\dot{k}$ and $\dot{\boldsymbol{b}}$, is not available in the experimental data but can be obtained from the (trained) surrogate neural network model. Theoretically, it would be possible to apply symbolic regression on the experimental data to circumvent the neural network model. However, this would result in a nested optimization with parameter identification for each proposed symbolic expression. Due to the vast number of possible combinations in symbolic regression, this approach quickly becomes unfeasible. Another possibility is taking the penalization from the sparse regression and identifying parameters for a model that includes all candidate functions. However, as discussed in Flaschel et al. (2021), the sparsity regularization makes the optimization problem highly non-convex even when combined with linear regression. Combining sparsity regularization with the already complicated material parameter identification problem will likely lead to an even harder problem. This approach is out of the present study's scope. Finally, we note that combining neural network training with equation discovery has been successful for other application areas in physics, see e.g. Udrescu and Tegmark (2020).

In practice, we do not discover the equations g_{iso} and g_{kin} in Fig. 2 directly. Instead, we discover the network outputs, $[\text{NN}_{\text{iso}}, \text{NN}_{\text{k},\beta}, \text{NN}_{\text{k},\nu}]$, and then use Eqs. (11) and (14) to calculate g_{iso} and g_{kin}. This approach has the advantage that positive function values are sufficient for thermodynamical consistency.

3.2.1. Generation of input data

Following the discussion in the previous section, the equation discovery requires input data consisting of pairs of network outputs, $[\text{NN}_{\text{iso}}, \text{NN}_{\text{k},\beta}, \text{NN}_{\text{k},\nu}]$, and network inputs, $[I_1, \dots, I_6]$. Given a neural network that accurately describes the evolution laws for all possible scenarios, we could use a large set of random invariant values as input data to generate a database of the network's outputs. In practice, however, the network is only trained on invariants encountered during training. Consequently, we run the full training batch while calculating and saving the invariants, $I_1, \dots, I_6$ in each time step. We then calculate the network outputs corresponding to invariants in each time step.

Each invariant, as well as each network output, is scaled by its maximum absolute value. The scaled invariants fulfill $\hat{I}_i \in [-1, 1]$ for all time steps. Due to the activation function in the last layer, the scaled outputs fulfill $\hat{y}_\bullet \in [0, 1]$ in all time steps (where $\bullet$ represents iso, k, ν, or k, β). Finally, duplicate values are removed from the training set.

3.2.2. Generation of candidate functions

Each candidate function, $f_i(\hat{\underline{\boldsymbol{I}}})$, should take as input a vector $\hat{\underline{\boldsymbol{I}}}$ of scaled invariants (discussed above) and return a scalar output. The approximation of the scaled total output, $\hat{y}$, is then

$$\hat{y} \approx \sum_i \hat{p}_i f_i(\hat{\underline{\boldsymbol{I}}}) \tag{34}$$

For this approach to be suitable, considering the scaling of the invariants, we require that f are positively homogeneous functions. This implies that polynomials are allowed, but not, e.g., trigonometric functions or exponential functions. Given these restrictions, we build a basis with the following elementary basis functions:

- Polynomials of I_j up to 2nd order.
- Polynomials of $\langle I_j \rangle$ up to 2nd order.
- Polynomials of $\langle -I_j \rangle$ up to 2nd order.
- Combinations of $\langle I_j \rangle^n$ with rational and integer exponents, $n \in \{0, 1, 2, 1/2, 1/3\}$ (max 3 factors per term).
- Combinations of $\langle -I_j \rangle^n$ with rational and integer exponents, $n \in \{0, 1, 2, 1/2, 1/3\}$ (max 3 factors per term).

where the use of the Macaulay bracket, $\langle \bullet \rangle$, enables different parameters for positive and negative values of the same inputs. This gives in total 2680 possible functions after filtering out some known equivalent bases (e.g. we know that $I_3 = \boldsymbol{\beta} : \boldsymbol{\beta} \geq 0$ so all terms containing $\langle -I_3 \rangle$ can be removed). Additionally, from each group of functions with matching values for every sample in the dataset, only the one with the lowest number of active invariants is retained.

The limitation of positive homogeneous function can be lifted by not scaling the invariant inputs, which could allow more general candidate functions, such as exponential functions. In these cases, it often makes sense to include a parameter inside the functions, i.e. $\exp(p\kappa)$, where p is a parameter. While this is not directly compatible with the standard regression technique, Marino et al. (2023) use a large set of parameterized exponential functions in combination with k-means clustering to find suitable values. Similar to symbolic regression, this approach makes it possible to use more general candidate functions.

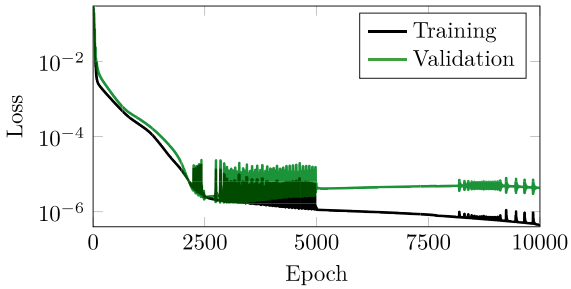


Fig. 3. Loss function evolution for model training on artificial data.

3.2.3. Discovery

Given a number of candidate functions, the goal is to identify which functions can best represent the neural network outputs. To study the trade-off between complexity and accuracy, we would like to construct the Pareto front by comparing the loss and the number of active functions. For a low number, N_A, of active functions and a reasonable number, N_C, of candidate functions, it is possible to try all possible combinations. The number of combinations is the binomial; N_C choose N_A. Consequently, the number of possible combinations increases quickly with both the number of active functions and the available candidate functions. For the experimental data discussed later, after filtering based on the dataset, 869 candidate functions remained. For this case, there are about 110 million possible combinations for three terms, which takes about 10 min to evaluate (on a regular laptop) for each network output. Increasing to four terms yields about 24 billion candidates, which, while still feasible, motivates the use of more efficient algorithms. Therefore, we employ the sparse regression algorithm described in Flaschel et al. (2021) to reduce the number of candidate functions. Specifically, running the sparse regression with multiple initial guesses determines the most frequently selected terms. We then run a binomial search using these terms as candidate functions. This selection method is validated by running the binomial search directly for up to three terms.

4. Results

4.1. Artificial data

To validate the proposed workflow, we use artificial data generated by a known plasticity model. Specifically, the Ohno–Wang kinematic hardening law (Ohno and Wang, 1993),

$$\dot{\boldsymbol{b}} = -\dot{\lambda}\left[\boldsymbol{\nu} - \frac{3}{2}\frac{\boldsymbol{\beta}}{f_{\mathrm{vM}}(\boldsymbol{\beta})}\frac{\langle \boldsymbol{\nu} : \boldsymbol{\beta}\rangle}{\beta_\infty}\left[\frac{f_{\mathrm{vM}}(\boldsymbol{\beta})}{\beta_\infty}\right]^m\right] \tag{35}$$

In contrast to the Armstrong–Frederick hardening law, the neural network cannot exactly represent this equation. For the training data generation, we use kinematic hardening parameters $H_{\mathrm{kin}} = 500\,\mathrm{GPa}$, $\beta_\infty = 500\,\mathrm{MPa}$, and $m = 2$. In addition, we have elastic parameters $E = 210\,\mathrm{GPa}$ and $\nu = 0.3$, initial yield limit $Y_0 = 350\,\mathrm{MPa}$, Voce-type isotropic hardening (Eq. (5c)) with $H_{\mathrm{iso}} = 25\,\mathrm{GPa}$, and $\kappa_\infty = 100\,\mathrm{MPa}$, and Norton overstress (Eqs. (8) and (9)) with $t^* = 1\,\mathrm{s}$ and $n = 2$.

4.1.1. Model training

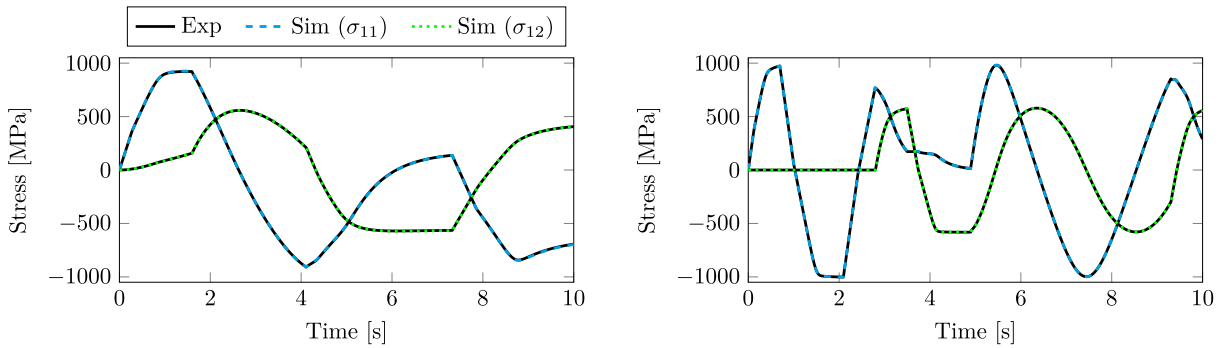
The artificial training data is generated by simulating strain-controlled biaxial experiments, corresponding to axial-torsion loading of tubular specimens. The loading is defined by ϵ_{11} and ϵ_{12}, while $\epsilon_{13} = \epsilon_{23} = 0$ and $\sigma_{22} = \sigma_{33} = 0$. Four different time histories of ϵ_{11} and ϵ_{12} are simulated to create three mini-batch sets for training and one set for validation. We then follow the training strategy outlined in Section 3, resulting in the loss function evolution in Fig. 3. In this case, the lowest validation loss occurs during the mini-batch training phase in epoch 2845.

The low relative losses in Fig. 3 show that the neural network evolution can replicate the material response for the Ohno–Wang evolution law. Furthermore, Fig. 4a shows the axial and shear stress responses compared to the first training batch with artificial experimental data. The results are similar for the two other training batches. As for the training data, no visual difference can be observed between the neural network model and the artificial experiment data in the validation case in Fig. 4b.

4.1.2. Equation discovery

The previous subsection showed that the neural network could approximate the Ohno–Wang evolution law considering the overall material response. The next question is whether this evolution law can be re-identified from the trained neural network, assuming the correct equations are part of the candidate functions. For the specific material parameters used to generate the artificial data, in particular $m = 2$, and considering the structure of the kinematic evolution law in Eq. (11a), we expect to identify

$$\mathrm{NN}_{\mathrm{iso}} = p_1, \qquad \mathrm{NN}_{\mathrm{k},\beta} = p_2\,\langle I_3\rangle\,\sqrt{I_2}, \quad I_3 = \boldsymbol{\nu} : \boldsymbol{\beta},\ I_2 = \boldsymbol{\beta} : \boldsymbol{\beta}, \qquad \mathrm{NN}_{\mathrm{k},\nu} = 0 \tag{36}$$



a) First training case. b) Validation case.

Fig. 4. Simulated (Sim) axial, σ_{11}, and shear, σ_{12}, stress responses compared to the artificial experimental data (Exp).

Table 1
Discovered equations for artificial training data.

Loss	Equation
NN_{iso}	
$1.2 \cdot 10^{2}$	–
$2.4 \cdot 10^{-2}$	p_1
$6.9 \cdot 10^{-3}$	$p_1 + p_2 \langle I_1 \rangle^2 \sqrt[3]{\langle I_2 \rangle} \langle I_3 \rangle^2$
$NN_{k,\beta}$	
$7.1 \cdot 10^{0}$	–
$9.0 \cdot 10^{-4}$	$p_1 \sqrt{\langle I_2 \rangle} \langle I_3 \rangle$
$6.1 \cdot 10^{-4}$	$p_1 \sqrt{\langle I_2 \rangle} \langle I_3 \rangle - p_2 \sqrt{\langle I_3 \rangle}$
$NN_{k,\nu}$	
$1.7 \cdot 10^{-4}$	–
$1.2 \cdot 10^{-4}$	$p_1 \langle -I_5 \rangle$
$2.1 \cdot 10^{-5}$	$p_1 \sqrt[3]{\langle I_1 \rangle} \sqrt[3]{\langle I_2 \rangle} \sqrt{\langle I_3 \rangle} + p_2 \sqrt[3]{\langle I_1 \rangle} \sqrt[3]{\langle I_3 \rangle}$

where p_1 and p_2 are constants that depend on the scaling used in the sparse regression. Since we only expect to find one term per network output, we just run the binomial search for up to two terms. Table 1 also includes the loss when using zero terms (expected for $NN_{k,\nu}$). The reported losses are the L2-losses divided by the number of samples and the square of the scaling factor for the network output. This scaling enables comparisons with the losses for experimental data in the next section.

For the NN_{iso} output, the loss reduces four orders of magnitude by using one constant term, which is also the expected final expression. Adding a second term does reduce the loss further, but only by a factor of 3.5. The loss also drops four orders of magnitude when adding a single term to the $NN_{k,\beta}$ output. The added term is equivalent to that in Eq. (36), noting that $I_2 = \boldsymbol{\beta} : \boldsymbol{\beta} \geq 0$, such that $\langle I_2 \rangle = I_2$. Finally, the loss for the $NN_{k,\nu}$ output is very low without any terms. Adding one or two terms reduces the loss only marginally. These findings show that the expressions in Eq. (36) have been identified.

4.2. Experimental data

Having demonstrated that the proposed procedure works for artificial data with a known solution, we apply it to uniaxial experimental data from Meyer and Ahlström (2023).

4.2.1. Model training

Similar to above, we choose five different cases: For training, we the use monotonic loading as well as the cyclic response for $\epsilon_{11} = \pm 0.25\,\%$ (first 50 cycles), $\epsilon_{11} = \pm 0.8\,\%$ (first 3 cycles), and $\epsilon_{11} = \pm 1.2\,\%$ (first 3 cycles). As validation, we use the first 25 cycles for $\epsilon_{11} = \pm 0.4\,\%$. The number of cycles are chosen so that we obtain stabilized hysteresis loops. The four different mini-batches consist of (1) the first five (out of 18) steps for the monotonic loading, (2) the first ten cycles for $\epsilon_{11} = \pm 0.25\,\%$, (3) the first cycle for $\epsilon_{11} = \pm 0.8\,\%$, and (4) the first cycle for $\epsilon_{11} = \pm 1.2\,\%$.

Fig. 5 shows the evolution of loss during the training process for the neural network and Chaboche models. For the Chaboche model, the training and validation losses decrease rapidly before remaining constant during each training phase. After the mini-batch training phase (first 5000 epochs), an additional slight decrease in the losses occur , resulting in a validation loss of $0.58\,\%$.

The training loss for the neural network model shows (on average) a continuous reduction in the loss, initially relatively rapid before slowing down. The validation loss for the neural network model reduces to an initial minimum within the first 50 epochs before increasing again during the following 250 epochs. Thereafter, it slowly decreases following the trend of the training loss in

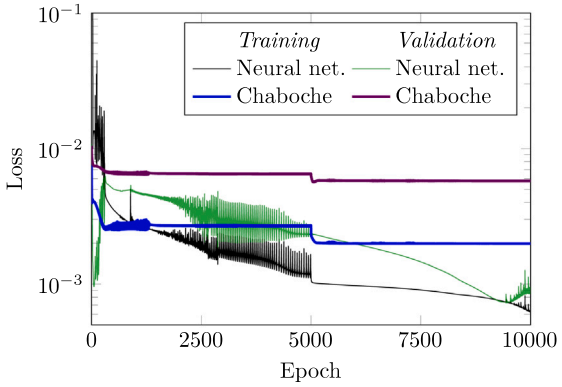


Fig. 5. Loss function evolution for model training on experimental data.

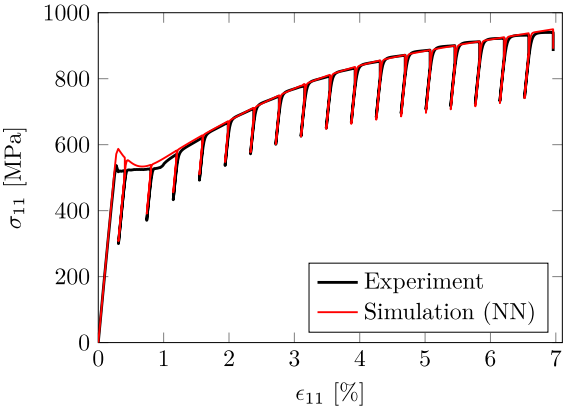


Fig. 6. Response of the neural network model for monotonic loading (training).

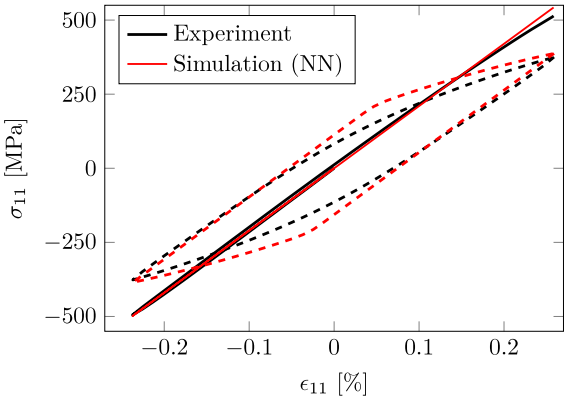


Fig. 7. Response of the neural network model in the first (solid) and 50th cycle (dashed) for $\epsilon = \pm 0.25\,\%$ (training).

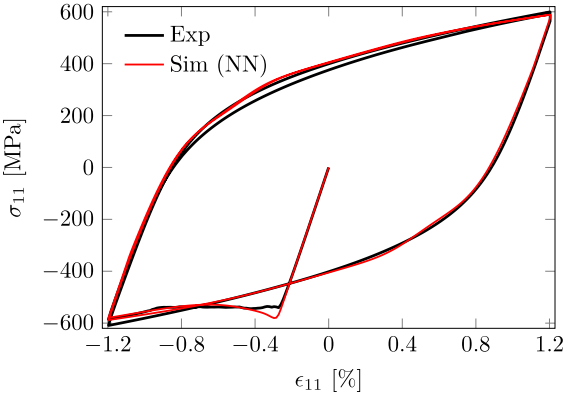


Fig. 8. Response of the neural network model in the two first cycles for $\epsilon = \pm 1.2\,\%$ (training).

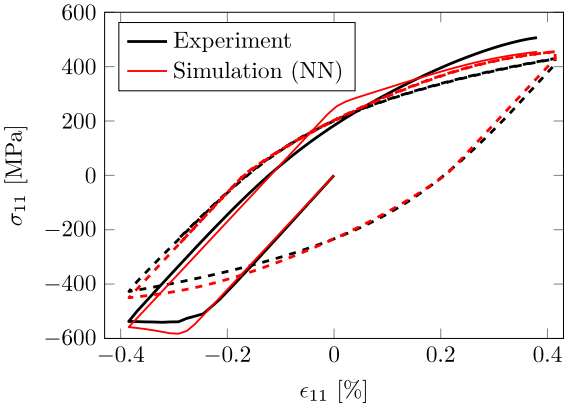


Fig. 9. Response of the neural network model in the first (solid) and 25th cycle (dashed) for$\epsilon = \pm 0.40\,\%$ (validation).

the mini-batch updates. In the full-batch updates, it reduces further before rising toward the end. This increase indicates overfitting, and the parameters from epoch 9467, which had the lowest validation loss (0.073 %), are used. This loss is almost one order of magnitude lower than that of the Chaboche model discussed above.

The material in Meyer and Ahlström (2023) exhibits a stress plateau followed by hardening. When analyzing the split between isotropic and kinematic hardening, a rapid decrease in isotropic yield strength combined with a corresponding increase in back-stress (Fig. 10 in Meyer and Ahlström, 2023). The trained material behavior approximates this behavior as well, seen by the hardening

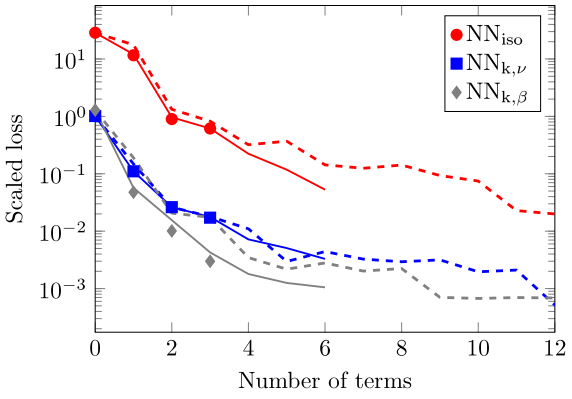


Fig. 10. Scaled loss for equation discovery using direct binomial search (markers), sparse regression followed by binomial (solid lines), and sparse regression with increasing penalty (dashed).

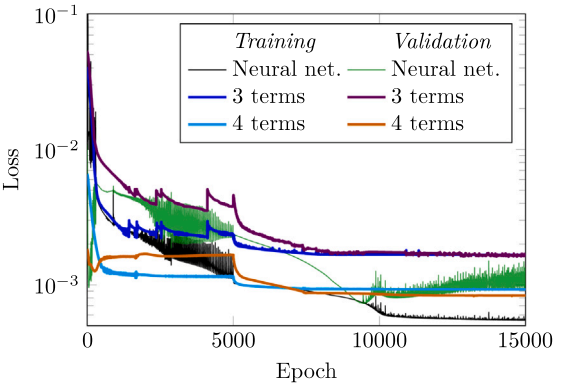


Fig. 11. Loss function evolution for training of identified models.

moduli, $H_{iso} \approx -162\,\text{GPa}$ and $H_{kin} \approx 131\,\text{GPa}$ (see Table 2), which approximately balance each other, as seen in Fig. 6. After the initial plateau phase, where the model shows too high stresses, the fit is almost perfect for the remaining loading.

The load case of $\epsilon_{11} = \pm 0.25\,\%$, Fig. 7, is hard to fit because the behavior is almost fully elastic in the initial cycle. Due to isotropic softening, the plastic strain amplitude increases in subsequent cycles, making the model very sensitive to the initial behavior. After training, this experiment has the highest relative loss of all cases, including the validation case. Despite these challenges, the model can obtain a reasonable loop shape in the 50th cycle. In the training case with $\epsilon_{11} = \pm 1.20\,\%$, Fig. 8, the overall loop shapes fit very well with experimental results. Even so, there is a slight waviness compared to the experimental curves.

For the validation case, in Fig. 9, there are some discrepancies between the model response and the experiment in the second half-cycle. The model shows a more distinct elastic-to-plastic transition compared to the experiments. However, in the 25th cycle, the stress–strain curves are very similar, indicating that the model works quite well for the validation case.

4.2.2. Equation discovery

The results in the previous section show that the neural network model can simulate the experimental results reasonably well. While this model could be used directly, it contains black-box elements for the evolution laws. Furthermore, it is less efficient than a regular material model due to the embedded networks. Instead, we identify sparse symbolic approximations of the network outputs. For 0–3 terms, we use a direct binomial search. This method finds the best solution since it tests all possible combinations. To be able to find expressions with more than three terms, we apply the selection method described in Section 3.2.3. We then run a binomial search on the selected candidate functions to identify up to 6 terms. As a final comparison, we also perform a direct sparse regression with a varying penalty factor to obtain expressions with different numbers of terms. The results in Fig. 10 show that (i) the direct binomial search always finds the lowest loss for the given number of terms, and (ii) that losses for sparse regression followed by the binomial search are only moderately higher. In most cases, the sparse regression followed by a binomial search leads to lower losses than simply increasing the penalty parameter until solutions with few enough terms are found. The variability in the sparse regression results highlights the issues with local minima discussed in Flaschel et al. (2021).

4.2.3. Discovered model

Based on the results in Fig. 10, we evaluate three models closely: (i) three terms from direct binomial search, (ii) four and (iii) six terms from sparse regression followed by a binomial search. Fig. 11 shows the training evolution for the two first cases compared to the neural network model training. Starting with the exact values for the identified parameters would yield a faster convergence, but all models start with a random initial guess to ensure a fair comparison. Using three identified terms for each network output slightly improves the training loss compared to the Chaboche model and greatly reduces the validation loss (from 0.58 % to 0.14 %). With four terms, the discovered model obtains training and validation losses comparable to the neural network model , both obtaining validation losses of 0.083 %. Using the identified model with six terms gives a marginal improvement: The training loss goes from 0.092 % to 0.080 %, and the validation loss from 0.083 % to 0.067 %. Such marginal improvements do not justify the increased model complexity, and Fig. 11 excludes these results for clarity.

As four terms seem required for an accurate model, the remaining discussion in this section considers this case, for which the evolution equation for isotropic hardening is

$$\dot{k} = \dot{\lambda}\left[1 + \frac{\kappa}{Y_0} + \kappa\left\langle p_{\text{iso},1}\,\kappa + p_{\text{iso},2}\,\boldsymbol{\nu} : \boldsymbol{\epsilon}_{\text{p}} + p_{\text{iso},3}\,\langle \boldsymbol{\nu} : \boldsymbol{\beta}\rangle + p_{\text{iso},4}\,\langle -\boldsymbol{\nu} : \boldsymbol{\beta}\rangle^2\right\rangle\right] \tag{37}$$

where $p_{\text{iso},i}$ are new material parameters. The last term within the outer brackets corresponds to $\kappa \text{NN}_{\text{iso}}$ in Eq. (14). Positive network output values, $\text{NN}_{\text{iso}} \geq 0$, used in the sparse regression, do not guarantee that the approximation of NN_{iso} is always positive. The

Table 2
Trained parameter values for each model.

	Neural Network model	Identified model (4 terms).	Chaboche model	Unit
Y_0	495.0	487.9	486.8	MPa
H_{iso}	−162.2	−159.9	−70.02	GPa
H_{kin}	131.4	114.3	33.51	GPa
t^*	51.87	55.26	71.31	ms
n	6.885	7.473	6.591	–

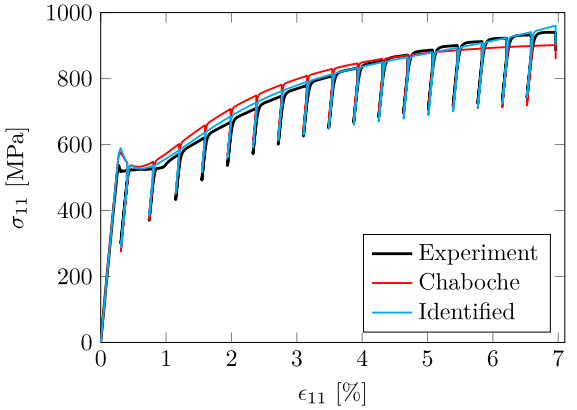


Fig. 12. Response of the Chaboche and identified model for monotonic loading (training).

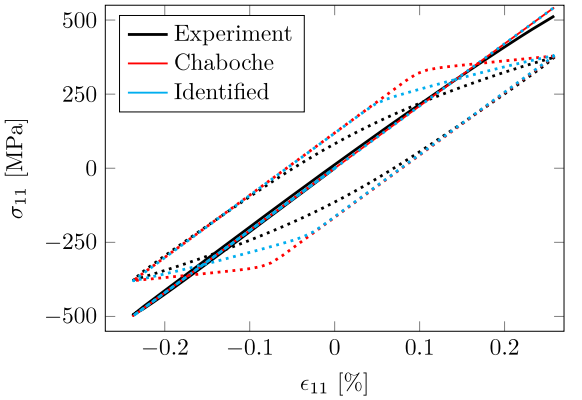


Fig. 13. Response of the Chaboche and identified model in the first (solid) and 50th cycle (dashed) for $\epsilon = \pm 0.25\,\%$ (training).

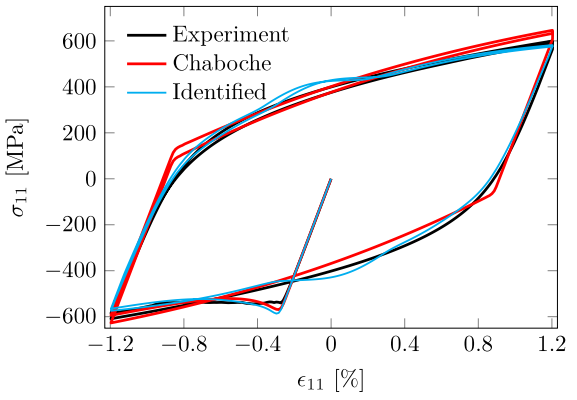


Fig. 14. Response of the Chaboche and identified model for the first two cycles with $\epsilon = \pm 1.2\,\%$ (training).

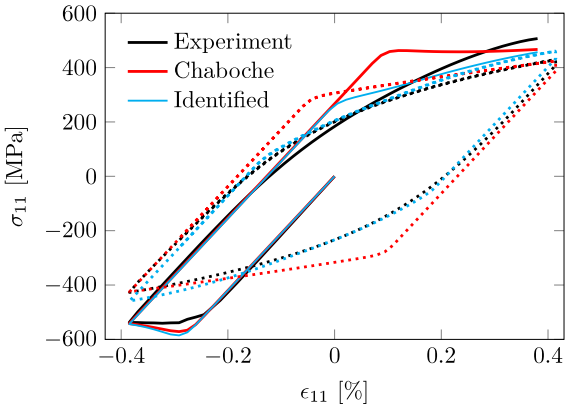


Fig. 15. Response of the Chaboche and identified model in the first (solid) and 25th cycle (dashed) for $\epsilon = \pm 0.40\,\%$ (validation).

expression corresponding to NN_{iso} is, therefore, wrapped in a Macaulay bracket. In addition to ensuring thermodynamic consistency, this addition improved the predictive ability of the model of the yield strength evolution in Section 4.2.4.

For kinematic hardening, the $\text{NN}_{\text{k},\nu}$ and $\text{NN}_{\text{k},\beta}$ parts have four terms each, with the complete evolution equation,

$$\dot{\boldsymbol{b}} = \dot{\lambda}\Bigg[-\boldsymbol{\nu} + [\boldsymbol{\nu} : \boldsymbol{\beta}]\left\langle p_{\text{k},\nu_1}\kappa + p_{\text{k},\nu_2}\kappa\,||\boldsymbol{\beta}||^2 + p_{\text{k},\nu_3}\,||\boldsymbol{\beta}|| + p_{\text{k},\nu_4}\sqrt[3]{\langle -\boldsymbol{\nu} : \boldsymbol{\beta}\rangle}\right\rangle\boldsymbol{\nu} + \left\langle p_{\text{k},\beta_1}[\boldsymbol{\beta} : \boldsymbol{\nu}]\,||\boldsymbol{\beta}||^2 + p_{\text{k},\beta_4}\sqrt{\langle \boldsymbol{\nu} : \boldsymbol{\epsilon}_{\text{p}}\rangle}\left|\left|\boldsymbol{\epsilon}_{\text{p}}\right|\right|^{2/3} + p_{\text{k},\beta_3}\langle \boldsymbol{\nu} : \boldsymbol{\epsilon}_{\text{p}}\rangle\left|\left|\boldsymbol{\epsilon}_{\text{p}}\right|\right|^2 + p_{\text{k},\beta_2}\langle \boldsymbol{\beta} : \boldsymbol{\nu}\rangle\right\rangle\boldsymbol{\beta}\Bigg] \tag{38}$$

using the same approach as in Eq. (37) to ensure a positive dissipation. Table 2 shows that all models have similar initial yield stress, Y_0, and viscoplastic parameters (t^* and n). While the hardening moduli differ between the models, especially for the Chaboche model, the initial hardening modulus, $H_{\text{iso}} + H_{\text{kin}}$ is within 15 GPa for all models.

As expected from the lower training loss, the identified model fits the monotonic stress–strain curve in Fig. 12 much better than the Chaboche model. However, compared to the neural network model in Fig. 6, the stress increase deviates more from the

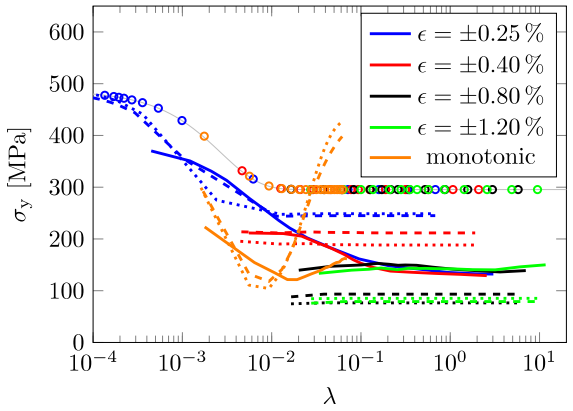


Fig. 16. Yield evolution at cycle peaks for different loading types versus accumulated plasticity, λ. The different line styles indicate experiments (solid lines), Chaboche model (markers), neural network (dotted), and identified model (dashed). The gray background line is the analytical Voce hardening with the Chaboche model's parameters.

experiment in the last two cycles. In the case of $\epsilon_{11} = \pm 0.25\,\%$ (Figs. 7 and 13), all three models behave practically fully elastic in the first cycle. In the 50th cycle, however, the neural network and identified models give a smoother elastic-to-plastic transition, much closer to the observed experimental behavior than the Chaboche model.

For the case of $\epsilon_{11} = \pm 1.20\,\%$ in Fig. 14, the identified is more accurate than the Chaboche model during the elastic-to-plastic transition and toward the end of each cycle. For this case, the neural network model, Fig. 8, exhibited a slight waviness, as previously discussed. This waviness is much more evident for the identified model, the causes for which are discussed in Section 5.3.

In the validation case with $\epsilon_{11} = \pm 0.40\,\%$ (Fig. 15), the identified model performs similar to the neural network model, with a more distinct elastic-to-plastic transition than the experiment in the second half-cycle, but with an excellent agreement in the 25th cycle. The Chaboche model cannot capture this smoother loop shape and predicts an almost bi-linear stress–strain curve for both cycles.

4.2.4. Yield evolution

Meyer and Ahlström (2023) measured the yield strength explicitly by probing. Due to the smooth elastic-to-plastic transition for the material, it is generally difficult to accurately detect the yield point directly from the stress–strain curve. Consequently, this is also difficult for the material model to learn based solely on stress–strain data. The primary motivation for the study in Meyer and Ahlström (2023) was to investigate the hypothesis that the isotropic hardening only depends on the accumulated plasticity, λ. Fig. 16 shows that the Chaboche model (circular markers), with Voce hardening, follows the expected gray line, $\sigma_y = Y_0 + \kappa_\infty[1 - \exp(-H_{\text{iso}}\lambda/\kappa_\infty)]$, for all different load cases. However, the experimental results show that the yield strength evolution depends on the loading case.

The neural network and the identified models have similar evolution behavior in all load cases. For $\epsilon_{11} = \pm 0.25\,\%$, the material softens initially before obtaining a constant yield stress. This evolution is qualitatively the same as the experimental results. All models underpredict the plasticity in the first cycles for this loading case, causing the curves to shift to the left compared to the experiments.

For $\epsilon_{11} = \pm 0.4\,\%$, the neural network and identified models correctly predict a somewhat lower yield strength than for $0.25\,\%$ but do not fully capture the continued softening in the first cycles down to the level of the higher strain amplitudes. For these amplitudes, however, lower yield strengths are correctly identified, as well as the constant level.

For the monotonic loading, the neural network and identified models capture the initial softening followed by hardening. While the models exaggerate this effect compared to the experiments, they could still qualitatively predict the yield strength behaviors from training on the stress–strain curves.

5. Discussion

5.1. Identification of known evolution laws

As previously discussed, Figs. 3 and 4 show that the neural network model can predict the response for hardening laws that the neural network cannot represent exactly. The results in Table 1 showed that the exact evolution laws could be re-identified by analyzing the trained neural network embedded in the material model. This test shows the potential of the proposed method.

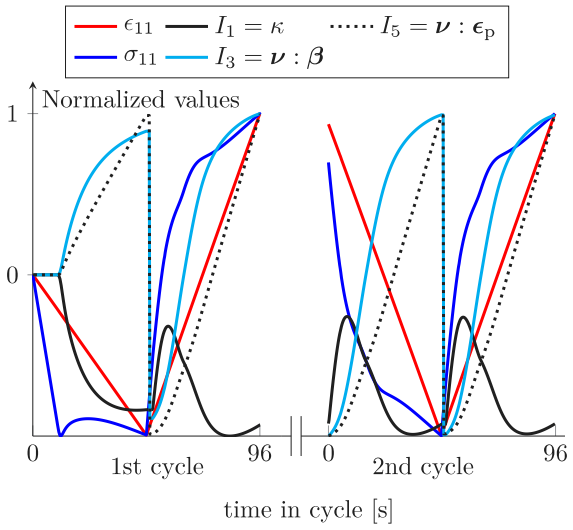


Fig. 17. Invariant evolution.

5.2. Capabilities of the neural network model

A sufficiently large feed-forward neural network is a universal function approximator that can describe any function (Hornik et al., 1989). From this point of view, one would expect that it should be possible to train the neural network model to fit the experimental data exactly, which the results show that it does not do. There are many possible explanations, some of which are discussed in the following paragraphs.

The network layout described in Section 2.3 is not a universal function approximator. That is not only due to an insufficient size but also because the last activation function prevents negative values. This constraint ensures fulfillment of the 2nd law of thermodynamics and is essential to ensure the physical correctness of the model. We have observed that it improves the model's predictive abilities, even though it increases the training loss slightly, as expected.

Using larger networks did not significantly improve the results, possibly due to the increased complexity of the non-convex optimization problem. However, this result may also indicate deficiencies in the model structure. For example, the standard model assumption of a well-separated elastic and plastic region is only valid for a homogeneous material. Actual test samples consist of a poly-crystalline material in which different grains yield at different stress levels, giving a smooth elastic-to-plastic transition upon load reversal. Modeling frameworks for rate-independent models with smooth elastic-to-plastic transitions were already proposed in the 90 s, by e.g. Lubliner et al. (1993). However, the smooth transition also occurs during reloading, see e.g. Lubliner et al. (1993), Rubin and Forest (2020), which is not the case for the results in Fig. 6. This transition remains a challenge, even with the very general evolution laws represented by the neural network, highlighting the need for alternative modeling frameworks.

5.3. Interpretation of the identified behavior

Overall, the identified evolution equation results in a reasonable behavior that is qualitatively similar to the experimental results. The main exception is the waviness observed in Fig. 14 for $\epsilon_{11} = \pm 1.20\,\%$. Fig. 17 shows the time evolution of three different invariants, the axial strain, and the axial stress for the two first cycles. All values are normalized by the maximum absolute value to visualize the evolution of quantities of different magnitudes. For comparison, the yield strengths shown in Fig. 16 are given by $Y_0 + \kappa$ at the end of each cycle.

In Fig. 17, $I_1 = \kappa$ decreases initially. After the load reverses, it increases rapidly before decreasing again. The remaining cycles follow this cyclic pattern of initial hardening followed by softening. The neural network model gives the same behavior. For the identified law, the term $\langle -\boldsymbol{\nu} : \boldsymbol{\beta} \rangle$ is responsible for the sudden hardening upon load reversal, which makes it possible to model the smooth elastic-to-plastic transition. After the peak of isotropic hardening, a waviness initiates. Many identified terms contain Macaulay brackets, resulting in discontinuous derivatives at zero, contributing to the continued waviness. However, in the remaining experiments and even in other parts of the $\epsilon_{11} = \pm 1.20\,\%$ experiment, the behavior is smooth.

While the Macaulay terms introduce some challenges in the model, they differentiate between loading cases. In particular, the difference in isotropic hardening for plastic loading and unloading in Fig. 17 seems to be the main factor in predicting the correct qualitative evolution of the yield strength in Fig. 16. Very few physical processes show abrupt changes in behavior: While it is reasonable to expect a different behavior directly upon load reversal, abrupt changes are unexpected when $\boldsymbol{\nu} : \boldsymbol{\epsilon}_p$ changes sign during monotonic loading. Smoother changes occur by considering, e.g., quadratic terms as $\langle \boldsymbol{\nu} : \boldsymbol{\epsilon}_p \rangle^2$. On the other hand, such terms will grow fast as $\boldsymbol{\epsilon}_p$ increases, leading to worse extrapolation. Nonlinear terms with sigmoidal shapes are attractive here (cf.

e.g. the tanh activation functions used in the neural network). While those functions are unsuitable for sparse regression, they are compatible with symbolic regression. Udrescu and Tegmark (2020) outperformed all available symbolic regression software for the physics problems in "Feynman Lectures on Physics" by first training a neural network on the data and then applying symbolic regression on the trained network. That approach aligns with the work presented herein and provides interesting future extensions.

5.4. Implications and opportunities for future studies

While the previous paragraph mentioned possibilities for improved equation discovery, the present section concerns how to improve the neural network model. The presented model starts from a basic plasticity model with isotropic and kinematic hardening. It is well known that plastic loading can result in both an anisotropic yield surface and stiffness degradation. These effects motivate extending the present model by introducing an evolving yield surface and damage evolution. While such extensions are straightforward, sufficient suitable experimental training data is required. A more straight-forward extension would be to consider finite strains following e.g. the framework in Meyer et al. (2018).

The presented work considers only homogenized behavior in both experiments and modeling. Even so, the same idea may be applied to, e.g., crystal plasticity models. However, stress–strain data for individual phases in heterogeneous microstructures are not readily available, and the training would require a finite element model to obtain the material data. One crucial concern for such inverse modeling is the identifiability of material responses for individual phases, grains, and interfaces.

The suitability of training data is also an important consideration when designing the experiments. The predicted yield strength evolution in Fig. 17 was unexpected for the authors. This finding offers essential information for designing experiments to understand yield evolution better. For example, varying the strain amplitude would provide more valuable data as the cyclic responses stabilize after only a few cycles.

5.5. Usage in finite element simulations

The proposed framework leads to material models with the same structure as standard material models for finite element simulations and may be used directly in existing finite element codes. However, the highly non-linear local problem does not permit large strain increments during the global equilibrium iterations. While testing the models in the finite element settings, we found that a good initial guess at the beginning of each time step ensured convergence, together with using sufficiently small time steps.

6. Contributions

In this paper, we present an approach that embeds a neural network within a conventional material modeling framework. Key advantages of our approach include the intrinsic satisfaction of thermodynamics, frame invariance, and the ability to employ implicit time integration. Moreover, we could discover analytical expressions for the evolution laws by learning from the trained neural network using both artificial and experimental data. The identified laws reveal an interaction between kinematic and isotropic hardening. These new interaction terms enabled the prediction of experimentally observed behaviors of yield strength evolution for various loading cases.

CRediT authorship contribution statement

Knut Andreas Meyer: Conceptualization, Methodology, Visualization, Formal analysis, Investigation, Software, Writing – original draft. **Fredrik Ekre:** Software, Investigation, Writing – original draft.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve the spelling, grammar and clarity of the abstract and contribution sections. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgments

The work in this paper has made use of several open source software packages written in the Julia programming language (Bezanson et al., 2017). The automatic differentiation used `ForwardDiff.jl` (Revels et al., 2016), and the implementation relied heavily on the `Tensors.jl` package (Carlsson and Ekre, 2019). The training used optimizers implemented in `Flux.jl`, Innes (2018).

Appendix. Sensitivity for parameter identification

Because neural networks have many parameters, a gradient-based optimization algorithm is desired. It requires the sensitivity of the loss, $L_k(\boldsymbol{p})$, in Eq. (32), with respect to the material parameters $\boldsymbol{p}$. As L_k is an explicit function of the simulated time history of stresses, it suffices to calculate the derivative of stress wrt. material parameters. We start by briefly reviewing the implementation of an elasto-plastic material model to make it easier to explain the various functions required to obtain the sought sensitivity.

A.1. Material response

A general strain-driven, history-dependent, material model, can be considered as two functions,[1]

$$\boldsymbol{\sigma} = m_\sigma(\boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t) \tag{39a}$$

$$\boldsymbol{s} = m_s(\boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t) \tag{39b}$$

where $\boldsymbol{s}$ denotes the hidden material state and Δt is the time increment. The notation ${}^n\bullet$ implies the value of $\bullet$ in the previous time step. For the models considered in this study, the state is given by $\boldsymbol{s} = [\boldsymbol{\epsilon}_\mathrm{p}, \kappa, \boldsymbol{\beta}]$.

A standard plasticity model implementation (excluding tangent stiffness calculation for brevity) consists of the following steps.

```
1  function material_response(p, ϵ, ⁿs, Δt)
2      σ_trial = elastic_response(p, ϵ, ⁿs)
3      Φ_trial = yield_criterion(p, σ_trial, ⁿs)
4      if Φ_trial <= 0 # Elastic response
5          return σ_trial, ⁿs
6      else # Plastic response
7          # Find X such that R(X, p, ϵ, ⁿs, Δt) = 0
8          X = solve_nonlinear(R, p, ϵ, ⁿs, Δt)
9          σ, s = plastic_response(X, p, ϵ, ⁿs, Δt)
10         return σ, s
11     end
12 end
```

The trial stress, $\boldsymbol{\sigma}^\mathrm{trial} = \mathbf{E} : \left[\boldsymbol{\epsilon} - {}^n\boldsymbol{\epsilon}_\mathrm{p}\right]$, on Line 2, is the stress assuming that the current load stress is elastic. If the trial state is inside the current yield surface, $\Phi \leq 0$, the response is elastic, and the trial stress is the correct solution (Line 5). Otherwise, the response is plastic, and the state variables and stress must be calculated by solving a nonlinear equation system, $\boldsymbol{R}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t)$ (cf. Box 7.1 in Neto et al., 2008 for a simplified example). Given the solution, $\boldsymbol{X}(\boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t)$, which is an implicit function via the nonlinear iterations, the updated state variables and stress can be calculated (Line 9).

To later calculate the sensitivity of the loss, $\mathrm{d}L_k/\mathrm{d}\boldsymbol{p}$, we will need the 6 sensitivities,

$$\frac{\mathrm{d}m_y}{\mathrm{d}z}, \quad \begin{matrix} y \in \{\boldsymbol{\sigma}, \boldsymbol{s}\} \\ z \in \{\boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \boldsymbol{p}\} \end{matrix} \tag{40}$$

The derivatives are trivial and efficient to compute for elastic steps with automatic differentiation. In the case of a plastic response, however, the implicit function $\boldsymbol{X}(\boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t)$ poses a problem. Therefore, we extend a standard approach for calculating the material tangent stiffness. Specifically, the residual function will always be zero for any input $\boldsymbol{p}$, $\boldsymbol{\epsilon}$, or ${}^n\boldsymbol{s}$, i.e.

$$\frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}z} = 0 = \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}z} + \frac{\partial \boldsymbol{R}}{\partial z} \tag{41}$$

With the function `plastic_response` on Line 9 denoted as $\hat{m}_y(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{\epsilon}, {}^n\boldsymbol{s}, \Delta t)$, the sought derivatives are

$$\frac{\mathrm{d}m_y}{\mathrm{d}z} = \frac{\mathrm{d}\hat{m}_y}{\mathrm{d}\boldsymbol{X}}\frac{\mathrm{d}\boldsymbol{X}}{\mathrm{d}z} + \frac{\partial \hat{m}_y}{\partial z} = -\frac{\mathrm{d}\hat{m}_y}{\mathrm{d}\boldsymbol{X}}\left[\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{X}}\right]^{-1}\frac{\partial \boldsymbol{R}}{\partial z} + \frac{\partial \hat{m}_y}{\partial z} \tag{42}$$

for plastic loading ($\Phi_\mathrm{trial} > 0$). The functions R and $\hat{m}_y$ are explicit (do not contain any iterations), and their derivatives can be efficiently calculated with automatic differentiation.

A.2. Stress state iterations

In most experiments, not all stress and strain components are measured. For the uniaxial and biaxial experiments considered in this study, one normal and one shear stress component and the corresponding strain components are available in the results. The two other normal stresses are zero, while the remaining shear strains are assumed to be zero. The corresponding normal strains are unknown, and so are the corresponding shear stresses (although these are typically zero). The measured strains are the simulation inputs, and the measured stresses are the outputs. In addition, we have the constraint that the other assumed values (normal stresses and shear strains) must be zero. Hence, from the modeling point of view, we have the known stress components,

[1] Although in practice calculated together in a single function.

$\tilde{\boldsymbol{\sigma}}$, and the corresponding unknown strain components, $\tilde{\boldsymbol{\epsilon}}$. In this paper, these components are $\bullet_{22}$ and $\bullet_{33}$. Additionally, we have the unknown stress components, $\hat{\boldsymbol{\sigma}}$, and corresponding known strain components, $\hat{\boldsymbol{\epsilon}}$ (components $\bullet_{11}$, $\bullet_{12}$, $\bullet_{13}$, and $\bullet_{23}$ in this study). As the input to the material function is the full strain tensor, an iterative procedure is required to find the $\tilde{\boldsymbol{\epsilon}}$ resulting in the prescribed $\tilde{\boldsymbol{\sigma}}$. Three different outputs are the result of such an iterative procedure,

$$\hat{\boldsymbol{\sigma}} = g_{\hat{\sigma}}\left(\hat{\boldsymbol{\epsilon}}, {}^{n}\boldsymbol{s}, \boldsymbol{p}, \tilde{\boldsymbol{\sigma}}\right) \tag{43a}$$

$$\boldsymbol{s} = g_{s}\left(\hat{\boldsymbol{\epsilon}}, {}^{n}\boldsymbol{s}, \boldsymbol{p}, \tilde{\boldsymbol{\sigma}}\right) \tag{43b}$$

$$\tilde{\boldsymbol{\epsilon}} = g_{\tilde{\epsilon}}\left(\hat{\boldsymbol{\epsilon}}, {}^{n}\boldsymbol{s}, \boldsymbol{p}, \tilde{\boldsymbol{\sigma}}\right) \tag{43c}$$

where typically $\hat{\boldsymbol{\sigma}}$ are the components considered in the loss, L_k in Eq. (32).

A similar trick as for the material response gives the derivative of the unknown stresses, $\hat{\boldsymbol{\sigma}}$, wrt. the material parameters: We use that the iterative function will always give the same known stresses, $\tilde{\boldsymbol{\sigma}}$, i.e.,

$$\frac{\mathrm{d}\tilde{\boldsymbol{\sigma}}}{\mathrm{d}\boldsymbol{p}} = \boldsymbol{0} = \frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\tilde{\boldsymbol{\epsilon}}}\frac{\mathrm{d}g_{\tilde{\epsilon}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\left[{}^{n}\boldsymbol{s}\right]}\frac{\mathrm{d}\left[{}^{n}g_{s}\right]}{\mathrm{d}\boldsymbol{p}} \tag{44}$$

allowing the derivative of the unknown strains to be calculated as

$$\frac{\mathrm{d}g_{\tilde{\epsilon}}}{\mathrm{d}\boldsymbol{p}} = -\left[\frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\tilde{\boldsymbol{\epsilon}}}\right]^{-1}\left[\frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\left[{}^{n}\boldsymbol{s}\right]}\frac{\mathrm{d}\left[{}^{n}g_{s}\right]}{\mathrm{d}\boldsymbol{p}}\right] \tag{45}$$

Given that derivative, the sensitivity of the unknown stresses, $\hat{\boldsymbol{\sigma}}$, can be calculated as

$$\frac{\mathrm{d}g_{\hat{\sigma}}}{\mathrm{d}\boldsymbol{p}} = \frac{\mathrm{d}m_{\hat{\sigma}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{\tilde{\sigma}}}{\mathrm{d}\tilde{\boldsymbol{\epsilon}}}\frac{\mathrm{d}g_{\tilde{\epsilon}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{\hat{\sigma}}}{\mathrm{d}\left[{}^{n}\boldsymbol{s}\right]}\frac{\mathrm{d}\left[{}^{n}g_{s}\right]}{\mathrm{d}\boldsymbol{p}} \tag{46}$$

However, from the last term in Eqs. (45) and (46), the value of $\mathrm{d}\left[{}^{n}g_{s}\right]/\mathrm{d}\boldsymbol{p}$, which stems from the previous time step, has yet to be discussed: In the first time step, this derivative is zero because the initial state is independent of the material parameters. Assuming that the value in the previous time step is known, the value in the current time step can be calculated as

$$\frac{\mathrm{d}g_{s}}{\mathrm{d}\boldsymbol{p}} = \frac{\mathrm{d}m_{s}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{s}}{\mathrm{d}\tilde{\boldsymbol{\epsilon}}}\frac{\mathrm{d}g_{\tilde{\epsilon}}}{\mathrm{d}\boldsymbol{p}} + \frac{\mathrm{d}m_{s}}{\mathrm{d}\left[{}^{n}\boldsymbol{s}\right]}\frac{\mathrm{d}\left[{}^{n}g_{s}\right]}{\mathrm{d}\boldsymbol{p}} \tag{47}$$

This result can then be used in the following time step to calculate $\mathrm{d}g_{\hat{\sigma}}/\mathrm{d}\boldsymbol{p}$, and finally $\mathrm{d}L_k/\mathrm{d}\boldsymbol{p}$.

For performance reasons, it is important to note that for each local iteration in the material model, i.e. solving $\boldsymbol{R}(\boldsymbol{X}, \boldsymbol{p}, \boldsymbol{\epsilon}, {}^{n}\boldsymbol{s}, \Delta t) = 0$, only the derivative $\partial\boldsymbol{R}/\partial\boldsymbol{X}$ needs to be calculated. This derivative is also required without sensitivity calculations. Furthermore, the algorithmic tangent stiffness, $\mathrm{d}m_{\sigma}/\mathrm{d}\boldsymbol{\epsilon}$, must be calculated once per stress state iteration. This calculation is also always needed. The additional computations for calculating the material sensitivities occur only at the end of each time step after the local material model iterations and the stress state iterations converge. This possibility reduces the computational costs of obtaining the sensitivity.

References

Abdolazizi, P., Linka, K., Cyron, C.J., 2023. Viscoelastic Constitutive Artificial Neural Networks (vCANNs) - a framework for data-driven anisotropic nonlinear finite viscoelasticity. preprint, arXiv:2303.12164, http://dx.doi.org/10.48550/arXiv.2303.12164.

Amos, B., Xu, L., Kolter, J.Z., 2017. Input convex neural networks: Supplementary material. In: 34th International Conference on Machine Learning, ICML 2017, Vol. 1. pp. 192–206.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017. Julia: A fresh approach to numerical computing. SIAM Rev. 59 (1), 65–98. http://dx.doi.org/10.1137/141000671.

Boehler, J.P., 1977. On irreducible representations for isotropic scalar functions. ZAMM - J. Appl. Math. Mech. / Z. Angew. Math. Mech. 57 (6), 323–327. http://dx.doi.org/10.1002/zamm.19770570608.

Borkowski, L., Sorini, C., Chattopadhyay, A., 2022. Recurrent neural network-based multiaxial plasticity model with regularization for physics-informed constraints. Comput. Struct. 258, 106678. http://dx.doi.org/10.1016/j.compstruc.2021.106678.

Brunton, S.L., Proctor, J.L., Kutz, J.N., Bialek, W., 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. USA 113 (15), 3932–3937. http://dx.doi.org/10.1073/pnas.1517384113.

Carlsson, K., Ekre, F., 2019. Tensors.jl- tensor computations in Julia. J. Open Res. Softw. 7 (1), 2–6. http://dx.doi.org/10.5334/jors.182.

Flaschel, M., Kumar, S., De Lorenzis, L., 2021. Unsupervised discovery of interpretable hyperelastic constitutive laws. Comput. Methods Appl. Mech. Engrg. 381, 113852. http://dx.doi.org/10.1016/j.cma.2021.113852, arXiv:2010.13496.

Flaschel, M., Kumar, S., De Lorenzis, L., 2022. Discovering plasticity models without stress data. Npj Comput. Mater. 8 (1), 91. http://dx.doi.org/10.1038/s41524-022-00752-4.

Flaschel, M., Kumar, S., De Lorenzis, L., 2023. Automated discovery of generalized standard material models with EUCLID. Comput. Methods Appl. Mech. Engrg. 405, 115867. http://dx.doi.org/10.1016/j.cma.2022.115867, URL: https://www.sciencedirect.com/science/article/pii/S0045782522008234.

Frederick, C.O., Armstrong, P.J., 2007. A mathematical representation of the multiaxial Bauschinger effect. Mater. High Temp. 24 (1), 1–26. http://dx.doi.org/10.1179/096034007X207589, URL: http://www.tandfonline.com/doi/abs/10.1179/096034007X207589.

Fuchs, A., Heider, Y., Wang, K., Sun, W., Kaliske, M., 2021. DNN 2 : A hyper-parameter reinforcement learning game for self-design of neural network based elasto-plastic constitutive descriptions. Comput. Struct. 249, 106505. http://dx.doi.org/10.1016/j.compstruc.2021.106505.

Fuhg, J.N., Hamel, C.M., Johnson, K., Jones, R., Bouklas, N., 2023. Modular machine learning-based elastoplasticity: Generalization in the context of limited data. Comput. Methods Appl. Mech. Engrg. 407, 115930. http://dx.doi.org/10.1016/j.cma.2023.115930.

Ghaboussi, J., Sidarta, D.E., 1998. New nested adaptive neural networks (NANN) for constitutive modeling. Comput. Geotech. 22 (1), 29–52. http://dx.doi.org/10.1016/S0266-352X(97)00034-7.

Haghighat, E., Abouali, S., Vaziri, R., 2023. Constitutive model characterization and discovery using physics-informed deep learning. Eng. Appl. Artif. Intell. 120 (September 2022), 105828. http://dx.doi.org/10.1016/j.engappai.2023.105828.

Heider, Y., Wang, K., Sun, W.C., 2020. SO(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. Comput. Methods Appl. Mech. Engrg. 363, 112875. http://dx.doi.org/10.1016/j.cma.2020.112875.
Holland, J.H., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications To Biology, Control, and Artificial Intelligence. The MIT Press, http://dx.doi.org/10.7551/mitpress/1090.001.0001.
Holzapfel, G.A., Ogden, R.W., 2010. Constitutive modelling of arteries. Proc. R. Soc. A 466 (2118), 1551–1597. http://dx.doi.org/10.1098/rspa.2010.0058.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366. http://dx.doi.org/10.1016/0893-6080(89)90020-8.
Huang, S., He, Z., Chem, B., Reina, C., 2022. Variational Onsager Neural Networks (VONNs): A thermodynamics-based variational learning strategy for non-equilibrium PDEs. J. Mech. Phys. Solids 163 (3), 1–26. http://dx.doi.org/10.1016/j.jmps.2022.104856.
Innes, M., 2018. Flux: Elegant machine learning with julia. J. Open Source Softw. http://dx.doi.org/10.21105/joss.00602.
Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E., 2020. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. J. Comput. Phys. 404, 109136. http://dx.doi.org/10.1016/j.jcp.2019.109136.
Keskar, N.S., Nocedal, J., Tang, P.T.P., Mudigere, D., Smelyanskiy, M., 2017. On large-batch training for deep learning: Generalization gap and sharp minima. In: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings. pp. 1–16. http://dx.doi.org/10.48550/ARXIV.1609.04836, arXiv:1609.04836.
Kingma, D.P., Ba, J.L., 2015. Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. pp. 1–15. http://dx.doi.org/10.48550/arXiv.1412.6980.
Kirchdoerfer, T., Ortiz, M., 2016. Data-driven computational mechanics. Comput. Methods Appl. Mech. Engrg. 304, 81–101. http://dx.doi.org/10.1016/j.cma.2016.02.001.
Koeppe, A., Bamer, F., Selzer, M., Nestler, B., Markert, B., 2022. Explainable artificial intelligence for mechanics: Physics-explaining neural networks for constitutive models. Front. Mater. 8 (February), 1–16. http://dx.doi.org/10.3389/fmats.2021.824958.
Lee, Y., Hsieh, W., Huang, C., 2005. epsilon-SSVR: A smooth support vector machine for epsilon-insensitive regression. IEEE Trans. Knowl. Data Eng. 17 (05), 678–685. http://dx.doi.org/10.1109/TKDE.2005.77.
Lemaitre, J., Chaboche, J.-L., 1990. Mechanics of Solid Materials. Cambridge University Press, Cambridge, http://dx.doi.org/10.1017/CBO9781139167970.
Linden, L., Klein, D.K., Kalina, K.A., Brummund, J., Weeger, O., Kästner, M., 2023. Neural networks meet hyperelasticity: A guide to enforcing physics. J. Mech. Phys. Solids 179 (6), 105363. http://dx.doi.org/10.1016/j.jmps.2023.105363.
Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J., 2020. On the variance of the adaptive learning rate and beyond. In: International Conference on Learning Representations. pp. 1–13, URL: https://openreview.net/forum?id=rkgz2aEKDr.
Lubarda, V.A., 2008. On the Gibbs conditions of stable equilibrium, convexity and the second-order variations of thermodynamic potentials in nonlinear thermoelasticity. Int. J. Solids Struct. 45 (1), 48–63. http://dx.doi.org/10.1016/j.ijsolstr.2007.07.010.
Lubliner, J., Taylor, R.L., Auricchio, F., 1993. A new model of generalized plasticity and its numerical implementation. Int. J. Solids Struct. 30 (22), 3171–3184. http://dx.doi.org/10.1016/0020-7683(93)90146-X.
Malik, A., Abendroth, M., Hütter, G., Kiefer, B., 2022. A hybrid approach employing neural networks to simulate the elasto-plastic deformation behavior of 3D-Foam Structures. Adv. Eng. Mater. 24, 210064112. http://dx.doi.org/10.1002/adem.202100641.
Marino, E., Flaschel, M., Kumar, S., De Lorenzis, L., 2023. Automated identification of linear viscoelastic constitutive laws with EUCLID. Mech. Mater. 104643. http://dx.doi.org/10.1016/j.mechmat.2023.104643.
Masi, F., Stefanou, I., Vannucci, P., Maffi-Berthier, V., 2021. Thermodynamics-based Artificial Neural Networks for constitutive modeling. J. Mech. Phys. Solids 147 (October 2020), 104277. http://dx.doi.org/10.1016/j.jmps.2020.104277.
Maugin, G.A., 1992. Convexity. In: The Thermomechanics of Plasticity and Fracture. Cambridge University Press, Cambridge, pp. 283–292. http://dx.doi.org/10.1017/CBO9781139172400, chapter App. 2.
Meyer, K.A., 2020. Evaluation of material models describing the evolution of plastic anisotropy in pearlitic steel. Int. J. Solids Struct. 200–201, 266–285. http://dx.doi.org/10.1016/j.ijsolstr.2020.04.037.
Meyer, K.A., Ahlström, J., 2023. The role of accumulated plasticity on yield surface evolution in pearlitic steel. Mech. Mater. 179, 104582. http://dx.doi.org/10.1016/j.mechmat.2023.104582.
Meyer, K.A., Ekh, M., Ahlström, J., 2018. Modeling of kinematic hardening at large biaxial deformations in pearlitic rail steel. Int. J. Solids Struct. 130–131, 122–132. http://dx.doi.org/10.1016/j.ijsolstr.2017.10.007.
Meyer, K.A., Menzel, A., 2021. A distortional hardening model for finite plasticity. Int. J. Solids Struct. 232, 111055. http://dx.doi.org/10.1016/j.ijsolstr.2021.111055.
Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., Bessa, M.A., 2019. Deep learning predicts path-dependent plasticity. Proc. Natl. Acad. Sci. USA 116 (52), 26414–26420. http://dx.doi.org/10.1073/pnas.1911815116.
Neto, E.d.S., Peric, D., Owen, D.R.J., 2008. Computational Methods for Plasticity: Theory and Applications. John Wiley & Sons.
Ohno, N., Wang, J.-D., 1993. Kinematic hardening rules with critical state of dynamic recovery, part I: formulation and basic features for ratchetting behavior. Int. J. Plast. 9 (3), 375–390. http://dx.doi.org/10.1016/0749-6419(93)90042-O.
Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks. In: Dasgupta, S., McAllester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, Vol. 28, PMLR, Atlanta, Georgia, USA, pp. 1310–1318, URL: https://proceedings.mlr.press/v28/pascanu13.html.
Revels, J., Lubin, M., Papamarkou, T., 2016. Forward-mode automatic differentiation in Julia. arXiv:1607.07892 [cs.MS], URL: https://arxiv.org/abs/1607.07892.
Rosenkranz, M., Kalina, K.A., Brummund, J., Kästner, M., 2023. A comparative study on different neural network architectures to model inelasticity. Internat. J. Numer. Methods Engrg. 1–39. http://dx.doi.org/10.1002/nme.7319.
Rubin, M.B., Forest, S., 2020. Analysis of material instability of a smooth elastic-inelastic transition model. Int. J. Solids Struct. 193–194, 39–53. http://dx.doi.org/10.1016/j.ijsolstr.2020.01.023.
Taç, V., Rausch, M.K., Sahli Costabal, F., Tepole, A.B., 2023. Data-driven anisotropic finite viscoelasticity using neural ordinary differential equations. Comput. Methods Appl. Mech. Engrg. 411, 116046. http://dx.doi.org/10.1016/j.cma.2023.116046.
Tieleman, T., Hinton, G., 2012. Lecture 6.5: Rmsprop: divide the gradient by a running average of its recent magnitude. URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
Udrescu, S.M., Tegmark, M., 2020. AI Feynman: A physics-inspired method for symbolic regression. Sci. Adv. 6 (16), http://dx.doi.org/10.1126/sciadv.aay2631.
Vlassis, N.N., Sun, W.C., 2021. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. Comput. Methods Appl. Mech. Engrg. 377, 113695. http://dx.doi.org/10.1016/j.cma.2021.113695.
Wen, J., Zou, Q., Wei, Y., 2021. Physics-driven machine learning model on temperature and time-dependent deformation in lithium metal and its finite element implementation. J. Mech. Phys. Solids 153 (February), 104481. http://dx.doi.org/10.1016/j.jmps.2021.104481.
Xiao, Y., Chen, J., Cao, J., 2012. A generalized thermodynamic approach for modeling nonlinear hardening behaviors. Int. J. Plast. 38, 102–122. http://dx.doi.org/10.1016/j.ijplas.2012.05.004.