

Machine learning-accelerated time integration of plasticity models

Downloaded from: https://research.chalmers.se, 2026-11-11 12:49 UTC

Citation for the original published paper (version of record):
Talebi, N., Ekh, M., Meyer, K. (2027). Machine learning-accelerated time integration of plasticity models. Computer Methods in Applied Mechanics and Engineering, 463(part B). http://dx.doi.org/10.1016/j.cma.2026.119445

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

Computer Methods in Applied Mechanics and Engineering 463 (2027) 119445

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Machine learning-accelerated time integration of plasticity models

Nasrin Talebi *, Magnus Ekh, Knut Andreas Meyer

Division of Computational Mechanics and Materials Engineering, Department of Mechanical Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden

ARTICLE INFO

Keywords:
Cyclic plasticity modeling
Explicit time integration
Machine learning
Neural networks
Finite element simulations

ABSTRACT

Finite element simulations of structures with nonlinear material behavior require advanced material models to provide accurate predictions. However, the computational costs of these models can be high, as they solve coupled differential algebraic equations at each integration point, in each equilibrium iteration, in every time step. In this study, we propose a machine learning-based framework to accelerate these computations by explicitly calculating the state variable updates with neural networks, enabling large time steps with low computational costs. The neural networks operate on invariants, and the necessary and sufficient evolution directions are determined analytically based on training data. Furthermore, the proposed framework enforces exact fulfillment of the plastic consistency condition. To evaluate the proposed framework, two prototype material models based on the von Mises yield criterion and different nonlinear kinematic hardening laws are chosen. Only 10 cycles of two-dimensional multiaxial proportional loading are used to generate the training and validation data, while 100 cycles of three-dimensional loading are used as test data. After evaluating the proposed framework in material point simulations, we incorporate it into finite element simulations to assess its accuracy and computational efficiency in a boundary value problem. The results from both material point and finite element simulations show very promising numerical performance of the neural network-based time integrator. It achieves high accuracy and numerical stability, generalizes from two-dimensional training data to three-dimensional cyclic loading over longer loading histories, and provides a noticeable reduction in computational time for a single strain increment per load segment.

1. Introduction

To accurately predict the mechanical response of structural components in finite element (FE) simulations, advanced nonlinear material models are essential across a wide range of engineering applications. Some examples include the investigation of rolling contact fatigue in railway rails [1–4], geotechnical and soil simulations [5], and sheet metal forming processes [6]. Despite their predictive capabilities, nonlinear constitutive models remain computationally costly within the FE framework, since a set of Differential Algebraic Equations (DAEs) needs to be solved in every integration point in the FE mesh, for every iteration in each time step of the simulation. Therefore, efficient material model algorithms are crucial for computational efficiency. Explicit forward Euler integration of the DAEs results in fast material models, but requires very small time steps to be numerically robust [7]. Implicit backward Euler integration is often preferred due to its unconditional stability, which allows for larger time steps. However, many time steps are still required per loading cycle, and a nonlinear equation system must usually be solved iteratively. The challenge is particularly severe in cyclic simulations where no stabilized behavior is obtained [8,9]. To accelerate simulations with nonlinear

* Corresponding author.
E-mail address: nasrin.talebi@chalmers.se (N. Talebi).

https://doi.org/10.1016/j.cma.2026.119445
Received 2 June 2026; Received in revised form 27 August 2026; Accepted 19 September 2026

0045-7825/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

material models, this study proposes a machine learning (ML) based explicit time integrator that accurately integrates the DAEs while taking much larger time steps than existing methods.

Recent studies have demonstrated that employing ML algorithms, such as Neural Networks (NNs), can improve the accuracy of constitutive modeling; see, e.g., the reviews on machine learning in constitutive modeling by Fuhg et al. [10] and Dornheim et al. [11]. In early ML-based constitutive models, NNs were trained to approximate the mapping from strain, strain history, or state variables to stress or state variable increment [12–14]. However, in general, these approaches have low predictive accuracy beyond the training regime, as they do not account for the underlying physics [15]. To address this, several approaches have been proposed in the literature to satisfy physical principles in a weak form [16–19] or by construction [20,21].

Recently, ML-based methods have also been employed to accelerate the numerical integration of plasticity models, where the local solution of nonlinear evolution equations at each integration point can contribute significantly to the computational cost. Zhang and Mohr [22] reformulated the classical return-mapping algorithm for von Mises plasticity by rewriting the stress update using a single nonlinear function replaced by an NN. The NN estimates the apparent modulus without imposing a priori assumptions regarding the yield surface, flow rule, or hardening law. In [23], the stress integration algorithm during plastic loading was replaced by an NN operating in principal stress space for plane stress von Mises plasticity under monotonic and cyclic loading. The approach was subsequently extended to anisotropic plasticity for sheet metal forming applications [24]. Dettmer et al. [25] presented a stress-update framework in which the state variable update and stress prediction are separated and represented by state and response NNs, respectively. The approach was evaluated for uniaxial elastoplasticity, a uniaxial elastoplastic damage model, and plane strain elastoplasticity.

In contrast to the above approaches, which learn plasticity constraints primarily from data, several frameworks have also been proposed to satisfy physical constraints, while alleviating or eliminating the need to iteratively solve nonlinear constitutive equations. Using the idea of Physics-Informed Neural Networks (PINNs) [26], Eghbalian et al. [27] developed a surrogate model for elastoplasticity in which some features of classical elastoplasticity are enforced through the loss function. Rezaei et al. [28] proposed an unsupervised PINN-based approach to obtain solutions of nonlinear constitutive equations without repeated Newton iterations. Instead of purely surrogate-based models, Li et al. [29] introduced a PINN-driven framework that circumvents only the local stress integration in an elastoplastic soil constitutive model. The consistency condition error during training was incorporated into the loss function. In an unsupervised learning framework, Zhang [30] employed an NN to predict the stress increment during plastic loading, with residuals of the stress update and consistency condition included in the loss function. To approximate the solution of a local scalar nonlinear equation, Malleval et al. [31] employed NNs either as a surrogate or to provide an initial guess for the Newton–Raphson method in an elastoviscoplastic material model.

The key distinction of the present framework is that NNs are employed to construct an explicit time integrator for the evolution of the state variables, while retaining essential parts of the plasticity formulation in analytical form. The state variable increments are expressed in terms of invariant-dependent scalar functions and a suitable set of evolution directions inferred from training data. The distinction between elastic and plastic response is enforced by construction, and the plastic consistency condition is fulfilled exactly through an analytical correction factor. Hence, these fundamental plasticity constraints are not learned by the NNs or imposed weakly through the loss function. The objective of the NN-based time integrator is to retain the low computational cost of explicit integration while allowing substantially larger strain increments.

In the present study, we evaluate the numerical performance of the proposed NN-based time integration scheme by applying it to the DAEs of two plasticity models with kinematic hardening laws of different levels of complexity. After training on 10 two-dimensional (2D) loading cycles, the generalization capability of the proposed method is assessed considering 100 three-dimensional (3D) loading cycles. Finally, the method is implemented in FE simulations to evaluate its accuracy and computational efficiency.

The paper is organized as follows: In Section 2, we present the general formulation for explicit NN-based time integration of a generic rate-independent plasticity model. This is followed by the description of the adopted prototype models and the corresponding formulation of the NN-based time integration scheme in Section 3. We explain the considered FFNNs and the data generation strategy in Section 4. Finally, in Sections 5 and 6, we discuss and summarize our findings.

2. NN-accelerated time integration of a generic plasticity model

In this section, we first present the generic formulation of a thermodynamically consistent material model in a small-strain setting, followed by a description of the time integration schemes for this model. Finally, we propose an explicit NN-accelerated time integrator for solving the evolution equations of state variables.

The notation for different tensors is as follows: Second-order tensors are boldface, e.g. $\boldsymbol{t}$, fourth-order tensors are boldface and upright, e.g. $\mathbf{T}$. Sets of variables are written with blackboard bold, e.g. $\mathbb{T}$. Finally, $\bullet^{\mathrm{dev}} := \bullet - \mathrm{tr}(\bullet)\boldsymbol{I}/3$ denotes the deviatoric part of the tensor $\bullet$.

2.1. Generic small-strain plasticity model

The starting point for a generic small-strain, rate-independent, plasticity model is the Helmholtz's free-energy function, Ψ, that depends on the total strain, $\boldsymbol{\epsilon}$, and the set of N_{s} state variables, $\mathbb{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{N_{\mathrm{s}}}\}$,

$$\Psi = \Psi(\boldsymbol{\epsilon}, \mathbb{S}) \tag{1}$$

The stress, $\boldsymbol{\sigma}$, and the (hardening) variables, $\mathbb{A} = \{\mathfrak{a}_1,\ \mathfrak{a}_2,\ \dots,\ \mathfrak{a}_{N_\mathrm{a}}\}$, are defined as

$$\boldsymbol{\sigma}(\boldsymbol{\epsilon}, \mathbb{S}) = \frac{\partial \Psi}{\partial \boldsymbol{\epsilon}}, \quad \mathfrak{a}_i(\boldsymbol{\epsilon}, \mathbb{S}) = -\frac{\partial \Psi}{\partial \mathfrak{s}_i}$$

Next, a yield function, $\Phi(\boldsymbol{\sigma}, \mathbb{A})$, that defines the elastic domain ($\Phi < 0$) and the plastic domain ($\Phi = 0$), is formulated. By introducing the plastic multiplier, $\dot{\lambda}$, the Karush–Kuhn–Tucker (KKT) loading/unloading conditions are expressed as

$$\Phi \le 0, \quad \dot{\lambda} \ge 0, \quad \Phi\, \dot{\lambda} = 0 \tag{2}$$

The state variables only evolve during plastic loading, which are described by the generic evolution laws

$$\dot{\mathfrak{s}}_i = \dot{\lambda}\, \mathfrak{f}_i\,(\boldsymbol{\sigma}(\boldsymbol{\epsilon}, \mathbb{S}),\ \mathbb{A}(\boldsymbol{\epsilon}, \mathbb{S})) \tag{3}$$

2.2. Integration of a generic small-strain plasticity model

In an incremental strain-controlled algorithm, the role of a material model is to calculate the current stress, $\boldsymbol{\sigma}$, and the updated state variables, $\mathbb{S}$. Here, we assume that the strain increment (from time ${}^n t$ to t), $\Delta\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - {}^n\boldsymbol{\epsilon}$, is linear on a material point level, i.e.,

$$\boldsymbol{\epsilon}(\theta) = {}^n\boldsymbol{\epsilon} + \theta\, \Delta\boldsymbol{\epsilon} \tag{4}$$

for $\theta \in [0, 1]$ within the time step.[1] For a given $\Delta\boldsymbol{\epsilon}$, the standard procedure is to assume an elastic response and calculate the trial stress

$$\boldsymbol{\sigma}^{\mathrm{tr}} := \boldsymbol{\sigma}(\boldsymbol{\epsilon}, {}^n\mathbb{S}) \tag{5}$$

The assumption is true if $\Phi^{\mathrm{tr}} := \Phi(\boldsymbol{\sigma}^{\mathrm{tr}}, {}^n\mathbb{A}) < 0$, resulting in $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{tr}}$ and $\mathbb{S} = {}^n\mathbb{S}$; otherwise, the material model must solve the set of DAEs in Eqs. (2) and (3). Different time integration schemes, $\mathfrak{g}_i^{\mathrm{TI}}$, can be used to approximate the solution to Eq. (3)

$$\Delta\mathfrak{s}_i = \mathfrak{g}_i^{\mathrm{TI}}({}^n\boldsymbol{\epsilon},\ \Delta\boldsymbol{\epsilon},\ {}^n\mathbb{S}) \tag{6}$$

For implicit algorithms, $\mathfrak{g}_i^{\mathrm{TI}}$ typically cannot be described analytically, but requires solving a nonlinear equation system iteratively [32]. In explicit algorithms, upon transition from elastic to plastic behavior, the intersection of the trial stress path with the yield surface is identified to determine the plastic part of the applied strain increment [33,34]. This introduces the elastic limit stress, $\boldsymbol{\sigma}^{\mathrm{lim}}$, as

$$\boldsymbol{\sigma}^{\mathrm{lim}} = \boldsymbol{\sigma}\,({}^n\boldsymbol{\epsilon} + s\,\Delta\boldsymbol{\epsilon}, {}^n\mathbb{S}) \tag{7}$$

where the scalar variable $s \in [0, 1]$ is determined from $\Phi(\boldsymbol{\sigma}^{\mathrm{lim}}, {}^n\mathbb{A}) = 0$. During this elastic substep, no evolution of state variables occurs as $\Phi < 0$. Accordingly, $\mathfrak{g}_i^{\mathrm{TI}}$ only acts on the plastic loading part as

$$\Delta\mathfrak{s}_i = \mathfrak{g}_i^{\mathrm{TI}}({}^n\boldsymbol{\epsilon} + s\Delta\boldsymbol{\epsilon},\ [1 - s]\Delta\boldsymbol{\epsilon},\ {}^n\mathbb{S}) \tag{8}$$

For the remaining strain increment, the KKT condition (Eq. (2)) is replaced with $\Phi = 0$, which, coupled with Eq. (3), forms the DAEs to be solved to calculate $\Delta\mathfrak{s}_i$. The solution for the increment of the plastic multiplier, $\Delta\lambda$, is typically obtained using an iterative procedure for implicit integration, whereas it can be obtained analytically for explicit integration from $\Phi(\boldsymbol{\sigma}, \mathbb{A}) = 0$. This makes explicit time integrators numerically efficient, but they require small time steps to ensure numerical stability.

2.3. NN-accelerated time integration of a generic small-strain plasticity model

The idea of this paper is to construct an explicit time integrator using NNs, $\mathfrak{g}_{\mathfrak{s}_i}^{\mathrm{NN}}$, that can handle large strain increments while maintaining accuracy and numerical stability. We formulate $\mathfrak{g}_{\mathfrak{s}_i}^{\mathrm{NN}}$ for each state variable, $\mathfrak{s}_i$, as the sum of contributions associated with $\boldsymbol{\sigma}^{\mathrm{tr}}$, $\boldsymbol{\sigma}^{\mathrm{lim}}$, and ${}^n\mathfrak{a}_j$. Each contribution is expressed as a linear combination of the scalar outputs, $\mathcal{NN}$, of a neural network and the derivatives of a set of suitable invariants $\mathbb{I}(\boldsymbol{\sigma}^{\mathrm{tr}}, \boldsymbol{\sigma}^{\mathrm{lim}}, {}^n\mathfrak{a}_j) = \{I_1,\ I_2,\ \cdots,\ I_N\}$ with respect to $\boldsymbol{\sigma}^{\mathrm{tr}}$, $\boldsymbol{\sigma}^{\mathrm{lim}}$, and ${}^n\mathfrak{a}_j$. The derivatives define the evolution directions for each $\mathfrak{s}_i$. Accordingly, $\mathfrak{g}_{\mathfrak{s}_i}^{\mathrm{NN}}$ is formulated as

$$\mathfrak{g}_{\mathfrak{s}_i}^{\mathrm{NN}}({}^n\boldsymbol{\epsilon} + s\Delta\boldsymbol{\epsilon},\ [1 - s]\Delta\boldsymbol{\epsilon},\ {}^n\mathbb{S}) = \sum_{k=1}^{N}\left[\mathcal{NN}_{\mathfrak{s}_i,1,k}(\mathbb{I})\frac{\partial I_k}{\partial \boldsymbol{\sigma}^{\mathrm{tr}}} + \mathcal{NN}_{\mathfrak{s}_i,2,k}(\mathbb{I})\frac{\partial I_k}{\partial \boldsymbol{\sigma}^{\mathrm{lim}}} + \sum_{j=1}^{N_\mathrm{a}}\mathcal{NN}_{\mathfrak{s}_i,j+2,k}(\mathbb{I})\frac{\partial I_k}{\partial {}^n\mathfrak{a}_j}\right] \tag{9}$$

The choice to express the NNs in invariants eliminates coordinate system dependence and improves generalizability.

When applying the proposed time integration scheme, the increments of the state variables are computed as

$$\Delta\mathfrak{s}_i = c\ \mathfrak{g}_{\mathfrak{s}_i}^{\mathrm{NN}} \tag{10}$$

where we introduce the correction factor, c, to ensure that $\Phi(\boldsymbol{\sigma}, \mathbb{A}) = 0$ is satisfied exactly during plastic loading.

[1] For higher order time integration on the global finite element level, $\boldsymbol{\epsilon}$ may vary non-linearly during the increment, but such higher order space–time finite element formulations are not considered herein.

Table 1
Material model parameter values for the chosen prototype plasticity models. The parameters E, G, Y_0, H_{kin}, and β_∞ are common to both models, while m applies only to the OW-based material model.

Material parameters	E	G	Y_0	H_{kin}	β_∞	m
	200	75	400	150	500	2
Unit	GPa	GPa	MPa	GPa	MPa	–

3. Prototype plasticity models with kinematic hardening

This section first introduces the two prototype plasticity models considered for the training, validation, and testing of the proposed NN-based time integration scheme. A detailed description of the NN-based time integration is presented thereafter.

3.1. Prototype plasticity models

Considering the generic formulation of a small-strain plasticity model explained in Section 2.1, the prototype plasticity models assume linear isotropic elasticity and adopt kinematic hardening. The set of state variables for each model is $\mathbb{S} = \{\boldsymbol{\epsilon}^{\text{p}}, \boldsymbol{b}\}$, where $\boldsymbol{\epsilon}^{\text{p}}$ is the plastic strain and $\boldsymbol{b}$ is the state variable associated with kinematic hardening. The free-energy density function, Ψ, is formulated as

$$\Psi(\boldsymbol{\epsilon}, \mathbb{S}) = \frac{1}{2}\left[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{\text{p}}\right] : \mathbf{E}^{\text{e}} : \left[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{\text{p}}\right] + \frac{1}{3} H_{\text{kin}}\, \boldsymbol{b} : \boldsymbol{b} \tag{11}$$

where H_{kin} is the kinematic hardening modulus. Isotropic elasticity is considered such that the elasticity tensor, $\mathbf{E}^{\text{e}}$, is formulated as

$$\mathbf{E}^{\text{e}} = 2\,G\,\mathbf{I}^{\text{dev}} + K_{\text{b}}\,\boldsymbol{I} \otimes \boldsymbol{I} \quad \text{where} \quad K_{\text{b}} = \frac{E\,G}{3\,(3\,G - E)} \tag{12}$$

G and E are the shear and Young's moduli, respectively. $\mathbf{I}^{\text{dev}} = \mathbf{I} - \boldsymbol{I} \otimes \boldsymbol{I}/3$ is the 4th order deviatoric identity tensor, where $\boldsymbol{I}$ and $\mathbf{I}$ are the second- and fourth-order identity tensors. The stress, $\boldsymbol{\sigma}$, and the (hardening) variables, $\mathbb{A} = \{\boldsymbol{\sigma}^{\text{p}}, \boldsymbol{\beta}\}$, are obtained as

$$\boldsymbol{\sigma} = \frac{\partial \Psi}{\partial \boldsymbol{\epsilon}} = \mathbf{E}^{\text{e}} : \left[\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^{\text{p}}\right], \quad \boldsymbol{\sigma}^{\text{p}} = -\frac{\partial \Psi}{\partial \boldsymbol{\epsilon}^{\text{p}}} = \boldsymbol{\sigma}, \quad \boldsymbol{\beta} = -\frac{\partial \Psi}{\partial \boldsymbol{b}} = -\frac{2}{3} H_{\text{kin}}\, \boldsymbol{b} \tag{13}$$

where $\boldsymbol{\sigma}^{\text{p}}$ is the plastic stress and $\boldsymbol{\beta}$ the back-stress. The plasticity models adopt the isotropic von Mises yield function, which is formulated as

$$\Phi(\boldsymbol{\sigma}, \mathbb{A}) = f^{\text{vM}}\left(\boldsymbol{\sigma}^{\text{dev}}_{\text{red}}\right) - Y_0, \quad f^{\text{vM}}\left(\boldsymbol{\sigma}^{\text{dev}}_{\text{red}}\right) = \sqrt{\frac{3}{2}\boldsymbol{\sigma}^{\text{dev}}_{\text{red}} : \boldsymbol{\sigma}^{\text{dev}}_{\text{red}}} \quad \text{where} \quad \boldsymbol{\sigma}^{\text{dev}}_{\text{red}} = \boldsymbol{\sigma}^{\text{dev}} - \boldsymbol{\beta} \tag{14}$$

Y_0 is the initial yield stress. The evolution of the plastic strain, $\boldsymbol{\epsilon}^{\text{p}}$, follows the associative flow rule

$$\dot{\boldsymbol{\epsilon}}^{\text{p}} = \dot{\lambda}\,\frac{\partial \Phi}{\partial \boldsymbol{\sigma}} = \dot{\lambda}\,\boldsymbol{\nu} \quad \text{where} \quad \boldsymbol{\nu} = \sqrt{\frac{3}{2}}\,\frac{\boldsymbol{\sigma}^{\text{dev}}_{\text{red}}}{\left\|\boldsymbol{\sigma}^{\text{dev}}_{\text{red}}\right\|} \tag{15}$$

The two prototype models differ in their kinematic hardening laws. For the first model, the simple Armstrong–Frederick (AF) hardening law [35] is adopted, where the evolution of $\boldsymbol{b}$ is formulated as

$$\dot{\boldsymbol{b}} = \dot{\lambda}\left[-\boldsymbol{\nu} + \frac{3}{2}\,\frac{\boldsymbol{\beta}}{\beta_\infty}\right] \tag{16}$$

β_∞ is the saturation value of the back-stress, $\boldsymbol{\beta}$. To evaluate the proposed framework for a more complex and nonlinear hardening law, the second prototype model employs the Ohno–Wang (OW) kinematic hardening law [36], given by

$$\dot{\boldsymbol{b}} = \dot{\lambda}\left[-\boldsymbol{\nu} + \frac{3}{2}\,\frac{\boldsymbol{\beta}}{f^{\text{vM}}(\boldsymbol{\beta})}\,\frac{\langle \boldsymbol{\nu} : \boldsymbol{\beta}\rangle}{\beta_\infty}\left[\frac{f^{\text{vM}}(\boldsymbol{\beta})}{\beta_\infty}\right]^m\right] \tag{17}$$

where m is a material parameter, and $\langle \bullet \rangle = 1/2(\bullet + |\bullet|)$ is the Macaulay bracket. The material model parameter values, presented in Table 1, are inspired by those calibrated against experimental data for isotropic R260 pearlitic rail steel [4].

3.2. NN-based integration of the prototype plasticity models

When integrating the prototype plasticity models with the proposed explicit NN-based time integration scheme described in Section 2.3, for a given $\Delta\boldsymbol{\epsilon}$, we first calculate $\Phi^{\text{tr}} = \Phi({}^n\boldsymbol{\sigma} + \mathbf{E} : \Delta\boldsymbol{\epsilon}, {}^n\boldsymbol{\beta})$. If $\Phi^{\text{tr}} \leq 0$, the loading is elastic such that $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\text{tr}}$, $\boldsymbol{\epsilon}^{\text{p}} = {}^n\boldsymbol{\epsilon}^{\text{p}}$, and $\boldsymbol{b} = {}^n\boldsymbol{b}$. Otherwise, if $\Phi^{\text{tr}} > 0$, we need to determine if the first part of the load step is elastic. If the material response at time ${}^n t$ was elastic, i.e., ${}^n\Phi = \Phi({}^n\boldsymbol{\sigma}, {}^n\boldsymbol{\beta}) < 0$, this will be the case as illustrated in the left part of Fig. 1. However, even if ${}^n\Phi = 0$, the

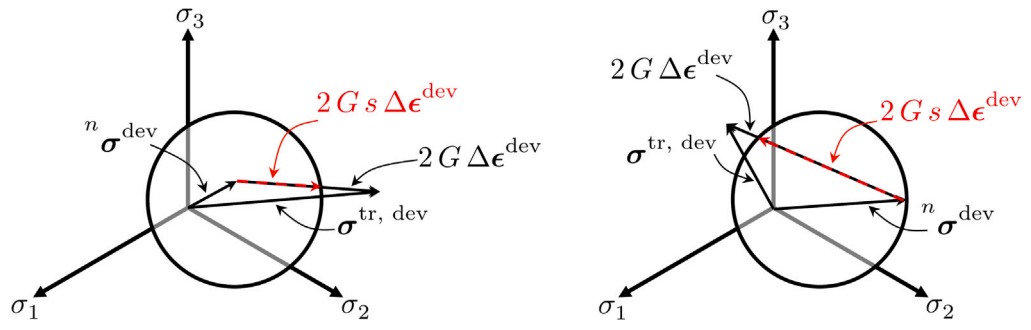


Fig. 1. Schematic illustration of two cases in which the scalar variable s needs to be determined. The left and right figures correspond, respectively, to the case where the old material response is elastic and to the case where the trial stress path crosses the yield surface. The illustration is presented in the deviatoric stress space.

first part could be elastic due to a load direction change; see the right part of Fig. 1. We detect this case by evaluating whether Φ decreases (i.e., becomes negative since the previous value was zero) at the beginning of the strain increment, specifically

$$\left.\frac{\partial \Phi({}^{n}\boldsymbol{\sigma} + \mathbf{E} : [\theta \Delta\boldsymbol{\epsilon}], {}^{n}\boldsymbol{\beta})}{\partial \theta}\right|_{\theta=0} \leq 0 \tag{18}$$

If either case is satisfied, we compute the plastic part of the applied $\Delta\boldsymbol{\epsilon}$. To be specific, the scalar variable $s \in [0, 1]$ is determined such that $\Phi(\boldsymbol{\sigma}^{\mathrm{lim}}, {}^{n}\boldsymbol{\beta}) = 0$, resulting in

$$f_s = \frac{3}{2}\left|\left|{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} + s\,\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] - {}^{n}\boldsymbol{\beta}\right|\right|^2 - Y_0^2 = 0 \tag{19}$$

which can be solved analytically.

To ensure that the response of the material model, using the NN-based time integrator, is independent of the choice of coordinate system, several approaches have been proposed in literature. These include training an NN on several transformations of raw data to learn frame invariance implicitly [37], using a suitable set of invariants as input features [20], or enforcing frame invariance at the neuron level [38]. As mentioned in Section 2.3, we adopt the second approach and choose the inputs to the NNs as invariants. This is computationally more efficient and requires less training data than the first approach [37]. Following [39], possible choices of invariants considering the prototype models are

$$\begin{array}{llll}
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{tr}}\right), & \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & \mathrm{tr}\left(\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^3\right), & \\
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{lim}}\right), & \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & \mathrm{tr}\left(\left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right]^3\right), & \\
 & {}^{n}\boldsymbol{\beta} : {}^{n}\boldsymbol{\beta}, & \mathrm{tr}\left([{}^{n}\boldsymbol{\beta}]^3\right), & \\
\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right]^2, & \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2 : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2 : \left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right]^2, \\
{}^{n}\boldsymbol{\beta} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & {}^{n}\boldsymbol{\beta} : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2, & [{}^{n}\boldsymbol{\beta}]^2 : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & [{}^{n}\boldsymbol{\beta}]^2 : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2, \\
\boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : {}^{n}\boldsymbol{\beta}, & \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : [{}^{n}\boldsymbol{\beta}]^2, & \left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right]^2 : {}^{n}\boldsymbol{\beta}, & \left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right]^2 : [{}^{n}\boldsymbol{\beta}]^2, \\
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} \cdot \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} \cdot {}^{n}\boldsymbol{\beta}\right) & & &
\end{array} \tag{20}$$

We restrict the nonlinearity of the inputs to the NNs to quadratic invariants; hence, the chosen invariants are

$$\begin{array}{lll}
J_{\mathrm{tr,tr}} = \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & J_{\mathrm{lim,lim}} = \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & J_{\beta,\beta} = {}^{n}\boldsymbol{\beta} : {}^{n}\boldsymbol{\beta}, \\
J_{\mathrm{tr,lim}} = \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & J_{\beta,\mathrm{tr}} = {}^{n}\boldsymbol{\beta} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & J_{\mathrm{lim},\beta} = \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : {}^{n}\boldsymbol{\beta}
\end{array} \tag{21}$$

and these are collected in the set $\mathbb{J}$.

For training the NNs, it is vital that inputs and outputs are scaled appropriately. By considering the generic formulation in Eq. (9) and the invariants in $\mathbb{J}$, we obtain the following formulations for the scaled NN-based time integrators, $\hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathbf{s}_i}$,

$$\hat{\mathfrak{g}}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} = \mathcal{NN}_{\epsilon^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \mathcal{NN}_{\epsilon^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \mathcal{NN}_{\epsilon^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)\,{}^{n}\hat{\boldsymbol{\beta}} \tag{22}$$

$$\hat{\mathfrak{g}}^{\mathrm{NN}}_{b} = \mathcal{NN}_{b,1}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \mathcal{NN}_{b,2}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \mathcal{NN}_{b,3}\left(\hat{\mathbb{J}}\right)\,{}^{n}\hat{\boldsymbol{\beta}} \tag{23}$$

with the unscaled time integrators,

$$\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} = \mathrm{SF}_{\epsilon^{\mathrm{p}}}\,\hat{\mathfrak{g}}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}}, \quad \mathfrak{g}^{\mathrm{NN}}_{b} = \mathrm{SF}_{b}\,\hat{\mathfrak{g}}^{\mathrm{NN}}_{b} \tag{24}$$

According to Eq. (9), the chosen set of ($N = 6$) invariants and ($N_{\mathrm{a}} = 1$) hardening variable result in $N(2 + N_{\mathrm{a}}) = 18$ terms for each state variable. However, some invariant derivatives are equal to zero, while others lead to identical evolution directions. After combining terms with identical evolution directions, Eqs. (22) and (23) are obtained. All scaled variables are denoted by a hat, $\hat{\bullet}$, and are scaled by the values in the training and validation data: all invariants in $\mathbb{J}$ are individually normalized by the range found in the data, and the evolution directions $\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$, $\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}$, ${}^{n}\boldsymbol{\beta}$ are individually normalized by their maximum Frobenius norms found

Algorithm 1: Numerical algorithm considering the prototype material models integrated using the proposed NN-based time integrator for a given $\Delta\boldsymbol{\epsilon}$

Compute Φ^{tr}

if $\Phi^{\mathrm{tr}} \leq 0$ **then**
- Elastic state
- $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{tr}} = {}^{n}\boldsymbol{\sigma} + \mathbf{E}^{\mathrm{e}} : \Delta\boldsymbol{\epsilon}$
- $\boldsymbol{\epsilon}^{\mathrm{p}} = {}^{n}\boldsymbol{\epsilon}^{\mathrm{p}}$
- $\boldsymbol{\beta} = {}^{n}\boldsymbol{\beta}$

else
- Plastic state
- **if** ${}^{n}\Phi < 0$ or $\left.\frac{\partial \Phi({}^{n}\boldsymbol{\sigma} + \mathbf{E} : [\theta \Delta\boldsymbol{\epsilon}], {}^{n}\boldsymbol{\beta})}{\partial \theta}\right|_{\theta=0} \leq 0$ **then**
  - Determine s from Eq. (19)
  - Compute $\boldsymbol{\sigma}^{\mathrm{lim}}$
    - $\boldsymbol{\sigma}^{\mathrm{lim}} = {}^{n}\boldsymbol{\sigma} + \mathbf{E}^{\mathrm{e}} : s\,\Delta\boldsymbol{\epsilon}$
- **else**
  - $\boldsymbol{\sigma}^{\mathrm{lim}} = {}^{n}\boldsymbol{\sigma}$
- Compute $\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}}$ and $\mathfrak{g}^{\mathrm{NN}}_{b}$
- Determine c from Eq. (26)
- Update the state variables and stresses
  - $\boldsymbol{\epsilon}^{\mathrm{p}} = {}^{n}\boldsymbol{\epsilon}^{\mathrm{p}} + c\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}}$
  - $\boldsymbol{\beta} = {}^{n}\boldsymbol{\beta} - \frac{2}{3} H_{\mathrm{kin}}\, c\, \mathfrak{g}^{\mathrm{NN}}_{b}$
  - $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{lim}} + \mathbf{E}^{\mathrm{e}} : [\Delta\boldsymbol{\epsilon} - \Delta\boldsymbol{\epsilon}^{\mathrm{p}}]$

in the data. Moreover, scaling factors $\mathtt{SF}_{\epsilon^{\mathrm{p}}}$ and $\mathtt{SF}_{b}$ are also calculated using maximum Frobenius norms of $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ and $\Delta\boldsymbol{b}$ from the data.

Finally, the increments of the state variables are calculated as

$$\Delta\boldsymbol{\epsilon}^{\mathrm{p}} = c\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} \quad \text{and} \quad \Delta\boldsymbol{b} = c\,\mathfrak{g}^{\mathrm{NN}}_{b} \tag{25}$$

where the correction factor, c, is computed from $\Phi(\boldsymbol{\sigma}, \mathbb{A}) = 0$. This leads to the expression

$$f_c = \frac{3}{2}\left|\left|\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta} - c\,\left[2\,G\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}^{\mathrm{NN}}_{b}\right]\right|\right|^2 - Y_0^2 = 0 \tag{26}$$

where we adopt $\Delta\boldsymbol{\beta} = -2/3\,H_{\mathrm{kin}}\,\Delta\boldsymbol{b}$. We choose the solution of c from Eq. (26) that is closest to unity. Note that, during the training of the NNs, $c = 1$ is assumed. The overall numerical algorithm based on the proposed NN-based integration scheme is presented in Algorithm 1.

4. Neural networks and training

This section describes the NN models employed to formulate $\mathfrak{g}^{\mathrm{NN}}_{\mathfrak{s}_i}$, the approach adopted to generate data, the method to evaluate the evolution directions in terms of necessity and sufficiency, as well as training results.

4.1. Feed-Forward Neural Networks (FFNNs)

We use fully connected FFNNs to formulate $\mathfrak{g}^{\mathrm{NN}}_{\mathfrak{s}_i}$ through $\mathcal{NN}_{\mathfrak{s}_i,1}$ - $\mathcal{NN}_{\mathfrak{s}_i,3}$ for each state variable, $\mathfrak{s}_i$. A fully connected FFNN establishes a mapping from an input vector, in our case $\hat{\mathbb{J}}$, to an output vector, in our case $\mathcal{NN}_{\mathfrak{s}_i,1}$ - $\mathcal{NN}_{\mathfrak{s}_i,3}$, by propagating information through m hidden layers. Each hidden layer l computes its output $\underline{\boldsymbol{x}}_l$ using an activation function a_l as $\underline{\boldsymbol{x}}_l = a_l\left(\underline{\underline{\boldsymbol{W}}}_l\,\underline{\boldsymbol{x}}_{l-1} + \underline{\boldsymbol{b}}_l\right)$, where $\underline{\underline{\boldsymbol{W}}}_l$ and $\underline{\boldsymbol{b}}_l$ are trainable weight matrices and bias vectors associated with layer l. In this study, the activation function for all hidden layers is chosen as $a_l = \tanh(x)$. The considered FFNNs consist of $m = 8$ hidden layers with 8 neurons and are trained over 100 000 epochs. The FFNNs' weights and biases are randomly initialized before being optimized with the gradient-based Rectified Adam (RAdam) optimizer [40] in the PyTorch library [41], using a learning rate of 10^{-4}. Although the selected hyperparameters were determined through trial and error to provide satisfactory training performance, further improvements in training efficiency may be possible through automated hyperparameter optimization; see [42].

To facilitate the optimization of the network parameters, i.e., weights and biases, during training [43], and to prevent vanishing gradients caused by large input values to the activation function, $\tanh(x)$ [44], scaling has been introduced as described in Section 3.2.

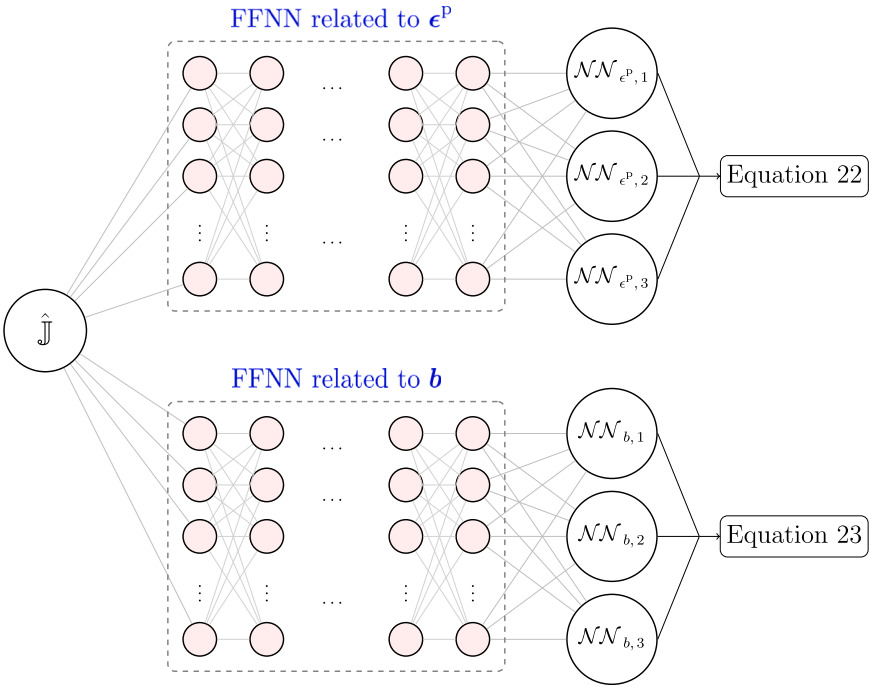


Fig. 2. Schematic illustration of the inputs and outputs of the NNs, which are embedded in Eqs. (22) and (23).

In Eqs. (22) and (23), we use one FFNN to formulate the increment of each state variable, $\Delta\mathfrak{s}_i$, as illustrated in Fig. 2. Each FFNN produces three scalar outputs denoted by $\mathcal{NN}_{\mathfrak{s}_i,1} - \mathcal{NN}_{\mathfrak{s}_i,3}$. The reason for considering separate FFNNs rather than a single FFNN with six outputs is to keep the networks smaller, thereby improving training efficiency and training performance. After using the outputs of the FFNNs in Eqs. (22) and (23) to predict $\hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathfrak{s}_i}$, the mean squared error loss, $\mathcal{L}_{\mathfrak{s}_i}$, can be calculated as

$$\mathcal{L}_{\mathfrak{s}_i} = \frac{1}{6\,N_{\mathrm{data}}} \sum_{j=1}^{N_{\mathrm{data}}} \left|\left| \left[\Delta\hat{\mathfrak{s}}_i\right]_j^{\mathrm{target}} - \hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathfrak{s}_i} \right|\right|^2 \tag{27}$$

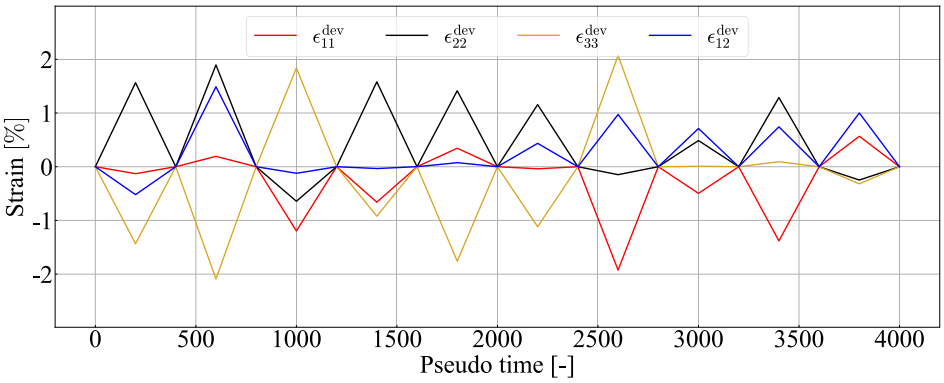
Here, N_{data} denotes the total number of training or validation data samples, and $\left[\Delta\hat{\mathfrak{s}}_i\right]^{\mathrm{target}}$ is the target state variable increments, normalized with $\mathtt{SF}_{\mathfrak{s}_i}$.

4.2. Data generation

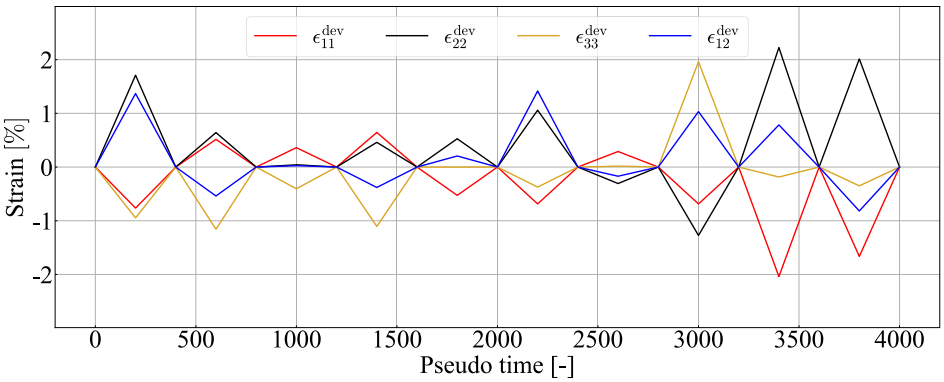
To train and validate the FFNNs, we generate data using the prototype plasticity models presented in Section 3.1 integrated using the implicit backward Euler time integration scheme. As mentioned in Section 2.2, on a material point level, we assume a linear variation of the strain increment within the considered time step. For training and validation data generation, two distinct 2D deviatoric multiaxial pulsating strain paths are generated over 10 loading cycles, each discretized into 400 time steps; see Figs. 3(a) and 3(b). A random peak value, $\left|\left|\boldsymbol{\epsilon}^{\mathrm{dev}}\right|\right| \in [0.0,\ 3.5]\%$, is generated for each cycle. The choice of pulsating loading is motivated by the application of railway mechanics, where rails are subjected to purely pulsating loading [9,45,46]. The strain histories are used in material point simulations, and the resulting stress–strain cycles are employed to generate data samples. Testing of the NN-based time integrator is performed separately from training by evaluating its performance using a 3D deviatoric multiaxial pulsating strain path comprising 100 cycles with a random peak value, $\left|\left|\boldsymbol{\epsilon}^{\mathrm{dev}}\right|\right| \in [0.0,\ 3.5]\%$, for each cycle.

We adopt an approach similar to that described in [17] to generate training and validation data samples, which is schematically presented in Fig. 4. We first split the data into load segments with linear strain paths, which, in this case, correspond to half-cycles. Next, data points corresponding to elastic material behavior, i.e., those for which $\Phi^{\mathrm{tr}} < 0$, are filtered out. As mentioned in Section 3.2, during the simulations, it is first assessed whether the scalar variable s needs to be computed. Thus, the data points marked with a gray circle in Fig. 4 are the first points used for data generation in each load segment. For a load segment with N_{p} points, $k-1$ data samples are generated for each point, $k \in [2, N_{\mathrm{p}}]$, which results in a total of $N_{\mathrm{p}}\,(N_{\mathrm{p}}-1)/2$ data samples. One data sample for the increment size $\Delta k \in [1,\ k-1]$ consists of

$$\begin{aligned} &\boldsymbol{\sigma}^{\mathrm{tr,\,dev}},\ \boldsymbol{\sigma}^{\mathrm{lim,\,dev}},\ {}^{n}\boldsymbol{\beta} \quad \text{where} \quad n = k - \Delta k \\ &\mathbb{J} = \{\boldsymbol{J}_{\mathrm{tr,tr}},\ \boldsymbol{J}_{\mathrm{lim,lim}},\ \boldsymbol{J}_{\beta,\beta},\ \boldsymbol{J}_{\mathrm{tr,lim}},\ \boldsymbol{J}_{\beta,\mathrm{tr}},\ \boldsymbol{J}_{\mathrm{lim},\beta}\} \\ &\Delta\boldsymbol{\epsilon}^{\mathrm{p}},\ \Delta\boldsymbol{b} \end{aligned} \tag{28}$$



(a) Strain paths for training data generation



(b) Strain paths for validation data generation

Fig. 3. 2D deviatoric multiaxial pulsating strain paths over 10 loading cycles.

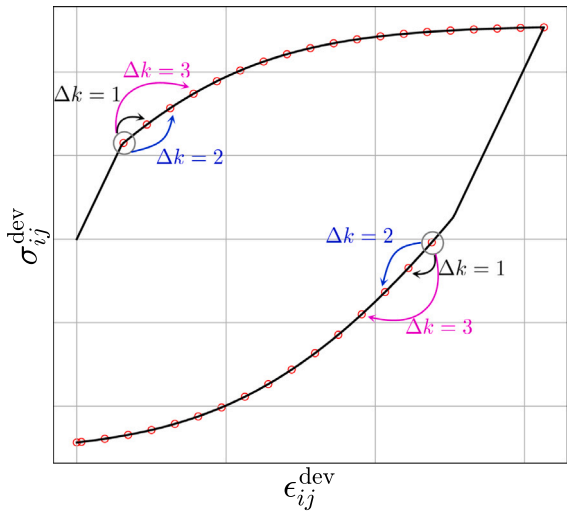


Fig. 4. Schematic illustration of the data generation approach after excluding the data points with elastic material behavior.

Since the aim is to accelerate the time integration of the DAEs of the plasticity models by allowing large $\Delta\epsilon$, we have not used all of the possible data points in each cycle for data generation. Instead, the data points are sampled at every 5th increment for training

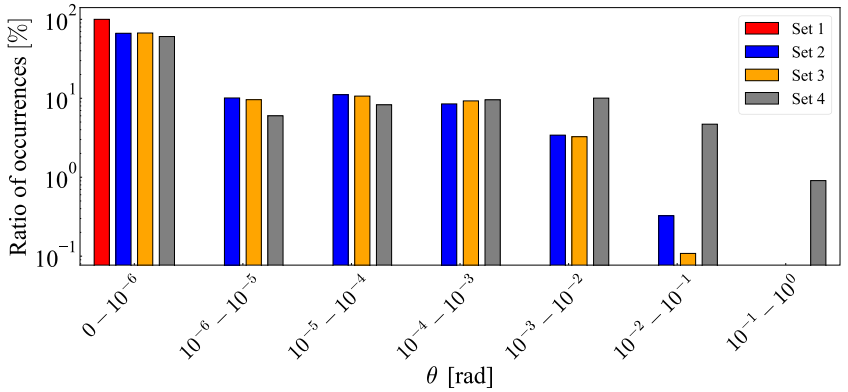


Fig. 5. Distribution of the computed angles, θ, between the target and predicted $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ considering the 3D training data, generated based on the AF-based prototype model. The four sets of evolution directions are: set 1: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, set 2: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\}$, set 3: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, and set 4: $\{\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$.

Table 2
Maximum values of θ between the target and predicted $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ and $\Delta\boldsymbol{b}$ considering the 3D training data. The evaluated sets are: set 1: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, set 2: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\}$, set 3: $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, and set 4: $\{\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$.

Sets	Set 1	Set 2	Set 3	Set 4
$\theta_{\max}$ (for $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$)	2.98×10^{-8}	0.34×10^{-1}	1.96×10^{-2}	5.43×10^{-1}
$\theta_{\max}$ (for $\Delta\boldsymbol{b}$)	2.98×10^{-8}	1.22×10^{-1}	3.31×10^{-2}	5.12×10^{-1}

data generation, which also improves the training efficiency. For validation, data samples are generated at every 10th increment to assess the performance of the NNs at larger increment sizes. This data generation strategy has produced 12 012 training samples and 2800 validation samples, respectively.

4.3. Evaluation of the considered evolution directions

The chosen six invariants in Eq. (21) result in three evolution directions, $\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$, $\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}$, and ${}^{n}\boldsymbol{\beta}$; see Eq. (9). The goal of this section is to determine if (a) these directions span all evolution directions and are sufficient for the model, and (b) if the number of considered directions can be reduced. To properly evaluate the three evolution directions, we have used training data generated based on a 3D deviatoric multiaxial strain path over 10 cycles with a random peak value, $\left\|\boldsymbol{\epsilon}^{\mathrm{dev}}\right\| \in [0.0,\ 3.5]\%$, for each cycle.

For each training data point, we have $\Delta\mathfrak{s}_i^{\mathrm{target}}$, i.e. $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ and $\Delta\boldsymbol{b}$, as well as $\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$, $\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}$ and ${}^{n}\boldsymbol{\beta}$. Considering Eqs. (22)–(25), we can determine coefficients α_{1i} - α_{3i} by a least square fit

$$\min \left\| \Delta\mathfrak{s}_i^{\mathrm{target}} - \mathrm{SF}_{\mathfrak{s}_i} \left[\alpha_{1i}\, \hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \alpha_{2i}\, \hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \alpha_{3i}\, {}^{n}\hat{\boldsymbol{\beta}} \right] \right\|^2 \tag{29}$$

With the identified coefficients, the predicted increments are obtained as

$$\Delta\mathfrak{s}_i^{\mathrm{pred}} = \mathrm{SF}_{\mathfrak{s}_i} \left[\alpha_{1i}\, \hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \alpha_{2i}\, \hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \alpha_{3i}\, {}^{n}\hat{\boldsymbol{\beta}} \right] \tag{30}$$

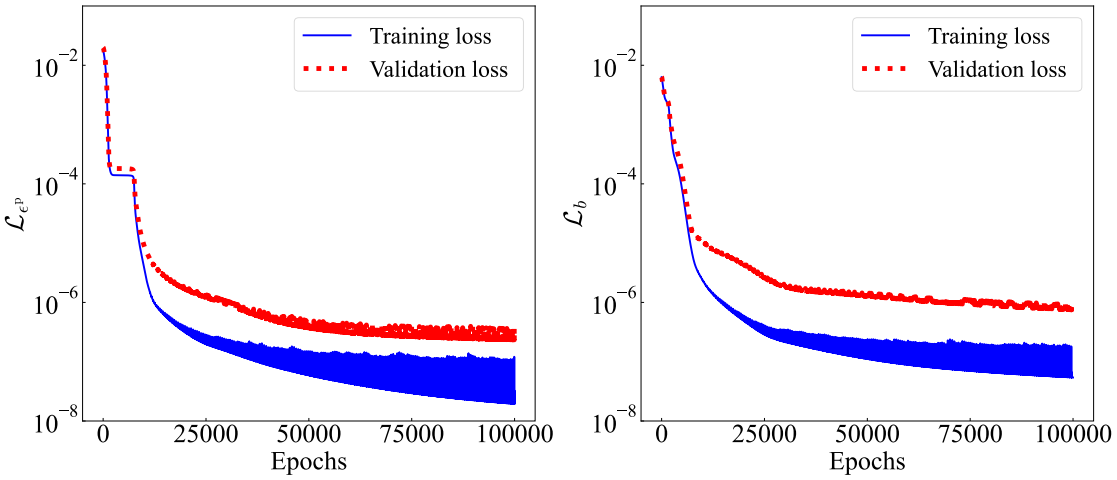
To evaluate whether the training data can be described by the chosen evolution directions, we compute the angular error, θ_i, as

$$\theta_i = \cos^{-1}\left(\frac{\Delta\mathfrak{s}_i^{\mathrm{target}} : \Delta\mathfrak{s}_i^{\mathrm{pred}}}{\left\|\Delta\mathfrak{s}_i^{\mathrm{target}}\right\| \left\|\Delta\mathfrak{s}_i^{\mathrm{pred}}\right\|} \right) \tag{31}$$

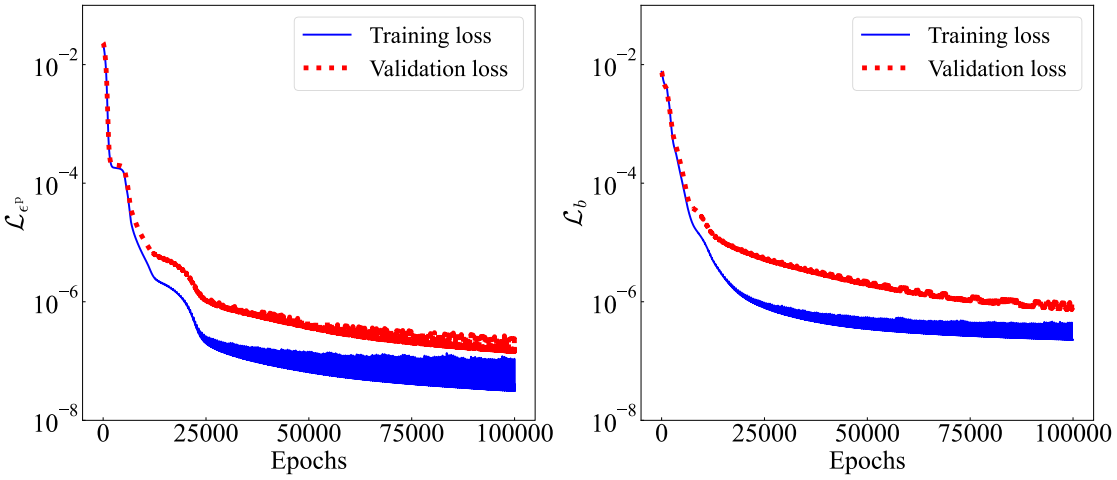
If θ_i is close to 0, then the selected evolution directions can accurately express the training data; otherwise, additional directions should be introduced. Using the 3D training data, generated based on the AF-based prototype model, we have compared the resulting θ_i values for four sets of evolution directions. Specifically, sets 1–4 are $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\}$, $\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, {}^{n}\boldsymbol{\beta}\}$, and $\{\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\}$. The computed θ_i values in terms of the relative number of occurrences considering $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ are illustrated in Fig. 5, showing that using all three directions (set 1) is both necessary and sufficient to describe the evolution of $\boldsymbol{\epsilon}^{\mathrm{p}}$. The same conclusion applies to $\boldsymbol{b}$, for which the angular errors are shown in Appendix A, Fig. A.18. Table 2 summarizes the maximum angular errors, where the maximum errors for set 1 are comparable to the numerical precision.

Table 3
Minimum training and validation losses ($\mathcal{L}_{\bullet}^{\text{train}}$ and $\mathcal{L}_{\bullet}^{\text{valid}}$) considering the 2D training data for different prototype models.

Model	$\mathcal{L}_{\epsilon^{\text{p}}}^{\text{train}}$ ($\times 10^{-7}$)	$\mathcal{L}_{\epsilon^{\text{p}}}^{\text{valid}}$ ($\times 10^{-7}$)	$\mathcal{L}_{b}^{\text{train}}$ ($\times 10^{-7}$)	$\mathcal{L}_{b}^{\text{valid}}$ ($\times 10^{-7}$)
AF	0.19	2.35	0.54	7.52
OW	0.31	1.48	2.28	7.40



(a) AF-based prototype model



(b) OW-based prototype model

Fig. 6. Training and validation loss evolutions against the 2D training data for $\mathcal{L}_{\epsilon^{\text{p}}}$ (left) and $\mathcal{L}_{b}$ (right).

4.4. FFNN training results

The evolution of the training and validation losses against the 2D data for the AF- and OW-based prototype models are presented in Figs. 6(a) and 6(b). In all cases, both losses show an overall continuous decrease before a slower decay toward convergence, and no sign of overfitting can be observed. The training and validation losses corresponding to the epoch with the lowest training loss are presented in Table 3 for both prototype models.

Figs. 7(a) and 7(b) show the distribution of the norms of absolute errors in $\Delta\boldsymbol{\epsilon}^{\text{p}}$ and $\Delta\boldsymbol{b}$ for all the 2D training data generated from the AF-based prototype material model. The training data generation strategy produces more data samples for small strain increments than for large ones. However, the largest errors for both $\Delta\boldsymbol{\epsilon}^{\text{p}}$ and $\Delta\boldsymbol{b}$ are observed for $\left\|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right\| < 0.01$, although they occur with low frequency. As $\left\|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right\|$ increases, the maximum error decreases. This trend breaks for $\left\|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right\| > 0.02$, where there

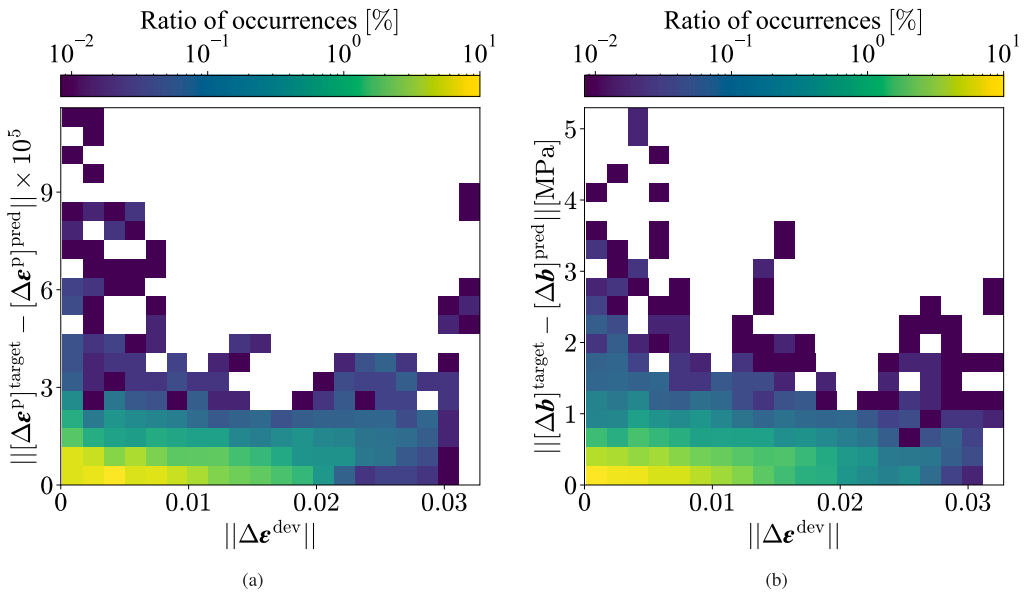


Fig. 7. 2D histograms of the norm of absolute errors in (a) $\Delta\epsilon^{\mathrm{p}}$ and (b) $\Delta\boldsymbol{b}$ versus $||\Delta\epsilon^{\mathrm{dev}}||$ using the 2D training data generated from the AF-based prototype model. White bins correspond to zero occurrences.

is less training data, and the accuracy of the trained FFNNs decreases. The corresponding results when using the 2D training data generated from the OW-based prototype model are provided in Appendix A, Fig. A.19.

5. Results and discussions

In this section, we present and discuss the results from the performance evaluation of the proposed NN-based time integration. In particular, we have employed the NN-based integration scheme in material point and FE simulations and assessed its performance in terms of accuracy and computational time.

5.1. Material point simulations

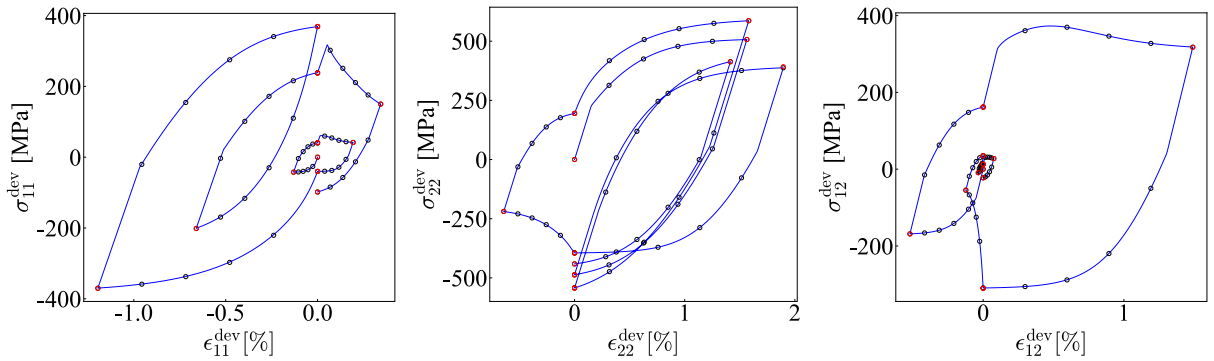
The results from the material point simulations are presented and discussed in this section. In the following, the reference solution is obtained with either the AF- or OW-based prototype material models, integrated with the implicit backward Euler time integration algorithm. By using very small time steps, i.e., 400 time steps per cycle, the reference solution is regarded as the true solution to the DAEs that are being solved.

5.1.1. AF-based prototype material model

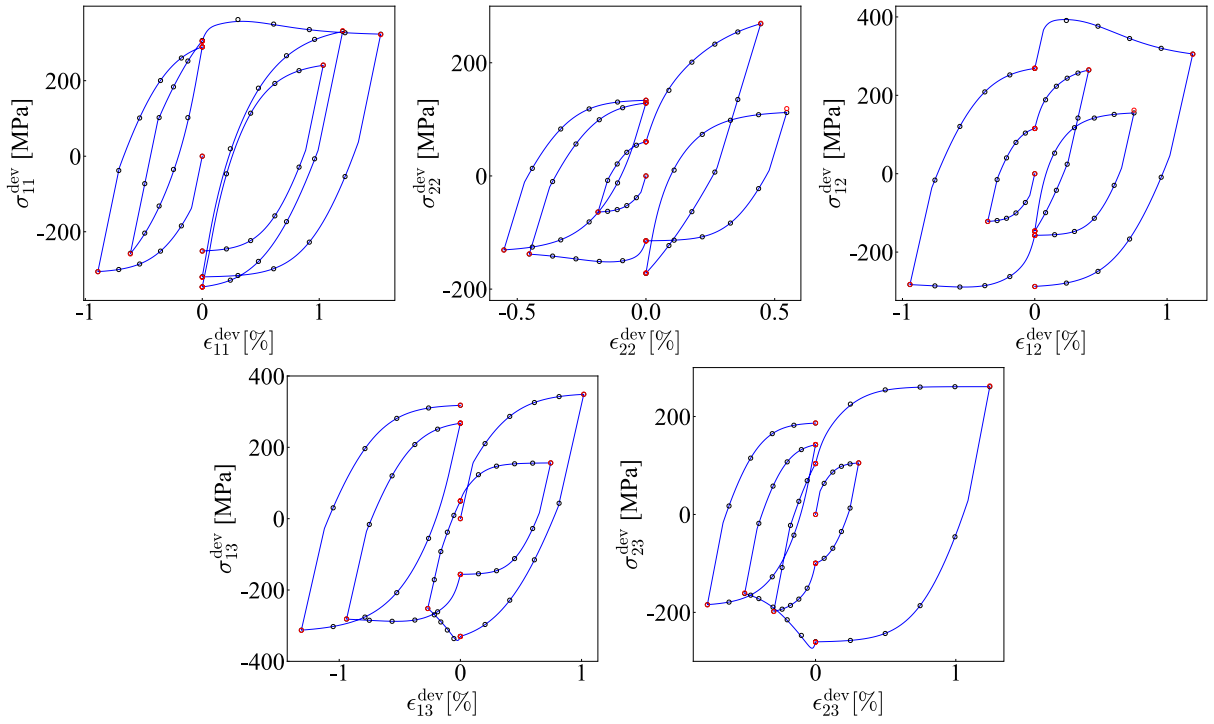
For the AF-based prototype model, the deviatoric axial and shear responses over the first 5 training data cycles are presented in Fig. 8(a). The predictions obtained with the NN-based time integrator, using 5 strain increments and even a single increment per half-cycle, show excellent agreement with the reference solution.

To investigate the influence of not accounting for the correction factor, c, i.e., $c = 1$ during the simulations, on the predictions obtained with the NN-based time integrator, we have calculated the errors in $||\Delta\epsilon^{\mathrm{p}}||$ as $[c-1]\,||\Delta\epsilon^{\mathrm{p}}||$ and presented them in Fig. 9. Considering the case with 5 strain increments per half-cycle, the c factors have not been calculated for every strain increment within a cycle, since some increments correspond to elastic loading/unloading, e.g., in cycle 4. In contrast, for the case with 1 increment per half-cycle, they have been computed at each increment. The errors arising from the assumption of $c = 1$ during plastic loading are more significant in the case using 1 increment with larger $||\Delta\epsilon^{\mathrm{p}}||$. The results achieved using this assumption in Eq. (25) are further shown in Fig. 10, which compares the von Mises stress values, σ^{vM}, at the peak strain of each cycle. It should be noted that not enforcing the consistency condition, $\Phi = 0$, during plastic loading might lead to the calculation of the s variable in the next step if ${}^{n}\Phi < 0$; see Section 3.2. In cycle 2 for the case with 1 increment per half-cycle, ${}^{n}\Phi > 0$, which is inconsistent. Although Eq. (18) was fulfilled, no solution for the s variable from Eq. (19) was found. Moreover, even though the predictions obtained with $c = 1$ for the case with 5 increments per half-cycle are in close agreement with those when c is analytically calculated, inconsistencies occurred when the model was tested over a larger number of cycles. This highlights the importance of the c factor in our proposed NN-based framework. Hereafter, the discussion is based on the results obtained when c is calculated during plastic loading.

Considering Fig. 10, the simulation of 10 cycles has resulted in a maximum absolute error of 1.12 MPa considering the NN-based time integrator with 5 strain increments per half-cycle and 5.09 MPa with 1 increment. Given that the maximum σ^{vM} obtained with



(a) 2D training data



(b) 3D test data

Fig. 8. The first 5 stress–strain cycles from the training and test data generated with the AF-based material model (blue) compared with predictions obtained with the NN-based time integration scheme. The black and red circles correspond to the NN-based predictions using 5 and 1 strain increments per half-cycle, respectively. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

the reference solution is approximately 900 MPa, the results demonstrate an excellent capability of the NN-based time integrator to reproduce the reference solutions. Moreover, despite the NN-based time integration being explicit, we note that it provides very good accuracy for very large strain increments.

To assess the generalizability of the NN-based time integrator, we have simulated 100 cycles under 3D loading with variable strain ranges in each cycle. Fig. 8(b) shows a very good fit with the reference results, considering both 5 and 1 strain increments per half-cycle, even for the out-of-plane shear components. This good agreement can also be observed in Fig. 11(a), presenting the σ^{vM} values at the peak strains in each cycle. To be specific, the maximum absolute errors in σ^{vM} over 100 cycles are roughly 7.14 MPa and 15.8 MPa for 5 and 1 strain increments per half-cycle, respectively, which should be viewed in relation to the maximum σ^{vM} of about 900 MPa. The solutions show that the AF-based material model integrated with the NN-based time integration scheme trained only on the 2D data provides very good predictions for the unseen 3D loading conditions. Note that training the NNs based on the 3D training data used in Section 4.3 led to comparable predictive accuracy.

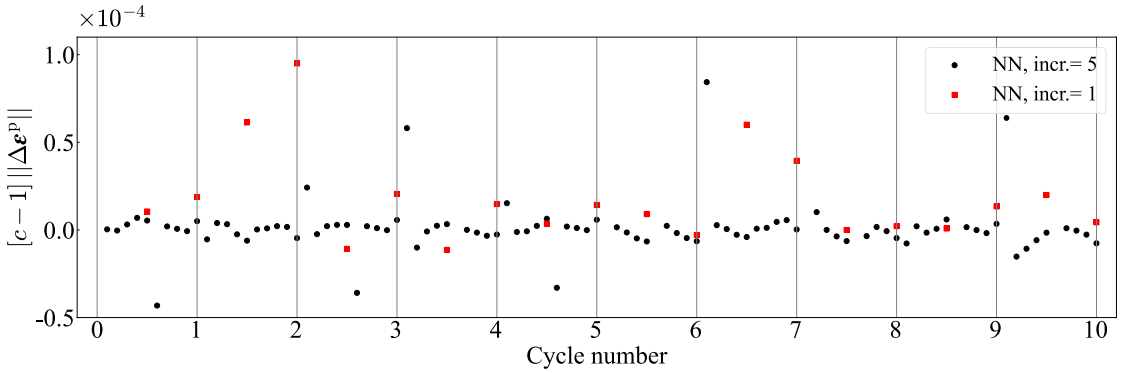


Fig. 9. Calculated errors in $||\Delta\epsilon^{p}||$ due to assuming $c = 1$, i.e., not enforcing the consistency condition, in each cycle when taking 5 and 1 strain increments per half-cycle (incr.), using the 2D training data. NN refers to the NN-based time integrator. The vertical dashed lines indicate the boundaries between consecutive cycles.

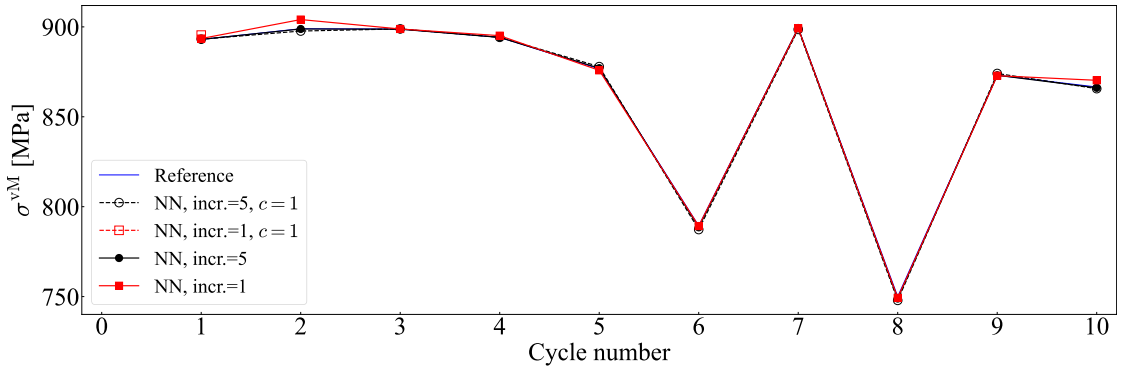


Fig. 10. von Mises stresses at the peak strains over 10 cycles for the 2D training data. Reference and NN refer to the AF-based material model integrated using the backward Euler and NN-based time integrators, respectively. incr. denotes the number of increments per half-cycle. $c = 1$ means that the consistency condition has not been enforced.

5.1.2. OW-based prototype material model

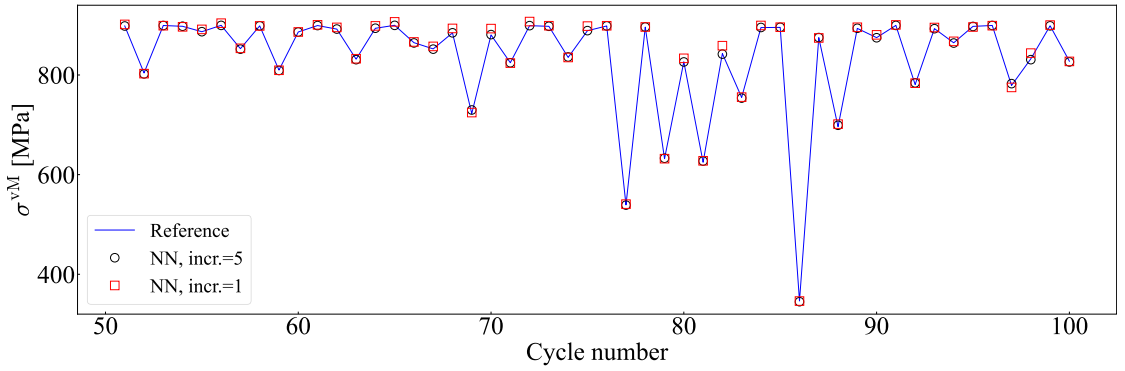
We have also applied the NN-based time integration scheme, trained against the 2D training data, to solve the DAEs of the OW-based prototype model. The predicted axial and shear responses during the first 5 cycles are shown in Fig. 12 for the 3D test data and in Fig. A.20 in Appendix A for the 2D training data. Similar to the results presented in Section 5.1.1, excellent agreement can be observed for both 5 and 1 increments per half-cycle. Fig. 11(b) evaluates the accuracy of σ^{vM} values at the peak strains. The maximum absolute errors are approximately 17.4 MPa and 15.7 MPa when using 5 and 1 strain increments per half-cycle, respectively. This assessment further demonstrates the promising performance of the proposed time integration scheme when applied to a more complex material model.

5.2. FE simulations

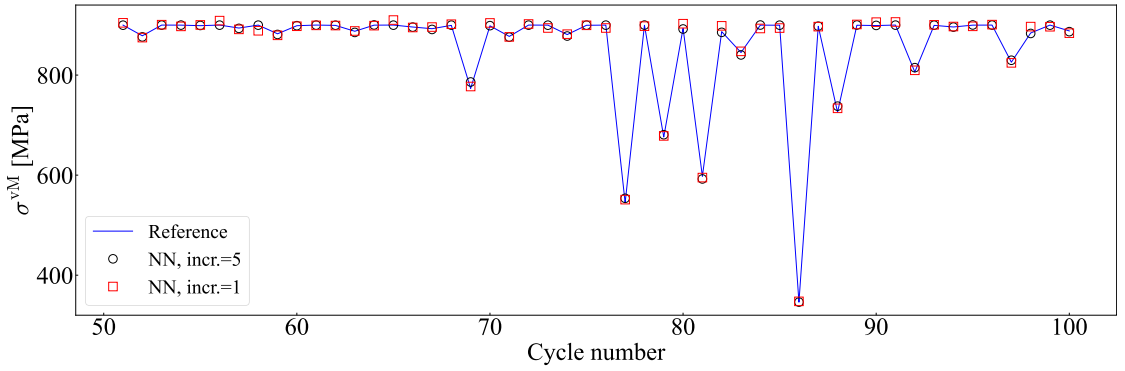
To demonstrate the applicability of the NN-based time integration to boundary value problems, we have incorporated it into the commercial FE code Abaqus [47]. The FE model is a quarter of a rectangular plate with a central circular hole. The plate geometry and the considered pulsating displacement-controlled loading with varying ranges are shown in Fig. 13. The FE mesh consists of 1362 linear plane strain triangular elements. The plate is subjected to 100 loading cycles, during which the vertical displacement is prescribed along the upper edge of the quarter model.

In all simulations, the AF-based prototype plasticity model is used, while only its time integration differs. The reference FE solution employs the backward Euler time integration algorithm with 100 displacement increments per half-cycle over the pseudo-time interval from 0 to 1. The NN-based time integrator is evaluated using 5 and 1 displacement increments per half-cycle. The material model using either of the time integrators has been implemented in Abaqus through user-defined subroutines. For the NN-based time integration, the algorithmic tangent stiffness presented in Appendix B has been used in the global equilibrium iterations. Line search has been enabled for all simulations to aid global convergence.

The von Mises stress, σ^{vM}, distributions in cycles 50 and 100 at the peak displacements are illustrated in Fig. 14a and c. The results obtained with the NN-based time integrator show close agreement with the reference solutions. To further show the differences



(a) AF-based material model



(b) OW-based material model

Fig. 11. von Mises stresses at the peak strains for the 3D test data. Reference and NN refer to the prototype material model integrated with the backward Euler and NN-based time integrators, respectively. incr. denotes the number of increments per half-cycle. The last 50 of 100 simulated cycles are shown.

between the solutions, spatial distributions of the absolute errors in σ^{vM} are presented in Fig. 14b and d. In cycle 50, the maximum absolute errors are 3.77 MPa and 4.90 MPa for 5 and 1 increments per half-cycle, respectively, while in cycle 100 they are 4.50 MPa and 3.50 MPa. Relative to the maximum σ^{vM} values in the plate, approximately 733 MPa in cycle 50 and 821 MPa in cycle 100, these errors show that the FE simulations employing the NN-based time integrator with both increment numbers provide accurate predictions.

To showcase the predictions obtained with the NN-based time integrator over the 100 simulated cycles in the highly stressed region near the hole, Fig. 15 compares σ^{vM} values at the peak displacements for the critical element indicated in Fig. 13. Despite the maximum absolute errors of 7.80 MPa and 5.84 MPa for 5 and 1 displacement increments per half-cycle, the predictions provided by the explicit NN-based integration scheme are very promising. Further, the strain histories during the last cycle for the critical element are shown in Fig. 16(a), comparing the reference solutions and the NN-based solutions when using 1 increment per half-cycle. Considering the reference results, the strain paths are nonlinear during the displacement increments, which violates our assumption of a linear variation of ϵ within a time step. Nevertheless, as shown in Fig. 16(b), the predicted stresses obtained with the NN-based time integrator are in close agreement with the reference solutions. It should be noted that σ_{11} and σ_{12} tend toward zero as the mesh is further refined near the hole.

Fig. 17 presents the distribution of the absolute errors in σ^{vM} over all cycles and finite elements. For both numbers of displacement increments, most errors are concentrated in the lowest interval of 0–3 MPa, accounting for approximately 88% to 98% of all occurrences. The relative frequency decreases rapidly as the error increases, with only a small fraction of cases showing errors larger than 9 MPa. The maximum absolute errors considering 5 and 1 displacement increments are 21.3 MPa and 9.51 MPa, respectively. Compared with the maximum σ^{vM} of 885 MPa from the reference solution, these errors indicate high accuracy of predictions when using the NN-based time integrator.

Finally, the computational efficiency of the FE simulations has been compared in terms of Central Processing Unit (CPU) time. Each FE simulation was repeated 5 times, and the averaged results are presented in Table 4. All simulations were performed on a Windows laptop equipped with an Intel Core Ultra 7 165H processor and 32 GB of RAM. The FE simulations employing the NN-based integrator with 5 and 1 displacement increments per half-cycle are approximately 14.6 and 33.3 times faster than the reference FE simulation. To provide a comparison at a similar level of accuracy, we have reduced the number of displacement increments used with the backward Euler scheme. This has been done until the maximum absolute error in σ^{vM}, evaluated over all cycles and

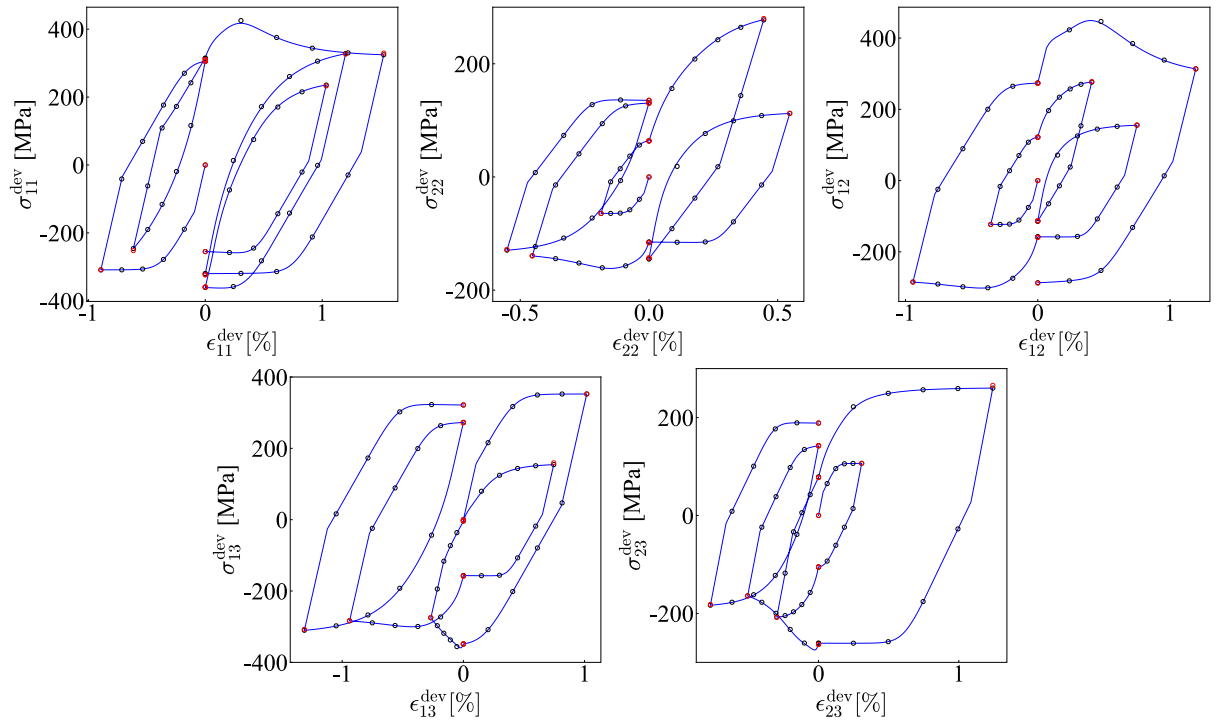


Fig. 12. The first 5 stress–strain cycles from the 3D test data generated with the OW-based material model (blue) compared with predictions obtained with the NN-based time integration scheme. The black and red circles correspond to the NN-based predictions using 5 and 1 strain increments per half-cycle, respectively. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

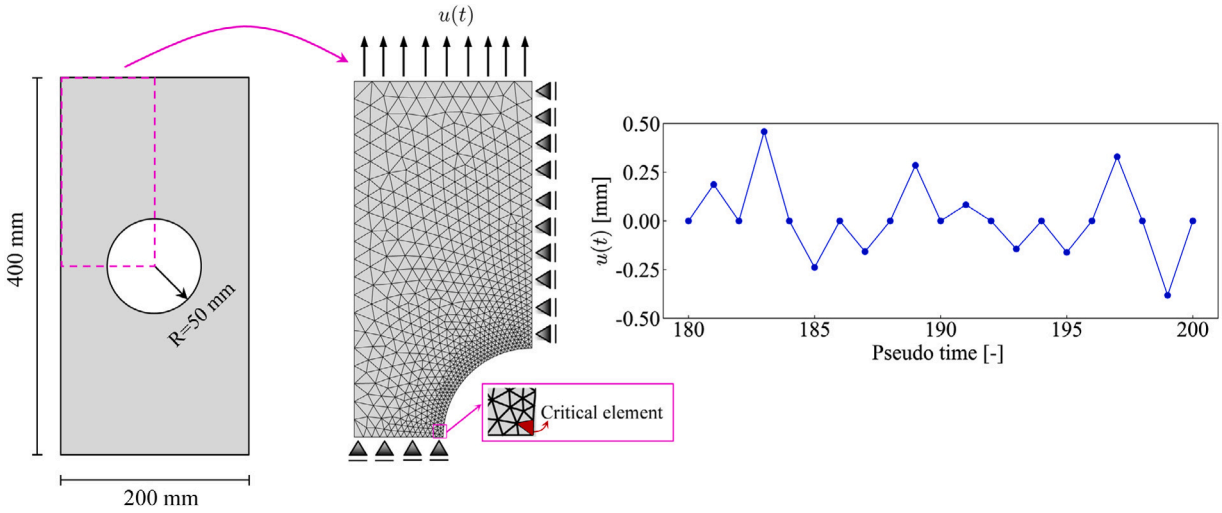


Fig. 13. Geometry and boundary conditions for the plate with a hole under displacement-controlled pulsating loading. The last 10 of 100 simulated cycles are shown.

finite elements relative to the highly resolved solution using 100 displacement increments per half-cycle, became comparable to that obtained with the NN-based integrator. For this purpose, the number of displacement increments has been decreased to 8 per half-cycle. This has resulted in the maximum absolute error of 16.1 MPa in σ^{vM}. It should be mentioned that lowering the number of increments to 5 did not give convergence during the displacement iterations. Compared with the simulations using the backward Euler scheme with 8 increments per half-cycle, the simulations using the NN-based time integrator with 5 and 1 increments are roughly 1.4 and 3.2 times faster, respectively.

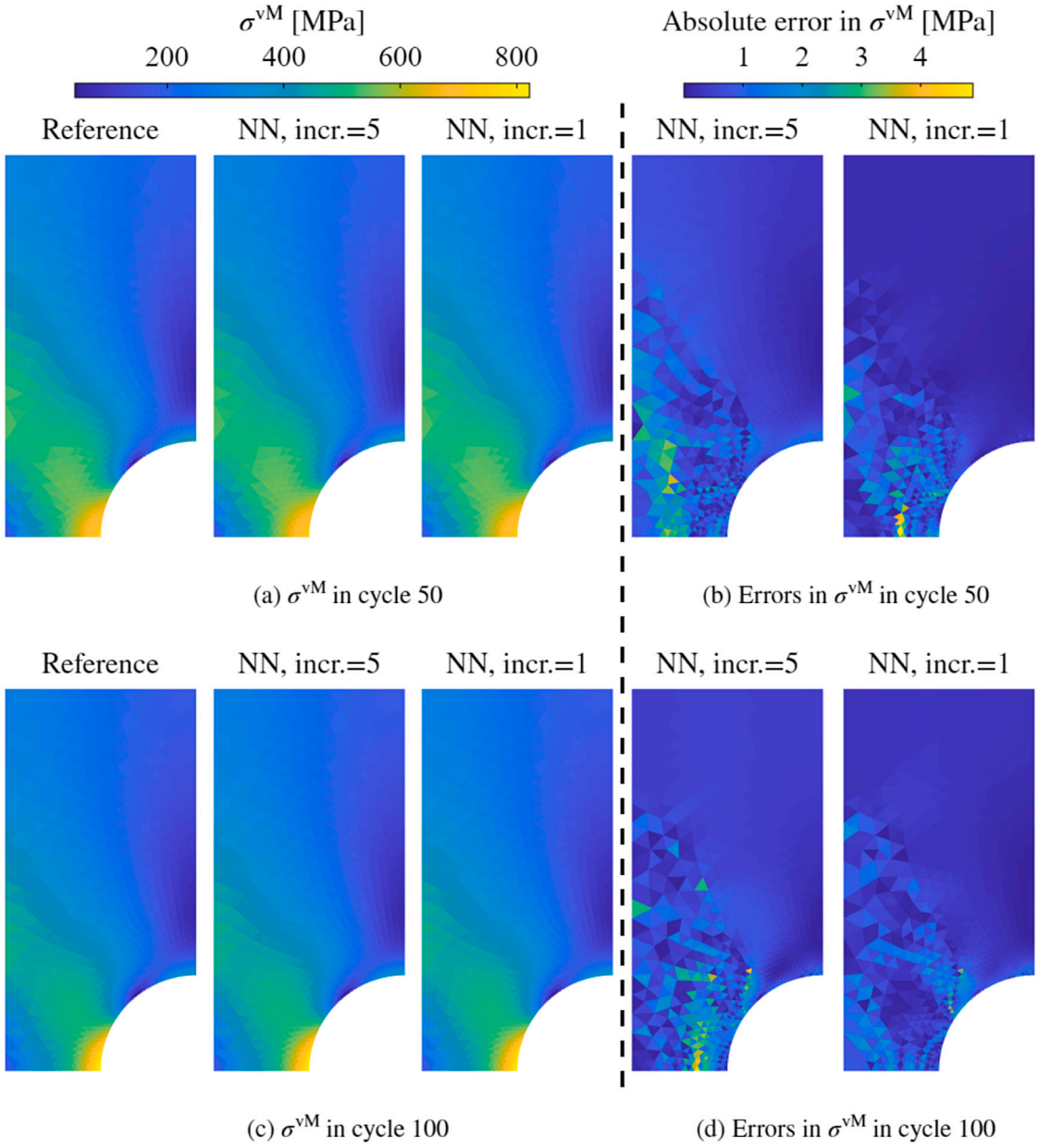


Fig. 14. von Mises stress, σ^{vM}, and the corresponding absolute error distributions in cycles 50 and 100, presented at the peak displacement. NN refers to the NN-based time integration scheme with the number of displacement increments per half-cycle indicated by incr.

Table 4
CPU time comparison between the FE simulations using the backward Euler (Reference) and NN-based time integrators.

Model	Reference		NN	
No. displacement increments	100	8	5	1
Averaged CPU time [s]	672[a]	63.8	46.2	20.2
Maximum error in σ^{vM} [MPa]	–	16.1	21.3	9.51

[a] Using the Abaqus built-in material model has resulted in an averaged CPU time of 670 [s].

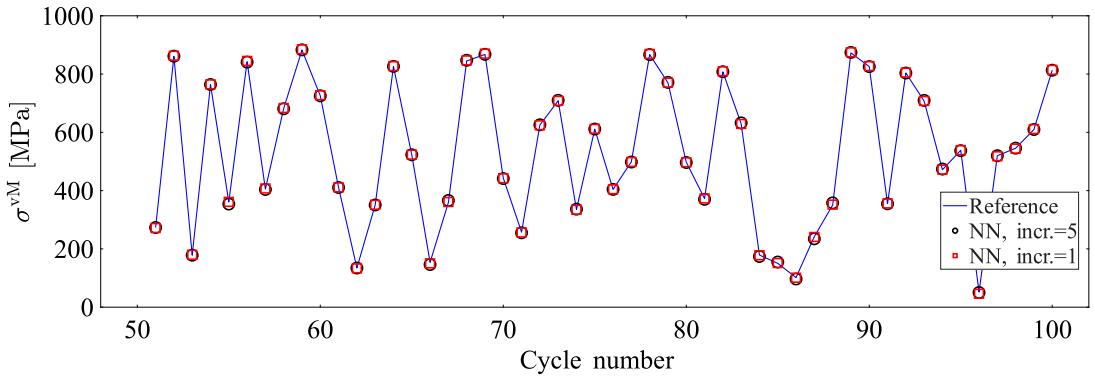


Fig. 15. von Mises stresses, σ^{vM}, in the critical element at the peak displacements. NN refers to the NN-based time integration with the number of increments per half-cycle denoted by incr. Results are shown for the last 50 of 100 simulated cycles.

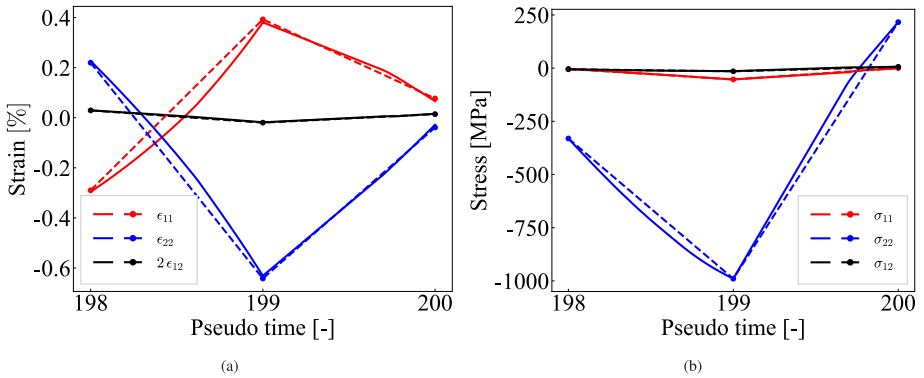


Fig. 16. (a) Strain histories and (b) stress histories for the critical element (indicated in Fig. 13) in the last cycle. The solid and dashed lines correspond to the results from the FE models using the backward Euler time integrator and the NN-based integrator with 1 displacement increment per half-cycle, respectively.

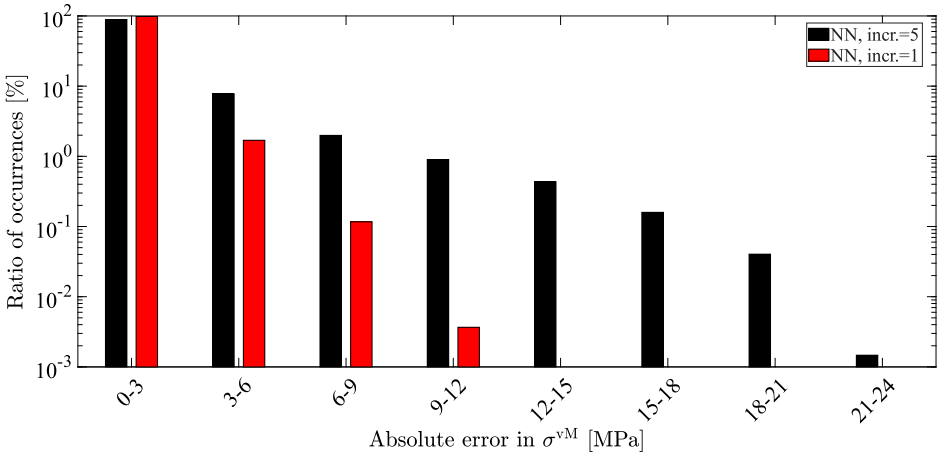


Fig. 17. Distribution of absolute errors in von Mises stresses (σ^{vM}) at the peak displacements over 100 cycles, considering all finite elements. NN, incr. = 5 and NN, incr. = 1 refer to the FE models using the NN-based time integrator with 5 and 1 displacement increments per half-cycle, respectively.

6. Concluding remarks

In this contribution, we have presented an NN-based framework to accelerate the time integration of DAEs in rate-independent nonlinear constitutive models. The key features of the framework are

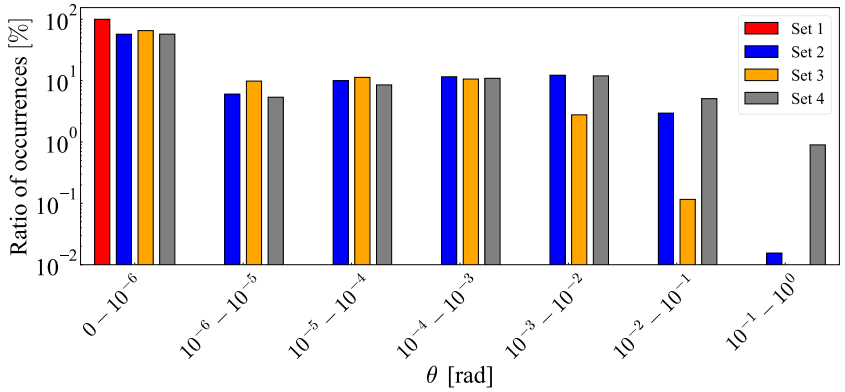


Fig. A.18. Distribution of the computed angles, θ, between the target and predicted $\Delta \boldsymbol{b}$ considering the 3D training data, generated based on the AF-based prototype model. The four sets of evolution directions are: set 1: $\{\boldsymbol{\sigma}^{\text{tr, dev}}, \boldsymbol{\sigma}^{\text{lim, dev}}, {}^{n}\boldsymbol{\beta}\}$, set 2: $\{\boldsymbol{\sigma}^{\text{tr, dev}}, \boldsymbol{\sigma}^{\text{lim, dev}}\}$, set 3: $\{\boldsymbol{\sigma}^{\text{tr, dev}}, {}^{n}\boldsymbol{\beta}\}$, and set 4: $\{\boldsymbol{\sigma}^{\text{lim, dev}}, {}^{n}\boldsymbol{\beta}\}$.

- Exact encoding of the separation between elastic and plastic responses.
- State variable updates formulated by NNs with invariants as inputs.
- Pre-identification of sufficient and necessary evolution directions based on the training data.
- Exact fulfillment of the plastic consistency condition.

We have chosen two distinct prototype material models with the von Mises yield function and different nonlinear kinematic hardening laws. Under two-dimensional pulsating proportional multiaxial loading, we generated training and validation data for the FFNNs using the prototype models integrated with the implicit backward Euler time integration algorithm. Moreover, 100 cycles of three-dimensional proportional multiaxial loading were used for testing. While only 10 loading cycles were used to generate the training data, the proposed NN-based time integration scheme gave excellent accuracy for material point simulations. Even when taking a single strain increment per half-cycle for 100 loading cycles, it achieved accurate predictions, indicating very good generalization capability of the NN-based time integrator. Its performance was further evaluated in cyclic FE simulations. The von Mises stress predictions were accurate, with a maximum error of less than 2% of the maximum stress over 100 cycles when using a single displacement increment per half-cycle. Moreover, the NN-based time integration of the FE model with a single increment per half-cycle is roughly 3 times faster than the backward Euler time integration at comparable accuracy.

CRediT authorship contribution statement

Nasrin Talebi: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Magnus Ekh:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **Knut Andreas Meyer:** Writing – review & editing, Supervision, Methodology, Conceptualization.

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.

Acknowledgments

This work is part of the ongoing activities within the National Center of Excellence CHARMEC (www.chalmers.se/charmec). Parts of the study have been funded by Europe's Rail project IAM4RAIL under grant agreement No. 101101966. Parts of the computations were enabled by resources provided by Chalmers e-Commons.

Appendix A. Additional results

This section provides supplementary results for Sections 4.3, 4.4, and 5.1.2. Fig. A.18 shows the computed θ_i values versus the relative number of occurrences considering $\Delta \boldsymbol{b}$, using the proposed method described in Section 4.3.

The distribution of the norms of absolute errors in $\Delta \boldsymbol{\epsilon}^{\text{p}}$ and $\Delta \boldsymbol{b}$ for all 2D training data with different strain increment sizes, considering the data from the OW-based prototype material model, are presented in Figs. A.19(a) and A.19(b).

Fig. A.20 compares the deviatoric axial and shear responses over the first 5 cycles from the reference OW-based model and those obtained with the NN-based time integrator, considering the 2D training data.

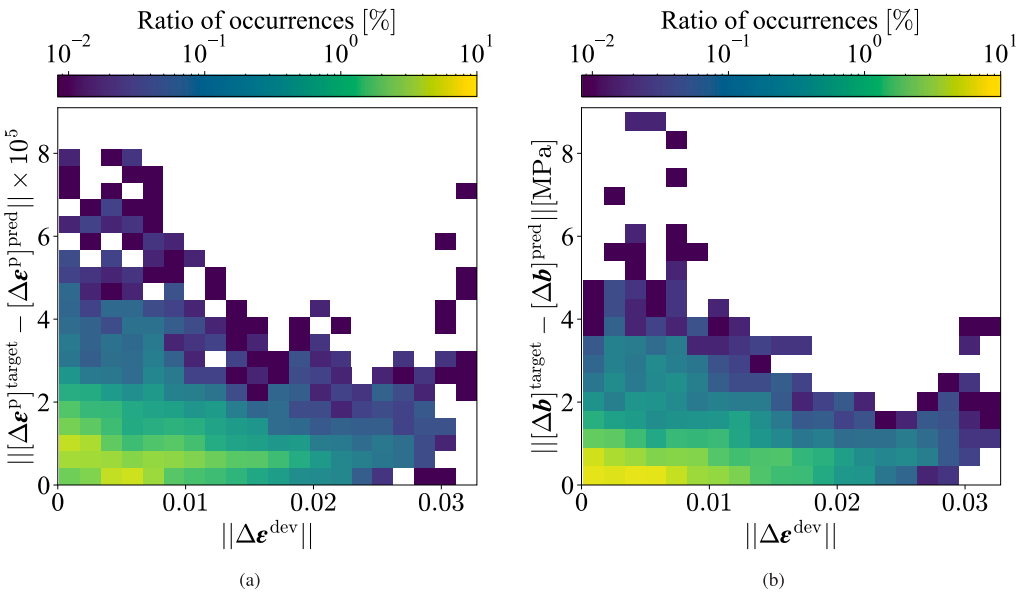


Fig. A.19. 2D histograms of the norm of absolute errors in (a) $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ and (b) $\Delta\boldsymbol{b}$ versus $||\Delta\boldsymbol{\epsilon}^{\mathrm{dev}}||$ using the 2D training data generated from the OW-based prototype model. White bins correspond to zero occurrences.

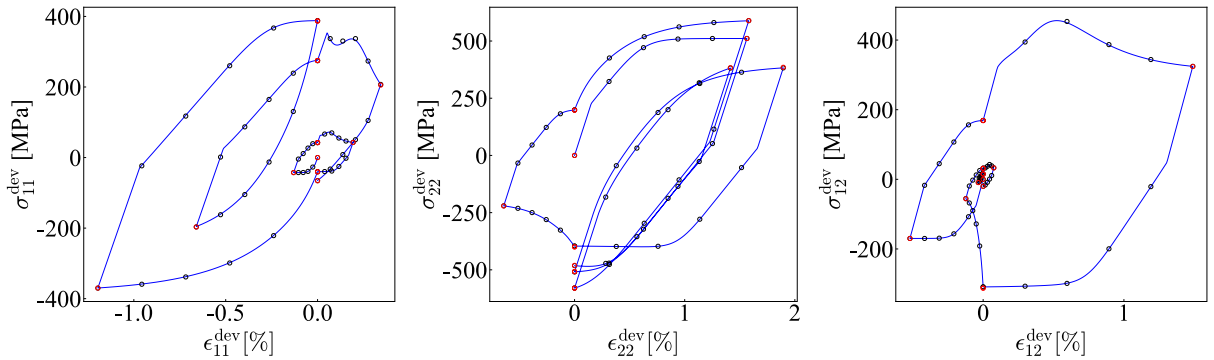


Fig. A.20. The first 5 stress–strain cycles from the 2D training data generated with the OW-based material model (blue) compared with predictions obtained with the NN-based time integration scheme. The black and red circles correspond to the NN-based predictions using 5 and 1 strain increments per half-cycle, respectively. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

Appendix B. Algorithmic tangent stiffness

This section presents the derived analytical expression for the Algorithmic Tangent Stiffness (ATS) employed in the global equilibrium iterations of the FE simulations using the NN-based time integrator. The ATS tensor is defined as

$$\mathbf{E}^{\mathrm{a}} = \frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\boldsymbol{\epsilon}} = \frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\Delta\boldsymbol{\epsilon}} \tag{B.1}$$

Considering that $\boldsymbol{\sigma} = {}^{n}\boldsymbol{\sigma} + \mathbf{E}^{\mathrm{e}} : [\Delta\boldsymbol{\epsilon} - \Delta\boldsymbol{\epsilon}^{\mathrm{p}}]$, the ATS tensor is expressed as

$$\mathbf{E}^{\mathrm{a}} = \mathbf{E}^{\mathrm{e}} - \mathbf{E}^{\mathrm{e}} : \frac{\mathrm{d}\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\mathrm{d}\Delta\boldsymbol{\epsilon}} = \mathbf{E}^{\mathrm{e}} - \mathbf{E}^{\mathrm{e}} : \frac{\partial\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} : \mathbf{E}^{\mathrm{e}} \tag{B.2}$$

With Eq. (25), the differentiation of $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ with respect to $\boldsymbol{\sigma}^{\mathrm{tr}}$ leads to

$$\frac{\partial\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} = c\,\frac{\mathrm{d}\mathbf{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} + \mathbf{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} \otimes \frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} \tag{B.3}$$

$\mathrm{d}\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$ is derived as

$$\begin{aligned}\frac{\mathrm{d}\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = \mathrm{SF}_{\epsilon^{\mathrm{p}}}\Bigg[&\mathcal{NN}_{\epsilon^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)\frac{1}{\left|\left|\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right|\right|_{\max}}\mathbf{I}^{\mathrm{dev}} + \mathcal{NN}_{\epsilon^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)\frac{1}{\left|\left|\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right|\right|_{\max}}\left[\frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} : \mathbf{I}^{\mathrm{dev}}\right] + \\ \hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}}\otimes\Bigg[&\frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}}2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,lim}}}\left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}} + \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\right] + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\beta}}{}^{n}\boldsymbol{\beta}\Bigg] + \\ \hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}}\otimes\Bigg[&\frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}}2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,lim}}}\left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}} + \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\right] + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\beta}}{}^{n}\boldsymbol{\beta}\Bigg] + \\ {}^{n}\hat{\boldsymbol{\beta}}\otimes\Bigg[&\frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}}2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,lim}}}\left[\boldsymbol{\sigma}^{\mathrm{lim,\,dev}} + \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\right] + \frac{\partial\mathcal{NN}_{\epsilon^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\beta}}{}^{n}\boldsymbol{\beta}\Bigg]\Bigg]\end{aligned} \tag{B.4}$$

where $\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}/\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$ is expressed as

$$\frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = \frac{\partial}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right]\right] = s\,\mathbf{I} + \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right]\otimes\frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} \tag{B.5}$$

Considering Eq. (19), $\mathrm{d}s/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$ is derived from

$$\frac{\mathrm{d}f_s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = \left.\frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\right|_{s} + \left.\frac{\partial f_s}{\partial s}\right|_{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = 0 \rightarrow \tag{B.6}$$

$$\frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = -\left[\frac{\partial f_s}{\partial s}\right]^{-1}\frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} \tag{B.7}$$

where

$$\frac{\partial f_s}{\partial s} = 3\left[\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} - {}^{n}\boldsymbol{\beta}\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right]\right] \tag{B.8}$$

and

$$\frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = 3\,s\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] - {}^{n}\boldsymbol{\beta}\right] \tag{B.9}$$

Considering Eq. (26), $\mathrm{d}c/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$ is derived from

$$\frac{\mathrm{d}f_c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = \left.\frac{\partial f_c}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}}\right|_{c} + \left.\frac{\partial f_c}{\partial c}\right|_{\boldsymbol{\sigma}^{\mathrm{tr}}}\frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = 0 \rightarrow \tag{B.10}$$

$$\frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = -\left[\frac{\partial f_c}{\partial c}\right]^{-1}\frac{\partial f_c}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} \tag{B.11}$$

where

$$\frac{\partial f_c}{\partial c} = -3\left[\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta}\right] : \left[2\,G\,\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right] + c\left|\left|2\,G\,\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right|\right|^{2}\right] \tag{B.12}$$

and

$$\frac{\partial f_c}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} = 3\left[\mathbf{I}^{\mathrm{dev}} - c\left[2\,G\,\frac{\partial\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} - 2/3H_{\mathrm{kin}}\frac{\partial\mathfrak{g}_{b}^{\mathrm{NN}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}}\right]\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta} - c\left[2\,G\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right]\right] \tag{B.13}$$

$\mathrm{d}\mathfrak{g}_{b}^{\mathrm{NN}}/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$ is derived in a similar way as that for $\mathrm{d}\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$. It should be noted that the required derivatives of the outputs of the FFNNs, i.e., $\mathcal{NN}_{\mathfrak{s}_i,1}$, $\mathcal{NN}_{\mathfrak{s}_i,2}$, and $\mathcal{NN}_{\mathfrak{s}_i,3}$, with respect to the inputs have been calculated analytically using chain rule. Further details can be found in, e.g., [17].

Data availability

Data will be made available on request.

References

[1] C. Lun, Q. Kan, P.J. Mutton, G. Kang, W. Yan, An efficient computational approach to evaluate the ratcheting performance of rail steels under cyclic rolling contact in service, Int. J. Mech. Sci. 101–102 (2015) 214–226.
[2] H. Su, C.L. Pun, P. Mutton, Q. Kan, W. Yan, Numerical study on the ratcheting performance of heavy haul rails in curved tracks, Wear 436–437 (2019) 203026.
[3] M. Ghodrati, M. Ahmadian, R. Mirzaeifar, Three-dimensional study of rolling contact fatigue using crystal plasticity and cohesive zone method, Int. J. Fatigue 128 (2019) 105208.
[4] N. Talebi, B. Andersson, M. Ekh, K.A. Meyer, Influence of a highly deformed surface layer on RCF predictions for rails in service, Wear (2025) 206173.

[5] C. Karg, S. François, W. Haegeman, G. Degrande, Elasto-plastic long-term behavior of granular soils: Modelling and experimental validation, Soil Dyn. Earthq. Eng. 30 (8) (2010) 635–646.
[6] I.N. Vladimirov, M.P. Pietryga, S. Reese, Anisotropic finite elastoplasticity with nonlinear kinematic and isotropic hardening and application to sheet metal forming, Int. J. Plast. 26 (5) (2010) 659–687.
[7] M. Safaei, M.-G. Lee, W. De Waele, Evaluation of stress integration algorithms for elastic–plastic constitutive models based on associated and non-associated flow rules, Comput. Methods Appl. Mech. Engrg. 295 (2015) 414–445.
[8] D. Leidermark, K. Simonsson, Procedures for handling computationally heavy cyclic load cases with application to a disc alloy material, Mater. At High Temp. 36 (5) (2019) 447–458.
[9] N. Talebi, K.A. Meyer, M. Ekh, Cycle-domain plasticity modeling using neural networks and symbolic regression, Comput. Struct. 321 (2026) 108086.
[10] J.N. Fuhg, G. Anantha Padmanabha, N. Bouklas, B. Bahmani, W. Sun, N.N. Vlassis, M. Flaschel, P. Carrara, L. De Lorenzis, A review on data-driven constitutive laws for solids, Arch. Comput. Methods Eng. (2024) 1–43.
[11] J. Dornheim, L. Morand, H.J. Nallani, D. Helm, Neural networks for constitutive modeling: From universal function approximators to advanced models and the integration of physics: J. Dornheim et al., Arch. Comput. Methods Eng. 31 (2) (2024) 1097–1127.
[12] Y.M. Hashash, S. Jung, J. Ghaboussi, Numerical implementation of a neural network based material model in finite element analysis, Internat. J. Numer. Methods Engrg. 59 (7) (2004) 989–1005.
[13] U. Ali, W. Muhammad, A. Brahme, O. Skiba, K. Inal, Application of artificial neural networks in micromechanics for polycrystalline metals, Int. J. Plast. 120 (2019) 205–219.
[14] D. Huang, J.N. Fuhg, C. Weißenfels, P. Wriggers, A machine learning based plasticity model using proper orthogonal decomposition, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113008.
[15] R. Lourenço, A. Tariq, P. Georgieva, A. Andrade-Campos, B. Deliktaş, On the use of physics-based constraints and validation KPI for data-driven elastoplastic constitutive modelling, Comput. Methods Appl. Mech. Engrg. 437 (2025) 117743.
[16] E. Haghighat, S. Abouali, R. Vaziri, Constitutive model characterization and discovery using physics-informed deep learning, Eng. Appl. Artif. Intell. 120 (2023) 105828.
[17] P. Weber, W. Wagner, S. Freitag, Physically enhanced training for modeling rate-independent plasticity with feedforward neural networks, Comput. Mech. 72 (4) (2023) 827–857.
[18] P. Weber, J. Geiger, W. Wagner, Constrained neural network training and its application to hyperelastic material modeling, Comput. Mech. 68 (5) (2021) 1179–1204.
[19] F. Foroughi Boroujeni, A. Faramarzi, W. Cha, B. Ghiassi, A physics-informed evolutionary regression framework for constitutive modelling of sand, Comput. Geotech. 197 (2026) 108179.
[20] K.A. Meyer, F. Ekre, Thermodynamically consistent neural network plasticity modeling and discovery of evolution laws, J. Mech. Phys. Solids 180 (2023) 105416.
[21] J.N. Fuhg, C.M. Hamel, K. Johnson, R. Jones, N. Bouklas, Modular machine learning-based elastoplasticity: Generalization in the context of limited data, Comput. Methods Appl. Mech. Engrg. 407 (2023) 115930.
[22] A. Zhang, D. Mohr, Using neural networks to represent von mises plasticity with isotropic hardening, Int. J. Plast. 132 (2020) 102732.
[23] D.P. Jang, P. Fazily, J.W. Yoon, Machine learning-based constitutive model for J2-plasticity, Int. J. Plast. 138 (2021) 102919.
[24] P. Fazily, J.W. Yoon, Machine learning-driven stress integration method for anisotropic plasticity in sheet metal forming, Int. J. Plast. 166 (2023) 103642.
[25] W.G. Dettmer, E.J. Muttio, R. Alhayki, D. Perić, A framework for neural network based constitutive modelling of inelastic materials, Comput. Methods Appl. Mech. Engrg. 420 (2024) 116672.
[26] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[27] M. Eghbalian, M. Pouragha, R. Wan, A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity, Comput. Geotech. 159 (April) (2023).
[28] S. Rezaei, A. Moeineddin, A. Harandi, Learning solutions of thermodynamics-based nonlinear constitutive material models using physics-informed neural networks, Comput. Mech. 74 (2) (2024) 333–366.
[29] Y. Li, J.M. Zhang, R. Wang, A PINN-driven stress integration paradigm for high-fidelity soil constitutive model considering cyclic response, Comput. Geotech. 196 (2026) 108170.
[30] P. Zhang, Neural network based numerical integration for elastoplastic constitutive relations, Comput. Geotech. 190 (2026) 107721.
[31] P.E. Malleval, V. Matray, F. Amlani, R. Scanff, Accelerating nonlinear finite element analysis via residual-aware neural network constitutive models, Finite Elem. Anal. Des. 251 (July) (2025) 104431.
[32] K. Runesson, S. Sture, K. Willam, Integration in computational plasticity, Comput. Struct. 30 (1) (1988) 119–130.
[33] H.L. Schreyer, R.F. Kulak, J.M. Kramer, Accurate numerical solutions for elastic-plastic models, J. Press. Vessel. Technol. 101 (3) (1979) 226–234.
[34] D.R. Owen, E. Hinton, Finite Elements in Plasticity, McGraw-Hill, New York, 1980.
[35] P.J. Armstrong, C.O. Frederick, et al., A Mathematical Representation of the Multiaxial Bauschinger Effect, Vol. 731, Berkeley Nuclear Laboratories Berkeley, CA, 1966.
[36] N. Ohno, J.-D. Wang, Kinematic hardening rules with critical state of dynamic recovery, part I: formulation and basic features for ratchetting behavior, Int. J. Plast. 9 (3) (1993) 375–390.
[37] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, J. Comput. Phys. 318 (2016) 22–35.
[38] K. Garanger, J. Kraus, J.J. Rimoli, Symmetry-enforcing neural networks with applications to constitutive modeling, Extrem. Mech. Lett. 71 (2024) 102188.
[39] J.P. Boehler, On Irreducible Representations for Isotropic Scalar Functions, ZAMM - J. Appl. Math. Mech. / Z. Für Angew. Math. Und Mech. 57 (6) (1977) 323–327.
[40] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, J. Han, On the variance of the adaptive learning rate and beyond, 2019, arXiv:1908.03265.
[41] A. Paszke, Pytorch: An imperative style, high-performance deep learning library, 2019, arXiv:1912.01703.
[42] F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, Springer, 2019.
[43] M.B. Gorji, M. Mozaffar, J.N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, J. Mech. Phys. Solids 143 (2020) 103972.
[44] M. Rosenkranz, K.A. Kalina, J. Brummund, M. Kästner, A comparative study on different neural network architectures to model inelasticity, Internat. J. Numer. Methods Engrg. 124 (21) (2023) 4802–4840.
[45] X. Li, M. Ekh, J.C. Nielsen, Three-dimensional modelling of differential railway track settlement using a cycle domain constitutive model, Int. J. Numer. Anal. Methods Geomech. 40 (12) (2016) 1758–1770.
[46] A. Bernasconi, M. Filippini, S. Foletti, D. Vaudo, Multiaxial fatigue of a railway wheel steel under non-proportional loading, Int. J. Fatigue 28 (5–6) (2006) 663–672.
[47] D.S.S. Corp., Abaqus Analysis User's Manual, 2025, Providence, Rhode Island, USA, 2025, Available from Dassault Systèmes.