Machine Learning-Accelerated Time Integration of Plasticity Models

Nasrin Talebi[1,*], Magnus Ekh[1], Knut Andreas Meyer[1]
[1]Division of Computational Mechanics and Materials Engineering, Department of Mechanical Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden

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 the training data. Furthermore, the proposed framework enforces exact fulfillment of the plastic consistency condition. To evaluate the proposed framework, a prototype model with the von Mises yield criterion and nonlinear kinematic hardening is chosen. Only 10 cycles of multiaxial proportional loading are used to generate the training data. After evaluating the proposed framework in material point simulations, we incorporate it into finite element simulations to evaluate its accuracy and computational efficiency in a boundary value problem. The results from both material point and finite element simulations show a very promising numerical performance of the neural network-based time integrator. It provides very good accuracy and numerical stability, as well as a noticeable gain in computational time for a single strain increment per load segment.

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

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, 2, 3, 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 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 reduce the computational cost of constitutive modeling, especially when integrated into FE frameworks; 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, 13, 14]. In general, these approaches have low predictive accuracy beyond the training regime, as they do not account for the underlying physics [15].

*Corresponding author: nasrin.talebi@chalmers.se

To address this, several approaches have been proposed in the literature to satisfy physical principles in a weak or strong form. Inspired by the idea of Physics-Informed Neural Networks (PINNs) [16], Haghighat et al. [17] proposed a PINN-based constitutive modeling framework in which violations of elastoplastic inequality constraints are penalized in the loss function. In [18], the introduced physics-informed framework adhered to the second law of thermodynamics during training by penalizing violating analytical expressions for hypoplasticity. Weber et al. [19] extended the idea of constrained NNs in [20] to rate-independent plasticity by enforcing physical constraints, such as material tangent symmetry and material stability, directly via error terms in the loss function. Despite their impressive ability to describe material behavior, these approaches do not guarantee physical constraints by construction. Instead, these constraints are only weakly enforced in the loss function during training. In contrast, Meyer and Ekre [21] proposed a framework in which Feed Forward Neural Networks (FFNNs) are embedded directly into the evolution equations for internal variables. Independent of the training, this formulation inherently satisfies physical requirements, such as thermodynamical consistency and objectivity. Using an alternative approach, Fuhg et al. [22] introduced a thermodynamically consistent framework based on isotropic and nonlinear kinematic hardening potentials.

ML-based approaches have recently offered a promising route to accelerate the numerical integration of plasticity models, in which the local solution of nonlinear evolution equations at each integration point dominates the computational cost. The following provides elaboration on some relevant studies from the literature focused on this objective. Zhang and Mohr [23] reformulated the classical return-mapping algorithm for von Mises plasticity using nonlinear isotropic hardening with explicit integration in an approximate manner. They employed an NN to estimate the tangent stiffness without imposing a priori assumptions about the yield surface, flow rule, or hardening law. In [24], during plastic loading, the stress integration algorithm was replaced with an NN in principal stress space, employing plane stress von Mises plasticity with nonlinear isotropic hardening under monotonic and cyclic loading. Further, they extended the approach to anisotropic plasticity for sheet metal forming applications [25]. Dettmer et al. [26] presented a framework for the stress update procedure that separates the state variable update from stress prediction, using state and response NN, respectively. The NNs were trained and evaluated based on cyclic data from uniaxial elastoplasticity with linear isotropic hardening, a uniaxial elastoplastic damage model, and plane strain elastoplasticity with tensor components used as input and output features. Several PINN-type surrogate models have been investigated in literature for bypassing the need for solving the nonlinear system of equations at the material point level. Examples include the work of Eghbalian et al. [27], who proposed a PINN-based surrogate model to replace an elastoplastic material model using incremental data for monotonic loading in principal stress space. Moreover, Rezaei et al. [28] adopted an unsupervised PINN-based approach to train a surrogate model for accelerating the incrementation of elastoplastic material models. They evaluated the methodology on a one-dimensional plasticity model with nonlinear isotropic hardening and on a three-dimensional interface damage model for cracking behavior. In contrast to purely surrogate-based formulations, Li et al. [29] introduced a PINN-driven framework that circumvents only the local stress integration in an elastoplastic soil constitutive model under cyclic loading. Aiming to fulfill the consistency condition during plastic loading, Zhang [30] presented an unsupervised learning framework in which an NN is leveraged to predict the stress increment accurately during plastic loading in each strain increment. This objective was achieved by minimizing a loss function that incorporates the residuals of the yield function and the stress increment. Recently, Malleval et al. [31] proposed using NNs to approximate the solution to a scalar nonlinear constitutive equation at each integration point for an elastoviscoplastic material model.

The core idea of this study is to propose an NN-based explicit time integration algorithm that enables robust integration over very large strain increments within a physics-encoded framework. The framework is designed such that loading and unloading conditions are fulfilled by construction. Training data are generated from highly resolved reference solutions obtained using a plasticity model with nonlinear kinematic hardening under multiaxial proportional loading. To ensure frame-independent predictions and reduce the dimensions of the NNs, the models are formulated with invariants as input features, combined with evolution directions inferred from the training data. Furthermore, the consistency condition during plastic loading is enforced in the framework by introducing a correction factor. Finally, we embed the proposed framework into FE examples to assess its numerical performance and computational efficiency for boundary value problems under different strain increment sizes.

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

2 ML-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 ML-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, $\mathbb{S} = \{\mathfrak{s}_1,\ \mathfrak{s}_2,\ \cdots,\ \mathfrak{s}_{N_{\mathrm{s}}}\}$,

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

The stress, $\boldsymbol{\sigma}$, and the (hardening) variables, $\mathbb{A} = \{\mathfrak{a}_1,\ \mathfrak{a}_2,\ \cdots,\ \mathfrak{a}_{N_{\mathrm{a}}}\}$, 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, $\Phi\left(\boldsymbol{\sigma}, \mathbb{A}\right)$, 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

$$\Phi \leq 0, \quad \dot{\lambda} \geq 0, \quad \Phi\,\dot{\lambda} = 0 \tag{2}$$

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

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

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.[2] 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 Equations 2 and 3. Different time integration schemes, $\mathfrak{g}_i^{\mathrm{TI}}$, can be used to approximate the solution to Equation 3

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

For implicit algorithms, $\mathfrak{g}_i^{\mathrm{TI}}$ can typically not 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

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

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 (Equation 2) is replaced with $\Phi = 0$, which, coupled with Equation 3, form 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] 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.

2.3 ML-accelerated integration of a generic small-strain plasticity model

The idea of this paper is to use ML to find an explicit time integrator, $\mathfrak{g}^{\mathrm{ML}}_{\mathfrak{s}_i}$, that can handle large strain increments with good accuracy and numerical stability. Specifically, we propose to use NNs to formulate $\mathfrak{g}^{\mathrm{ML}}_{\mathfrak{s}_i}$ for each state variable, $\mathfrak{s}_i$, as

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

where $\mathcal{NN}$ are scalar outputs of an NN, and $\mathbb{I} = \{I_1,\ I_2,\ \cdots,\ I_N\}$ is a set of suitable invariants that depends on $\boldsymbol{\sigma}^{\mathrm{tr}}$, $\boldsymbol{\sigma}^{\mathrm{lim}}$, and ${}^n\mathfrak{a}_j$. The choice to express the NNs in invariants eliminates coordinate system dependence and lowers their dimensions.

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

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

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

3 Prototype model: Plasticity model with kinematic hardening

This section first presents the prototype plasticity model, which will be used to train and evaluate the proposed NN-based time integration scheme. This is followed by a detailed description of the NN-based integration.

3.1 Prototype plasticity model

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

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

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

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

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

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

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

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

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

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

The evolution of the back-stress, $\boldsymbol{\beta}$, is assumed to follow the Armstrong-Frederick kinematic hardening law [35] as

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

Table 1: Material model parameter values for the prototype plasticity model.

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

β_∞ is the saturation value of the back-stress. The material model parameter values are presented in Table 1, which are inspired by those calibrated against experimental data for isotropic R260 pearlitic rail steel [4]. The prototype material model, integrated with the implicit Backward Euler time integration algorithm to solve the set of coupled DAEs in Equations 2, 15, and 16, is employed for training data generation and performance evaluation of the prototype model, integrated with the NN-based integration scheme in material point simulations; see Sections 4.2 and 5.1 respectively for further details.

3.2 NN-based integration of the prototype plasticity model

When integrating the prototype plasticity model 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 $\Phi^{\text{tr}} < 0$, 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 $\Phi^{\text{tr}} \geq 0$, 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 Figure 1. However, even if ${}^n\Phi = 0$, the first part could be elastic due to a load direction change, see the right part of Figure 1. We detect this case by evaluating if Φ decreases (i.e., becomes negative since the previous value was zero) at the beginning of the strain increment, specifically

$$\left.\frac{\partial\Phi({}^n\boldsymbol{\sigma} + \mathbf{E} : [\theta\Delta\boldsymbol{\epsilon}], {}^n\boldsymbol{\beta})}{\partial\theta}\right|_{\theta=0} \leq 0 \tag{17}$$

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 $\Phi(\boldsymbol{\sigma}^{\text{lim}}, {}^n\boldsymbol{\beta}) = 0$, resulting in

$$f_s = \frac{3}{2}\left|\left|{}^n\boldsymbol{\sigma}^{\text{dev}} + s\left[\boldsymbol{\sigma}^{\text{tr, dev}} - {}^n\boldsymbol{\sigma}^{\text{dev}}\right] - {}^n\boldsymbol{\beta}\right|\right|^2 - Y_0^2 = 0 \tag{18}$$

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 [36], using a suitable set of invariants as input features [21], or enforcing frame invariance at the neuron level [37]. 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

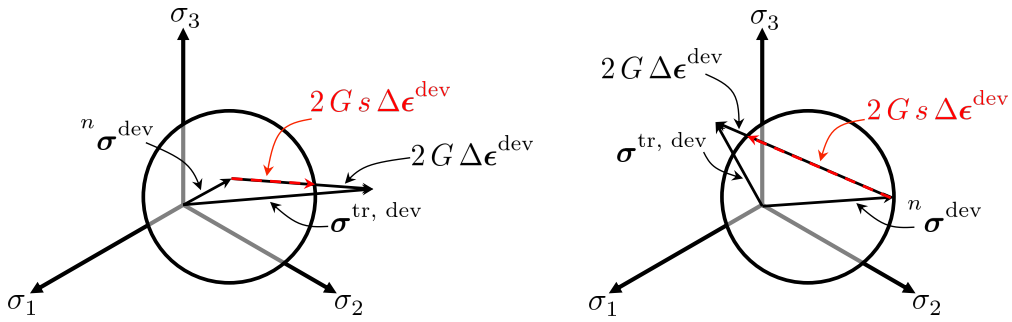


Figure 1: Schematic illustration of two cases in which the scalar variable s needs to be determined. The left and right figures correspond, respectively, to the case where the old material response is elastic and to the case where trial stress path crosses the yield surface. The illustration is presented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

than the first approach [36]. Following [38], possible choices of invariants considering the prototype model are

$$
\begin{array}{llll}
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{tr}}\right), & \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & \mathrm{tr}\left(\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^3\right), & \\
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{lim}}\right), & \boldsymbol{\sigma}^{\mathrm{lim,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,dev}}, & \mathrm{tr}\left(\left[\boldsymbol{\sigma}^{\mathrm{lim,dev}}\right]^3\right), & \\
\mathrm{tr}\left({}^{\mathrm{n}}\boldsymbol{\beta}\right), & {}^{n}\boldsymbol{\beta} : {}^{n}\boldsymbol{\beta}, & \mathrm{tr}\left(\left[{}^{\mathrm{n}}\boldsymbol{\beta}\right]^3\right), & \\
\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,dev}}, & \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \left[\boldsymbol{\sigma}^{\mathrm{lim,dev}}\right]^2, & \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2 : \boldsymbol{\sigma}^{\mathrm{lim,dev}}, & \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2 : \left[\boldsymbol{\sigma}^{\mathrm{lim,dev}}\right]^2, \\
{}^{n}\boldsymbol{\beta} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & {}^{n}\boldsymbol{\beta} : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2, & \left[{}^{n}\boldsymbol{\beta}\right]^2 : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & \left[{}^{n}\boldsymbol{\beta}\right]^2 : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}\right]^2, \\
\boldsymbol{\sigma}^{\mathrm{lim,dev}} : {}^{n}\boldsymbol{\beta}, & \boldsymbol{\sigma}^{\mathrm{lim,dev}} : \left[{}^{n}\boldsymbol{\beta}\right]^2, & \left[\boldsymbol{\sigma}^{\mathrm{lim,dev}}\right]^2 : {}^{n}\boldsymbol{\beta}, & \left[\boldsymbol{\sigma}^{\mathrm{lim,dev}}\right]^2 : \left[{}^{n}\boldsymbol{\beta}\right]^2, \\
\mathrm{tr}\left(\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} \cdot \boldsymbol{\sigma}^{\mathrm{lim,dev}} \cdot {}^{\mathrm{n}}\boldsymbol{\beta}\right) & & &
\end{array}
\tag{19}
$$

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

$$
\begin{array}{lll}
J_{\mathrm{tr,\,tr}} = \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & J_{\mathrm{lim,\,lim}} = \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & J_{\beta,\,\beta} = {}^{n}\boldsymbol{\beta} : {}^{n}\boldsymbol{\beta}, \\
J_{\mathrm{tr,\,lim}} = \boldsymbol{\sigma}^{\mathrm{tr,\,dev}} : \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, & J_{\beta,\,\mathrm{tr}} = {}^{n}\boldsymbol{\beta} : \boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, & J_{\mathrm{lim},\,\beta} = \boldsymbol{\sigma}^{\mathrm{lim,\,dev}} : {}^{n}\boldsymbol{\beta}
\end{array}
\tag{20}
$$

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

For training of the NNs, it is vital that inputs and outputs are scaled appropriately. With the chosen set of invariants, we obtain the following formulation for the scaled NN-based time integrators, $\hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathfrak{s}_i}$,

$$
\hat{\mathfrak{g}}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} = \mathcal{NN}_{\epsilon^{\mathrm{p}},\,1}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \mathcal{NN}_{\epsilon^{\mathrm{p}},\,2}\left(\hat{\mathbb{J}}\right)\,\hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \mathcal{NN}_{\epsilon^{\mathrm{p}},\,3}\left(\hat{\mathbb{J}}\right)\,{}^{n}\hat{\boldsymbol{\beta}} \tag{21}
$$

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

with the unscaled time integrators,

$$
\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} = \mathtt{SF}_{\epsilon^{\mathrm{p}}}\,\hat{\mathfrak{g}}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}}, \quad \mathfrak{g}^{\mathrm{NN}}_{b} = \mathtt{SF}_{b}\,\hat{\mathfrak{g}}^{\mathrm{NN}}_{b} \tag{23}
$$

All scaled variables are denoted by a hat, $\hat{\bullet}$, and are scaled by the values in the training data: all invariants in $\mathbb{J}$ are individually normalized by the range found in the training 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 in the training 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 training data.

Finally, the increments of the state variables are calculated as

$$
\Delta\boldsymbol{\epsilon}^{\mathrm{p}} = c\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} \;\text{ and }\; \Delta\boldsymbol{b} = c\,\mathfrak{g}^{\mathrm{NN}}_{b} \tag{24}
$$

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

$$
f_c = \frac{3}{2}\left|\left|\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta} - c\,\left[2\,G\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}^{\mathrm{NN}}_{b}\right]\right|\right|^2 - Y_0^2 = 0 \tag{25}
$$

where we also adopt $\Delta\boldsymbol{\beta} = -2/3\,H_{\mathrm{kin}}\,\Delta\boldsymbol{b}$ from Equation 16. We choose the solution of c from Equation 25 that is closest to unity. Note that, during the training of the neural networks, $c = 1$ is assumed. The overall numerical algorithm based on the proposed NN-based integration scheme is presented in Algorithm 1.

3.3 Evaluation of the considered evolution directions

The chosen six invariants in $\mathbb{J}$ result in three evolution directions, $\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$, $\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}$ and ${}^{n}\boldsymbol{\beta}$; see Equation 9. By analyzing the training data, 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.

For each training data, we have given $\Delta\boldsymbol{\mathfrak{s}}^{\mathrm{target}}_{i}$, 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 Equations 21- 24, we can determine coefficients α_{1i} - α_{3i} by a least square fit

$$
\min\;\left|\left|\Delta\boldsymbol{\mathfrak{s}}^{\mathrm{target}}_{i} - \mathtt{SF}_{\boldsymbol{\mathfrak{s}}_i}\left[\alpha_{1i}\,\hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \alpha_{2i}\,\hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \alpha_{3i}\,{}^{n}\hat{\boldsymbol{\beta}}\right]\right|\right|^2 \tag{26}
$$

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

Compute Φ^{tr}
if $\Phi^{\mathrm{tr}} < 0$ **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 Equation 18
    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 Equation 25
  Update the state variables and stresses
    $\boldsymbol{\epsilon}^{\mathrm{p}} = {}^{n}\boldsymbol{\epsilon}^{\mathrm{p}} + c\,\mathfrak{g}^{\mathrm{NN}}_{\epsilon^{\mathrm{p}}}$
    $\boldsymbol{\beta} = {}^{n}\boldsymbol{\beta} - \frac{2}{3}\,H_{\mathrm{kin}}\,c\,\mathfrak{g}^{\mathrm{NN}}_{b}$
    $\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathrm{lim}} + \mathbf{E}^{\mathrm{e}} : [\Delta\boldsymbol{\epsilon} - \Delta\boldsymbol{\epsilon}^{\mathrm{p}}]$

With the identified coefficients, the predicted increments are obtained as

$$\Delta\mathfrak{s}_i^{\mathrm{pred}} = \mathtt{SF}_{\mathfrak{s}_i}\left[\alpha_{1i}\,\hat{\boldsymbol{\sigma}}^{\mathrm{tr,\,dev}} + \alpha_{2i}\,\hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} + \alpha_{3i}\,{}^{n}\hat{\boldsymbol{\beta}}\right] \tag{27}$$

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

$$\theta_i = \cos^{-1}\left(\frac{\Delta\mathfrak{s}_i^{\mathrm{target}} : \Delta\mathfrak{s}_i^{\mathrm{pred}}}{\left|\left|\Delta\mathfrak{s}_i{}^{\mathrm{target}}\right|\right|\left|\left|\Delta\mathfrak{s}_i{}^{\mathrm{pred}}\right|\right|}\right) \tag{28}$$

If θ_i is close to 0, then the selected evolution directions can accurately express the training data; otherwise, additional directions should be introduced. We have compared the resulting θ_i values for four sets of evolution directions. Specifically, sets 1-4 are $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$, $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right\}$, $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$, and $\left\{\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$. The computed θ_i values in terms of relative number of occurrences considering $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ are illustrated in Figure 2, showing that using all the 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 B. Table 2 summarizes the maximum angular errors, where the maximum errors for set 1 are comparable to the numerical precision.

4 Neural networks and training

This section describes the NN model employed in the study to formulate $\mathfrak{g}^{\mathrm{NN}}_{\mathfrak{s}_i}$, the approach adopted to generate training data, and the 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 $\mathcal{NN}_{\mathfrak{s}_i,1}$ -

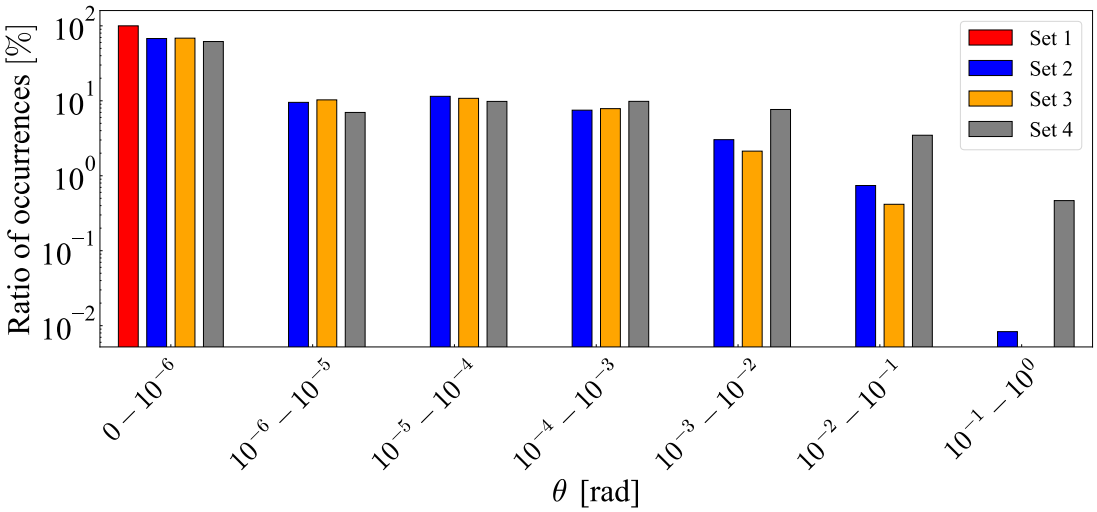


Figure 2: The distribution of the computed angles, θ, between the target and predicted $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ considering all training data for four sets of evolution directions: set 1: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\ dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\ dev}}, {}^{n}\boldsymbol{\beta}\right\}$, set 2: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\ dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\ dev}}\right\}$, set 3: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\ dev}}, {}^{n}\boldsymbol{\beta}\right\}$, and set 4: $\left\{\boldsymbol{\sigma}^{\mathrm{lim,\ dev}}, {}^{n}\boldsymbol{\beta}\right\}$.

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

Sets	Set 1	Set 2	Set 3	Set 4
θ (for $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$)	2.98×10^{-8}	1.00×10^{-1}	2.56×10^{-2}	1.48×10^{-1}
θ (for $\Delta\boldsymbol{b}$)	2.98×10^{-8}	1.38×10^{-1}	1.60×10^{-2}	1.11×10^{-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 30000 epochs. The FFNNs' weights and biases are randomly initialized before being optimized with the gradient-based Rectified Adam (RAdam) optimizer [39] in the PyTorch library [40], 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 [41].

To facilitate the optimization of the network parameters, i.e., weights and biases, during training [42], and to prevent vanishing gradients caused by large input values to the activation function, $\tanh(x)$ [43], scaling has been introduced as described in Section 3.2. In Equations 21 and 22, we use one FFNN to formulate the increment of each state variable, $\Delta\mathfrak{s}_i$, as illustrated in Figure 3. 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 better training performance. After using the outputs of the FFNNs in Equations 21 and 22 to predict $\hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathfrak{s}_i}$, the mean squared error type loss, $\mathcal{L}_{\mathfrak{s}_i}$, can be calculated as

$$\mathcal{L}_{\mathfrak{s}_i} = \frac{1}{N_{\mathrm{data}}} \sum_{j=1}^{N_{\mathrm{data}}} \left[[\Delta\hat{\mathfrak{s}}_i]_j^{\mathrm{target}} - \hat{\mathfrak{g}}^{\mathrm{NN}}_{\mathfrak{s}_i} \right]^2 + \lambda_{\mathrm{p}} \left|\left| \boldsymbol{p}_{\mathfrak{s}_i} \right|\right|_2^2 \tag{29}$$

Here, N_{data} denotes the total number of training data samples, and $[\Delta\hat{\mathfrak{s}}_i]^{\mathrm{target}}$ is the target state variable increments, normalized with $\mathtt{SF}_{\mathfrak{s}_i}$. We apply L2 regularization to enhance the generalizability of the FFNNs [44, 45]. This is done by augmenting the loss function with a differentiable penalty term corresponding to the square of the L2-norm of the network parameters $\boldsymbol{p}$, where $\lambda_{\mathrm{p}} = 10^{-6}$.

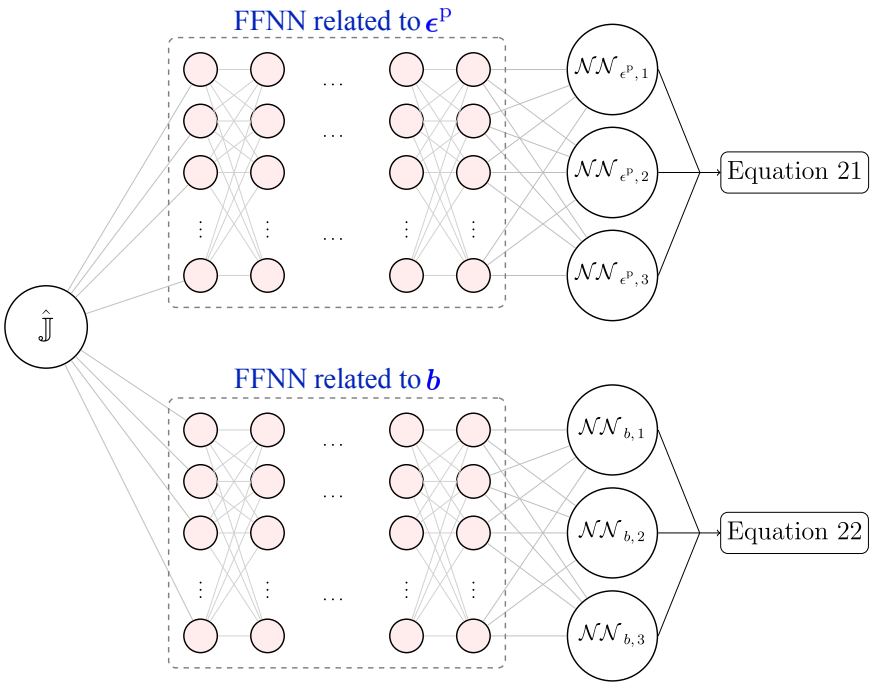


Figure 3: Schematic illustration of the inputs and outputs of the NNs, which are embedded in Equations 21 and 22.

4.2 Training data generation

To train the FFNNs, we generate data using the prototype plasticity model presented in Section 3.1, integrated with 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. A single deviatoric multiaxial pulsating strain path is generated over 10 loading cycles, each discretized into 400 time steps; see Figure 4a. 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 [46, 47, 9]. The strain histories are used in material point simulations, and the resulting stress-strain cycles are employed to generate training data samples.

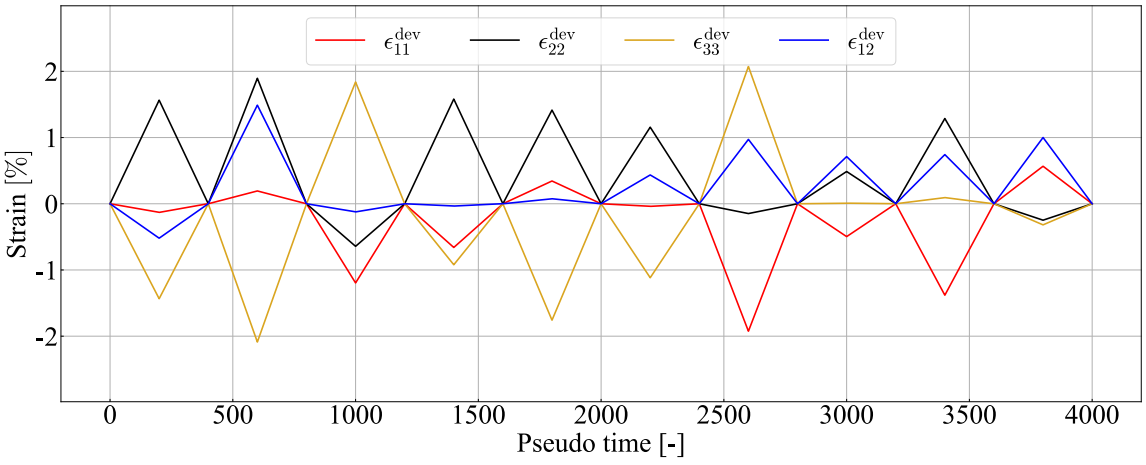
We adopt an approach similar to that described in [19] to generate training data, which is schematically presented in Figure 4b. We first split the data into load segments with linear strain paths, which, in this case, correspond to half-cycles. Next, data points corresponding to elastic material behavior, 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 Figure 4b are the first points used for training 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

$$\begin{aligned} &\boldsymbol{\sigma}^{\mathrm{tr,\ dev}},\ \boldsymbol{\sigma}^{\mathrm{lim,\ dev}},\ {}^{n}\boldsymbol{\beta} \quad \text{where} \quad n = k - \Delta k \\ &\mathbb{J} = \{J_{\mathrm{tr,\,tr}},\ J_{\mathrm{lim,\,lim}},\ J_{\beta,\,\beta},\ J_{\mathrm{tr,\,lim}},\ J_{\beta,\,\mathrm{tr}},\ J_{\mathrm{lim},\,\beta}\} \\ &\Delta\boldsymbol{\epsilon}^{\mathrm{p}},\ \Delta\boldsymbol{b} \end{aligned} \tag{30}$$

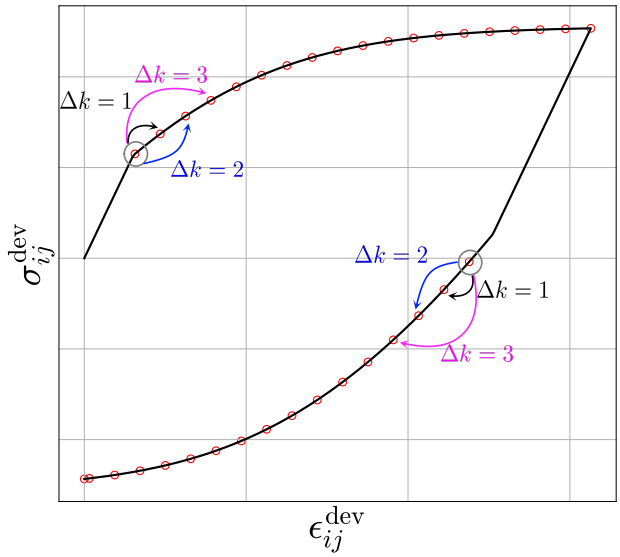
Since the aim of the NN-based integration scheme is to accelerate cyclic simulations by allowing large $\Delta\boldsymbol{\epsilon}$, we have not used all of the possible data points in each cycle for data generation. Instead, the data points are taken every 5 increments, which also results in more efficient training. The training data generation strategy has produced 12012 training data samples.

4.3 FFNN training

The evolutions of the loss functions, $\mathcal{L}_{\epsilon^{\mathrm{p}}}$ and $\mathcal{L}_b$, during the training procedure are shown in Figure 5. In the last epoch, $\mathcal{L}_{\epsilon^{\mathrm{p}}}$ and $\mathcal{L}_b$ reach their lowest values of 1.42×10^{-5} and 1.93×10^{-5}, respectively. More training did not result in a significant reduction in the training losses.



(a)



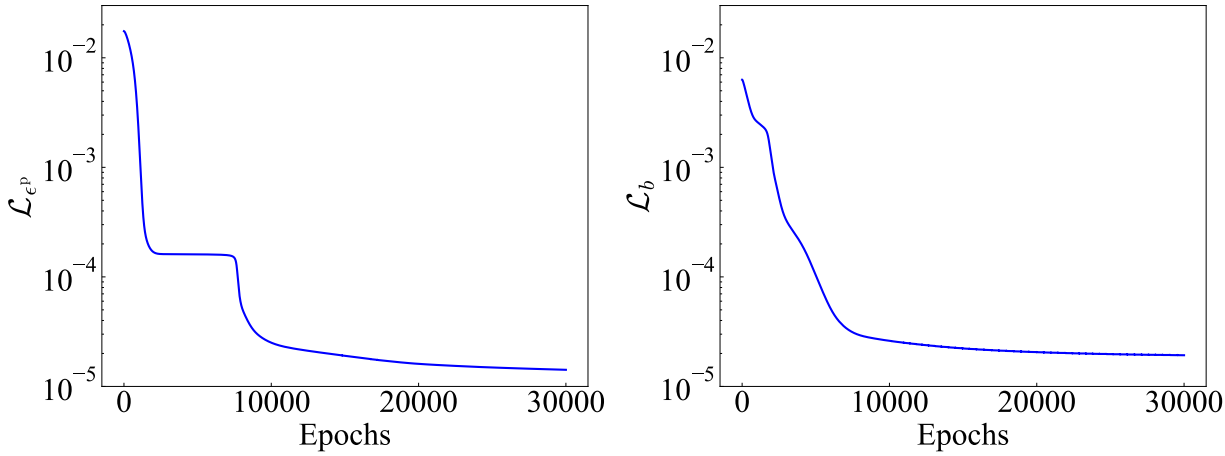
(b)

Figure 4: (a) Deviatoric multiaxial pulsating strain path for 10 loading cycles and (b) schematic illustration of data generation approach after excluding the data points with elastic material behavior.

Figures 6a and 6b present the distribution of the norms of absolute errors in $\Delta\boldsymbol{\epsilon}^{\text{p}}$ and $\Delta\boldsymbol{b}$ for all training data with different strain increment sizes. 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 $\left|\left|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right|\right| < 0.01$, although they occur with low frequency. As $\left|\left|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right|\right|$ increases, the maximum error decreases. This trend breaks for $\left|\left|\Delta\boldsymbol{\epsilon}^{\text{dev}}\right|\right| > 0.03$, where there is less training data, and the accuracy of the trained FFNNs decreases.

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. In the following, for both simulations, the reference model denotes the model that uses the prototype material model, integrated with the implicit backward Euler time integration algorithm. By using very small time steps, the reference model provides the true solution to the



(a) (b)

Figure 5: Loss function evolutions considering (a) $\mathcal{L}_{\epsilon^{\mathrm{P}}}$ and (b) $\mathcal{L}_b$ during training.

DAEs that are being solved. The NN-based model refers to the model that also adopts the prototype material model, but is integrated with the proposed NN-based integration. The NN-based model has been implemented according to Algorithm 1, with the trained FFNNs embedded into Equations 21 and 22.

5.1 Material point simulations

Figure 7a presents the deviatoric axial responses over the first 5 cycles for the reference and NN-based models, considering the material point simulation training data. The predictions of the NN-based model, using 5 strain increments and even a single increment per half-cycle, are in close agreement with those of the reference model, in which each half-cycle is discretized into 200 increments.

To investigate the influence of not accounting for the correction factor, c, i.e., $c = 1$ during the simulations, on the predictions of the NN-based model, we have calculated the errors in $||\Delta\boldsymbol{\epsilon}^{\mathrm{P}}||$ as $[c-1]\ ||\Delta\boldsymbol{\epsilon}^{\mathrm{P}}||$ and presented them in Figure 8. Considering the case with 5 strain increments per half-cycle, the c factors have not been calculated in every strain increment within a cycle, since some increments correspond to elastic loading/unloading, e.g., in cycle 4. In contrast, in the case with 1 increment per half-cycle, they have been computed in every increment. The errors arising from the assumption of $c = 1$ during plastic loading are more significant in the case using 1 increment with larger $||\Delta\boldsymbol{\epsilon}^{\mathrm{P}}||$. The consequences of this assumption in Equation 24 are further shown in Figure 9, which compares the von Mises stress values, σ^{vM}, at the peak strain of each cycle from the reference and NN-based models. 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 the last cycle for the case with 5 increments per half-cycle, ${}^{n}\Phi > 0$, which is inconsistent. Although Equation 17 was fulfilled, no solution for the s variable using Equation 18 was found. These results highlight the importance of the c factor for our proposed NN-based time integration framework. Hereafter, the discussion is based on the results obtained when c is calculated during plastic loading.

Considering Figure 9, the simulation of 10 cycles has resulted in a maximum absolute error of 4.8 MPa for the NN-based model with 5 strain increments per half-cycle and 22.6 MPa with 1 increment. Given that the maximum σ^{vM} predicted by the reference model is approximately 900 MPa, the results demonstrate a very good predictive capability of the NN-based model. Moreover, despite the NN-based time integration being explicit, we note that it provides good accuracy for very large strain increments. Additional results for the shear responses are provided in Appendix B.

To evaluate the extrapolation performance of the NN-based model, we have simulated 100 cycles with randomly generated variable strain ranges. Figure 7b shows an overall good fit with the reference results considering both 5 and 1 strain increments per half-cycle. This good agreement can also be observed in Figure 10, 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 49.3 MPa and 60.0 MPa for 5 and 1 strain increments per half-cycle, respectively, relative to the maximum σ^{vM} of about 900 MPa. The distribution of absolute errors in σ^{vM} at the peak strains over 100 cycles, presented in Figure 11, shows that most errors are distributed within the lowest interval of 0-10 MPa. Despite that 6% of the cases considering

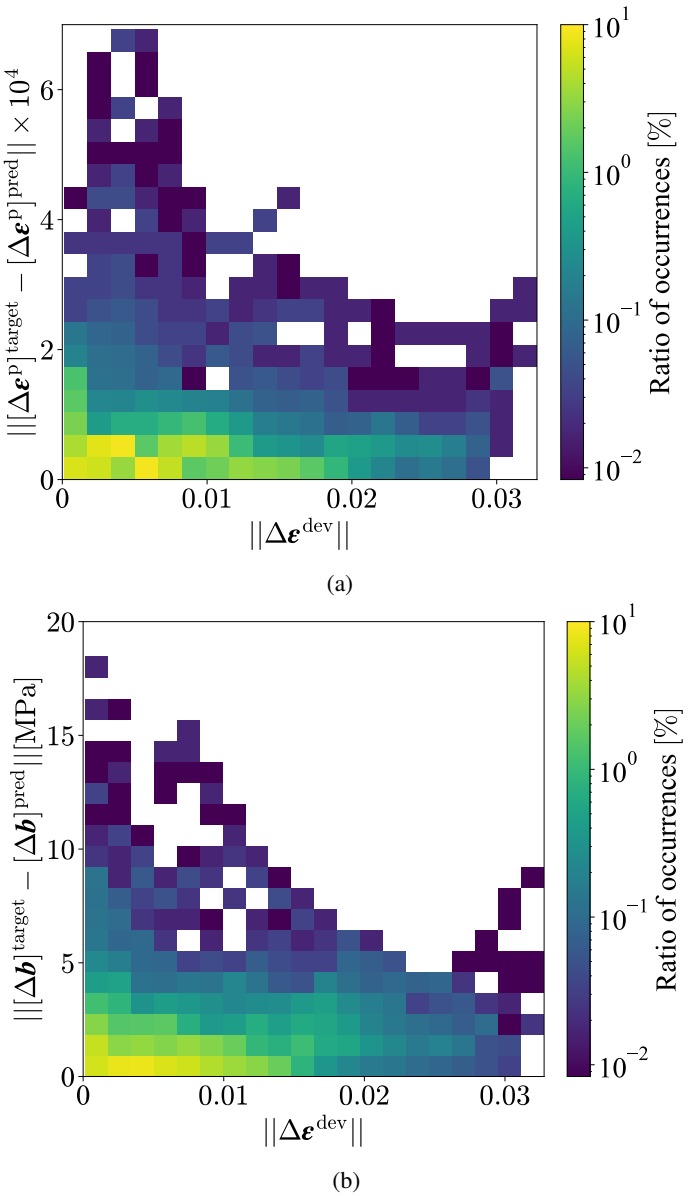
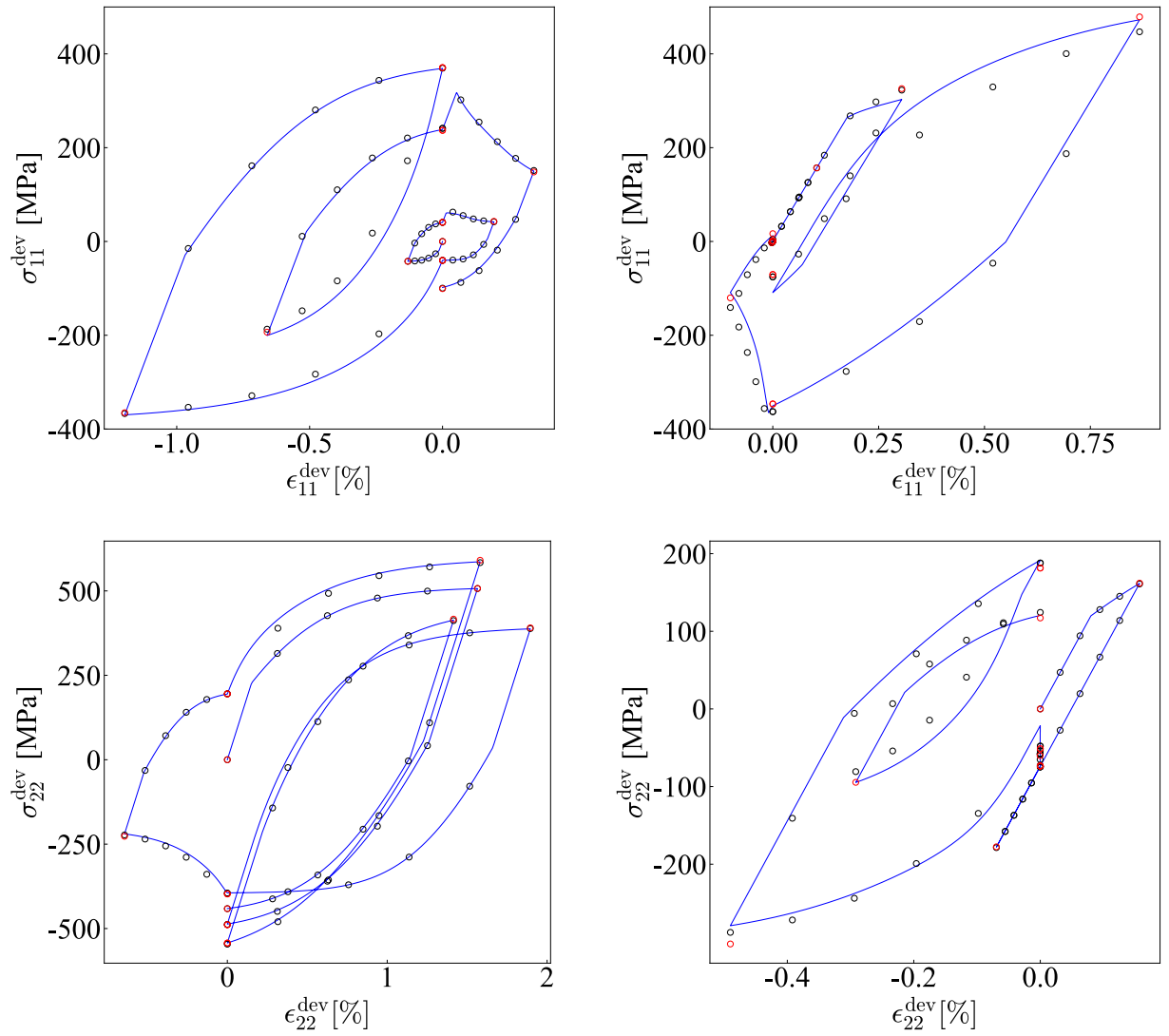


Figure 6: Two-dimensional histograms of the norm of absolute errors in (a) $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ and (b) $\Delta\boldsymbol{b}$ versus $\left|\left|\Delta\boldsymbol{\epsilon}^{\mathrm{dev}}\right|\right|$, considering the training data. White bins correspond to zero occurrences.

1 increment have resulted in absolute errors larger than 40 MPa compared to 1% for 5 increment cases, both show promising predictions.

5.2 FE simulations

To demonstrate the applicability of the NN-based time integration in boundary value problems, we have incorporated it into the commercial FE code Abaqus [48]. The FE model is a quarter of a rectangular plate with a central circular hole. The plate geometry and considered pulsating displacement-control loading with varying ranges are shown in Figure 12. 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. Considering the reference FE model, each half-cycle is discretized into 100 displacement increments over the pseudo time interval from 0 to 1. The results from the NN-based FE model, using 5 and 1 displacement increments per half-cycle, are compared against those from the reference FE model. The material models have been implemented in Abaqus as



(a) Training data

(b) Test data

Figure 7: The axial stress-strain responses from the reference material model (blue) and the NN-based model. The black and red circles correspond to the results from the NN-based model with 5 and 1 strain increments per half-cycle, respectively. Only the first 5 cycles are shown.

user-defined subroutines, and for the NN-based model, the displacement iterations have been performed using the algorithmic tangent stiffness presented in Appendix A. For both the reference and NN-based FE models, line search was enabled to aid convergence during the displacement iterations.

The von Mises stress, σ^{vM}, distributions in cycles 50 and 100 are illustrated in Figures 13a and 13c at the peak displacements for the reference and NN-based FE models. Overall, the results from the latter show a close agreement with the reference FE model results in both cycles. To further assess the differences between the models, spatial distributions of the absolute errors in σ^{vM} are presented in Figures 13b and 13d. In cycle 50, the maximum absolute errors are 8.5 MPa and 3.6 MPa for the NN-based model using 5 and 1 increments per half-cycle, respectively. The corresponding errors in cycle 100 are 4.4 MPa and 8.0 MPa. Relative to the maximum σ^{vM} values in the plate, approximately 732 MPa in cycle 50 and 821 MPa in cycle 100, these errors show that the FE models employing the NN-based integration with both increment numbers provide accurate predictions. To showcase the predictions of the NN-based model over the 100 simulated cycles in the highly stressed region near the hole, Figure 14 compares σ^{vM} values from the reference and NN-based FE models at the peak displacements for the critical element indicated in

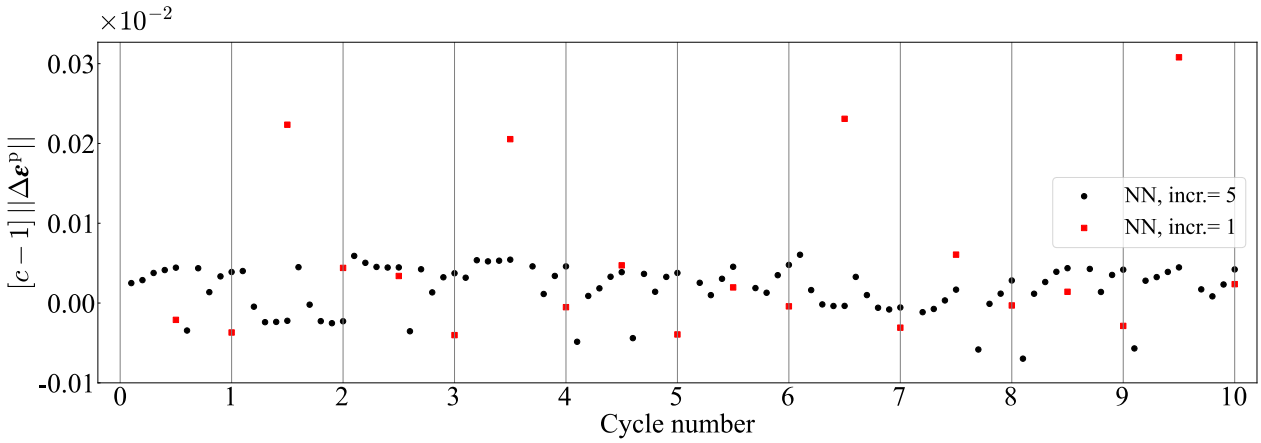


Figure 8: Calculated errors in $||\Delta\boldsymbol{\epsilon}^{\mathrm{p}}||$ due to assuming $c = 1$, i.e., not enforcing the consistency condition, in each cycle when taking 5 and 1 strain increments per half-cycle (incr.), using the training data. NN refers to the NN-based model. The vertical dashed lines indicate the boundaries between consecutive cycles.

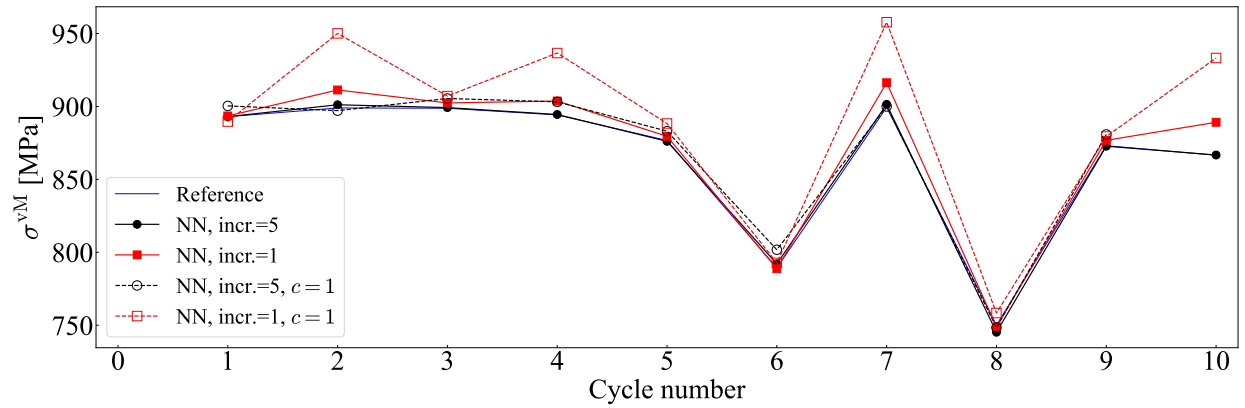


Figure 9: von Mises stresses at the peak strains over 10 cycles considering the training data. NN refers to the NN-based model, and incr. denotes the number of increments per half-cycle. $c = 1$ means that the consistency condition has not been enforced.

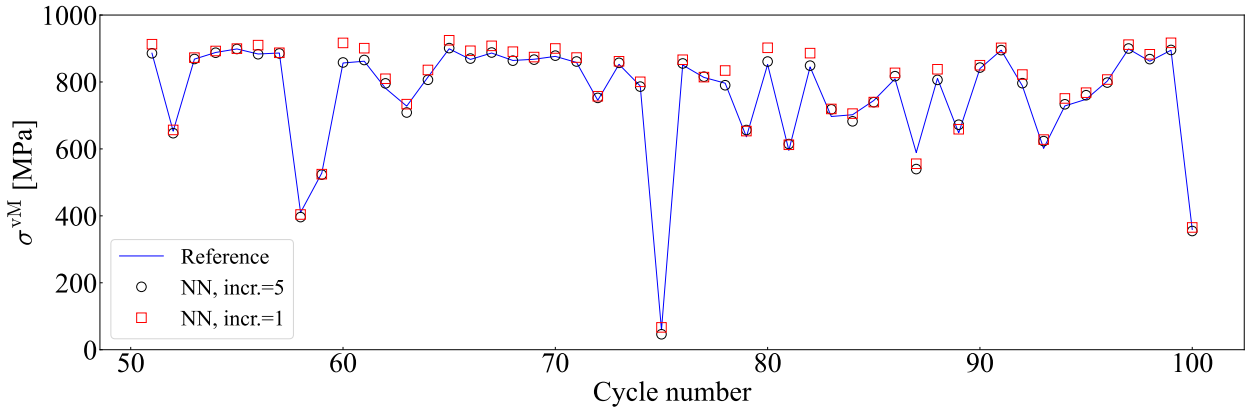


Figure 10: von Mises stresses at the peak strains for the test data (the last 50 of 100 simulated cycles are shown).

Figure 12. Despite the maximum absolute errors of 3.6 MPa and 3.5 MPa for 5 and 1 displacement increments per half-cycle, the predictions of the explicit NN-based integration are very promising. Further, the strain histories during the last cycle for the critical element are shown in Figure 15a, comparing the reference model and the NN-based FE

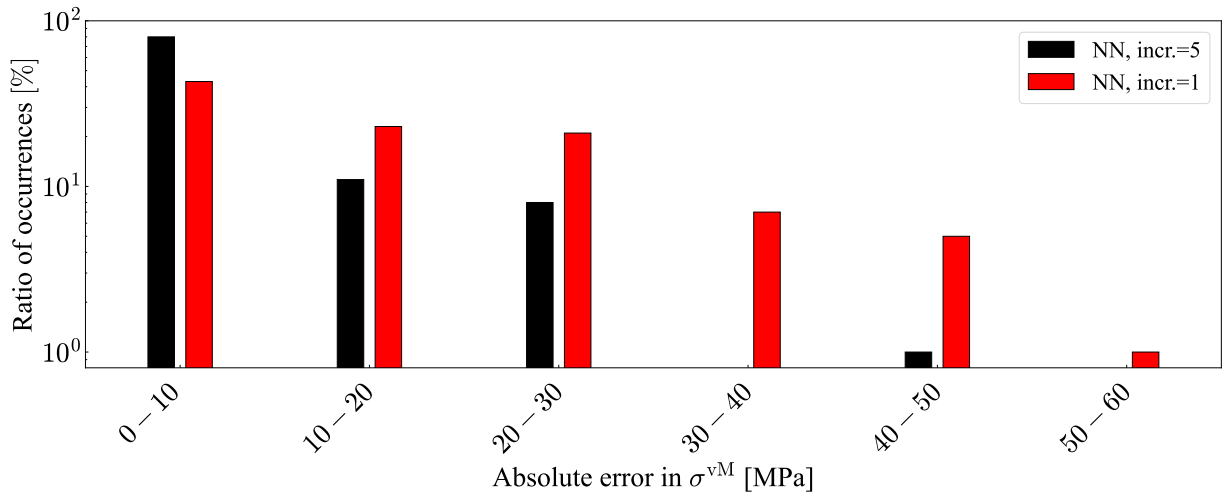


Figure 11: absolute error distribution in $\boldsymbol{\sigma}^{\mathrm{vM}}$, when comparing the results at the peak strains from the NN-based model with those from the reference model. incr. denotes the number of increments per half-cycle.

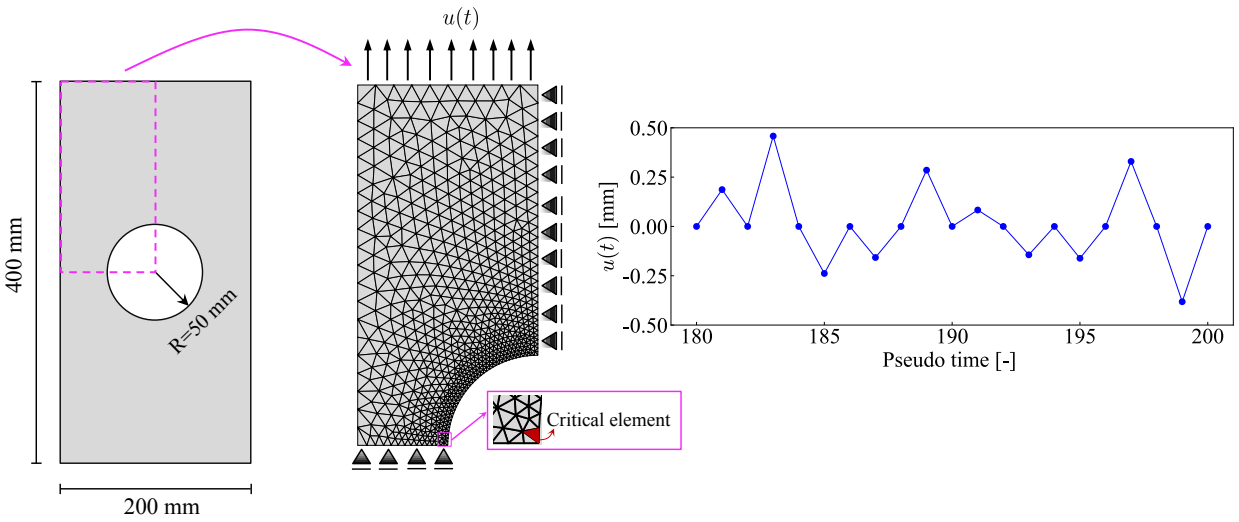


Figure 12: Geometry and boundary condition for the plate with a hole under displacement-controlled pulsating loading. The last 10 of 100 simulated cycles are shown.

model 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 $\boldsymbol{\epsilon}$ within a time step. Nevertheless, as shown in Figure 15b, the predicted stresses obtained with the NN-based model are in close agreement with those from the reference model. It should be noted that σ_{11} and σ_{12} tend toward zero as the mesh is further refined near the hole.

Figure 16 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 85% to 90% 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 18.1 MPa and 14.3 MPa, respectively. Compared to the maximum σ^{vM} of 885.3 MPa in the reference model, these errors indicate accurate predictions of the NN-based model.

Finally, the computational efficiency of the FE simulations using the reference and NN-based models has been compared in terms of Central Processing Unit (CPU) time. Each FE simulation was conducted 5 times, and the averaged results are presented in Table 3. 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 integration with 5 and 1

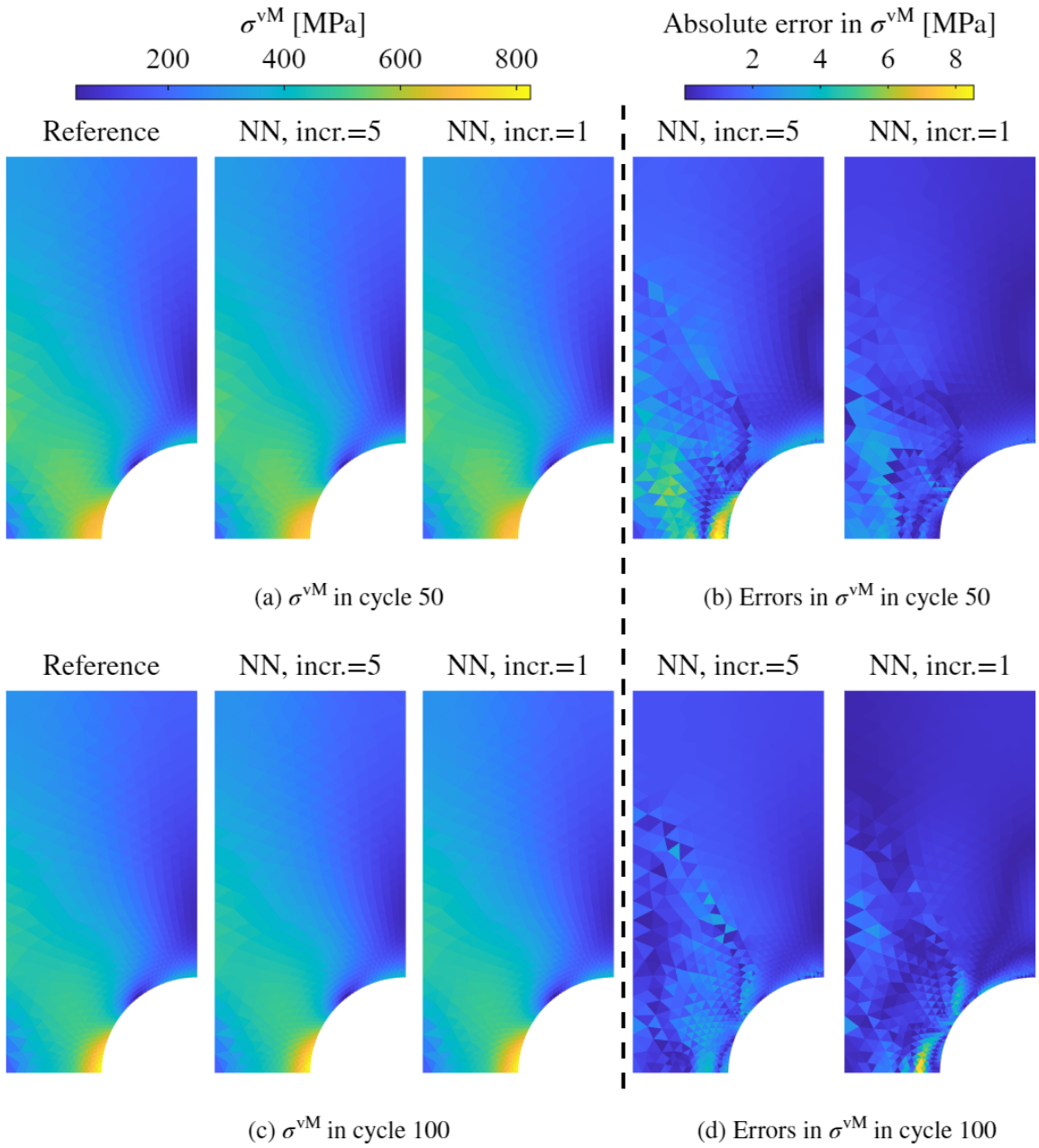


Figure 13: von Mises stress, σ^{vM}, and the corresponding absolute error distributions in cycle 50 and 100, presented at the peak displacement. NN refers to the NN-based FE model with the number of displacement increments per half-cycle indicated by incr.

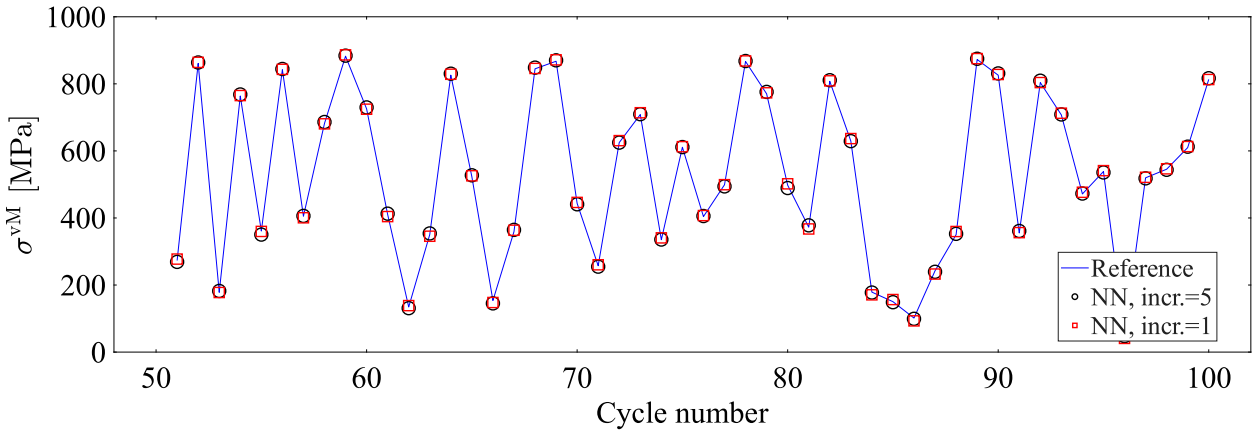
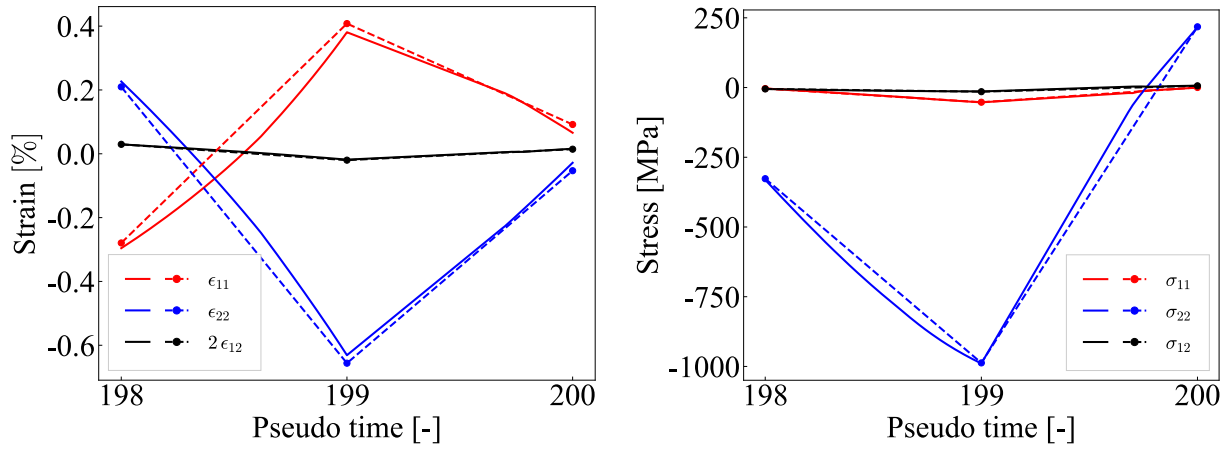


Figure 14: von Mises stresses, σ^{vM}, in the critical element at the peak displacements, predicted by the reference and the NN-based FE models. Results are shown for the last 50 of 100 simulated cycles. incr. denotes the number of increments per half-cycle.



(a) (b)

Figure 15: (a) Strain histories and (b) stress histories for the critical element (indicated in Figure 12) in the last cycle. The solid lines correspond to the reference FE model, and the dashed lines refer to the NN-based FE model with 1 displacement increment per half-cycle.

displacement increments per half-cycle are approximately 14.6 and 34.6 times faster than the reference model, respectively. Further, we have reduced the number of displacement increments for the reference model, using the backward Euler time integration scheme. This has been done until the maximum absolute error in σ^{vM}, evaluated over all cycles and finite elements relative to the highly resolved solution using 100 displacement increments per half-cycle, became comparable to that obtained with the FE models using the NN-based integration. For this purpose, the number of displacement increments for the reference model 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. When compared with the reference model using 8 increments per half-cycle, the NN-based simulations with 5 and 1 increments are roughly 1.4 and 3.3 times faster, 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

- Exact encoding of the separation between elastic and plastic responses.

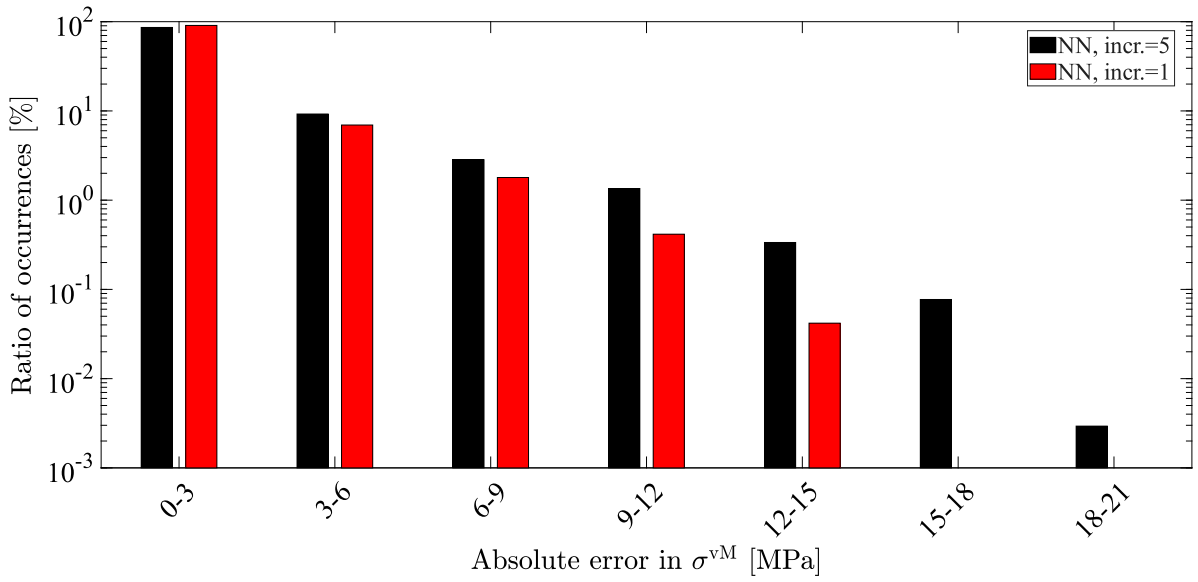


Figure 16: Distribution of absolute errors in von Mises stresses ($\boldsymbol{\sigma}^{\mathrm{vM}}$) at the peak displacements over 100 cycles, considering all finite elements. NN, incr.=5 and NN, incr.=1 refer to the FE models using the NN-based integration scheme with 5 and 1 displacement increments per half-cycle, respectively.

Table 3: CPU time comparison between the reference and NN-based FE models.

Model	Reference FE model	NN-based FE model		
No. displacement increments	100	8	5	1
Averaged CPU time [sec]	672*	63.8	46.0	19.4
Max error in σ^{vM} [MPa]	-	16.1	18.1	14.3

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

- State variable updates formulated by neural networks 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 a prototype material model with the von Mises yield function and nonlinear kinematic hardening. Under pulsating proportional multiaxial loading, we have generated training data for the FFNNs using the prototype model, integrated with the implicit backward Euler time integration algorithm. While only 10 loading cycles were used to generate training data, the proposed NN-based time integration scheme gave very good accuracy for material point simulations. Even when taking a single strain increment per half-cycle for 100 loading cycles, it achieved good predictions. Furthermore, the framework's performance has been evaluated in cyclic finite element (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 is roughly 3 times faster than the reference FE model when taking a single displacement increment per half-cycle.

7 Acknowledgements

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 Algorithmic tangent stiffness

This section presents the derived analytical expression for the Algorithmic Tangent Stiffness (ATS). The ATS tensor is defined as

$$\mathbf{E}^{\mathrm{a}} = \frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\boldsymbol{\epsilon}} = \frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\Delta\boldsymbol{\epsilon}} \tag{31}$$

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

$$\mathbf{E}^{\mathrm{a}} = \mathbf{E}^{\mathrm{e}} - \mathbf{E}^{\mathrm{e}} : \frac{\mathrm{d}\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\mathrm{d}\Delta\boldsymbol{\epsilon}} = \mathbf{E}^{\mathrm{e}} - \mathbf{E}^{\mathrm{e}} : \frac{\partial\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} : \mathbf{E}^{\mathrm{e}} \tag{32}$$

With Equation 24, the differentiation of $\Delta\boldsymbol{\epsilon}^{\mathrm{p}}$ with respect to $\boldsymbol{\sigma}^{\mathrm{tr}}$ leads to

$$\frac{\partial\Delta\boldsymbol{\epsilon}^{\mathrm{p}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} = c\,\frac{\mathrm{d}\mathfrak{g}^{\mathrm{NN}}_{\boldsymbol{\epsilon}^{\mathrm{p}}}}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} + \mathfrak{g}^{\mathrm{NN}}_{\boldsymbol{\epsilon}^{\mathrm{p}}} \otimes \frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} \tag{33}$$

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

$$\begin{aligned}
\frac{\mathrm{d}\mathfrak{g}^{\mathrm{NN}}_{\boldsymbol{\epsilon}^{\mathrm{p}}}}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = \mathtt{SF}_{\boldsymbol{\epsilon}^{\mathrm{p}}} \Bigg[& \mathcal{NN}_{\boldsymbol{\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}_{\boldsymbol{\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 \left[\frac{\partial\mathcal{NN}_{\boldsymbol{\epsilon}^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}} 2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\boldsymbol{\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}_{\boldsymbol{\epsilon}^{\mathrm{p}},1}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\,\beta}} {}^{n}\boldsymbol{\beta}\right] + \\
& \hat{\boldsymbol{\sigma}}^{\mathrm{lim,\,dev}} \otimes \left[\frac{\partial\mathcal{NN}_{\boldsymbol{\epsilon}^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}} 2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\boldsymbol{\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}_{\boldsymbol{\epsilon}^{\mathrm{p}},2}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\,\beta}} {}^{n}\boldsymbol{\beta}\right] + \\
& {}^{n}\hat{\boldsymbol{\beta}} \otimes \left[\frac{\partial\mathcal{NN}_{\boldsymbol{\epsilon}^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr,\,tr}}} 2\,\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} + \frac{\partial\mathcal{NN}_{\boldsymbol{\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}_{\boldsymbol{\epsilon}^{\mathrm{p}},3}\left(\hat{\mathbb{J}}\right)}{\partial J_{\mathrm{tr},\,\beta}} {}^{n}\boldsymbol{\beta}\right] \Bigg]
\end{aligned} \tag{34}$$

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

$$\frac{\partial\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = \frac{\partial}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right]\right] = s\,\mathbf{I} + \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] \otimes \frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} \tag{35}$$

Considering Equation 18, $\mathrm{d}s/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}$ is derived from

$$\frac{\mathrm{d}f_s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = \left.\frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}}\right|_{s} + \left.\frac{\partial f_s}{\partial s}\right|_{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} \frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = 0 \rightarrow \tag{36}$$

$$\frac{\mathrm{d}s}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = -\left[\frac{\partial f_s}{\partial s}\right]^{-1} \frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} \tag{37}$$

where

$$\frac{\partial f_s}{\partial s} = 3\left[\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} - {}^{n}\boldsymbol{\beta}\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right]\right] \tag{38}$$

and

$$\frac{\partial f_s}{\partial\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}} = 3\,s\left[{}^{n}\boldsymbol{\sigma}^{\mathrm{dev}} + s\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\sigma}^{\mathrm{dev}}\right] - {}^{n}\boldsymbol{\beta}\right] \tag{39}$$

Considering Equation 25, $\mathrm{d}c/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$ is derived from

$$\frac{\mathrm{d}f_c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = \left.\frac{\partial f_c}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}}\right|_{c} + \left.\frac{\partial f_c}{\partial c}\right|_{\boldsymbol{\sigma}^{\mathrm{tr}}} \frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = 0 \rightarrow \tag{40}$$

$$\frac{\mathrm{d}c}{\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}} = -\left[\frac{\partial f_c}{\partial c}\right]^{-1} \frac{\partial f_c}{\partial\boldsymbol{\sigma}^{\mathrm{tr}}} \tag{41}$$

where

$$\frac{\partial f_c}{\partial c} = -3\left[\left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta}\right] : \left[2\,G\,\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right] + c\,\left|\left|2\,G\,\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right|\right|^2\right] \tag{42}$$

and

$$\frac{\partial f_c}{\partial \boldsymbol{\sigma}^{\mathrm{tr}}} = 3\left[\mathbf{I}^{\mathrm{dev}} - c\left[2\,G\,\frac{\partial \mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}}}{\partial \boldsymbol{\sigma}^{\mathrm{tr}}} - 2/3 H_{\mathrm{kin}}\,\frac{\partial \mathfrak{g}_{b}^{\mathrm{NN}}}{\partial \boldsymbol{\sigma}^{\mathrm{tr}}}\right]\right] : \left[\boldsymbol{\sigma}^{\mathrm{tr,\,dev}} - {}^{n}\boldsymbol{\beta} - c\left[2\,G\mathfrak{g}_{\epsilon^{\mathrm{p}}}^{\mathrm{NN}} - 2/3\,H_{\mathrm{kin}}\,\mathfrak{g}_{b}^{\mathrm{NN}}\right]\right] \tag{43}$$

$\mathrm{d}\mathfrak{g}_{\mathrm{b}}^{\mathrm{NN}}/\mathrm{d}\boldsymbol{\sigma}^{\mathrm{tr}}$ 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., [19].

B Additional results

This section provides supplementary results for Sections 3.3 and 5.1. Figure 17 shows the computed θ_i values versus the relative number of occurrences considering $\Delta\boldsymbol{b}$, using the proposed method described in Section 3.3.

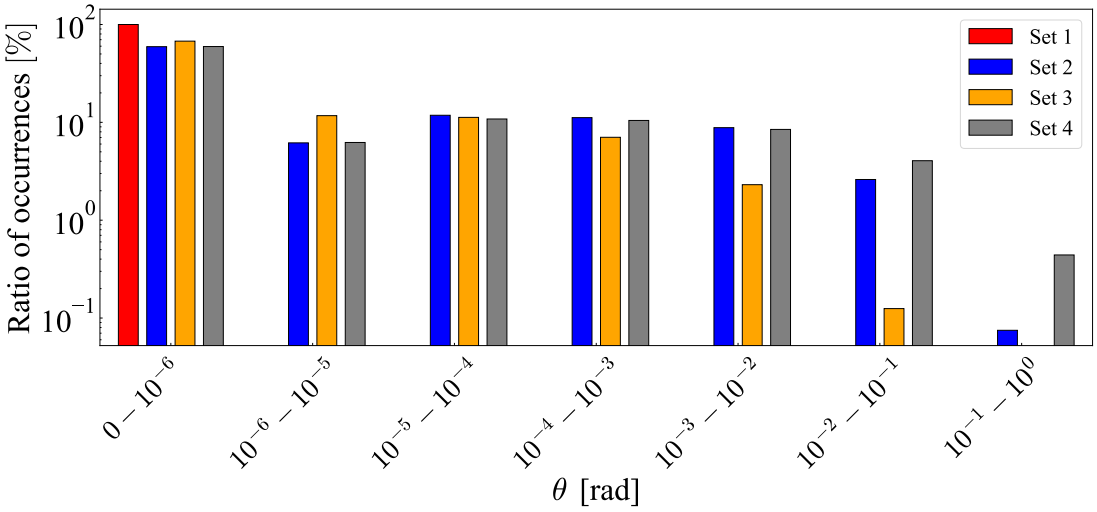
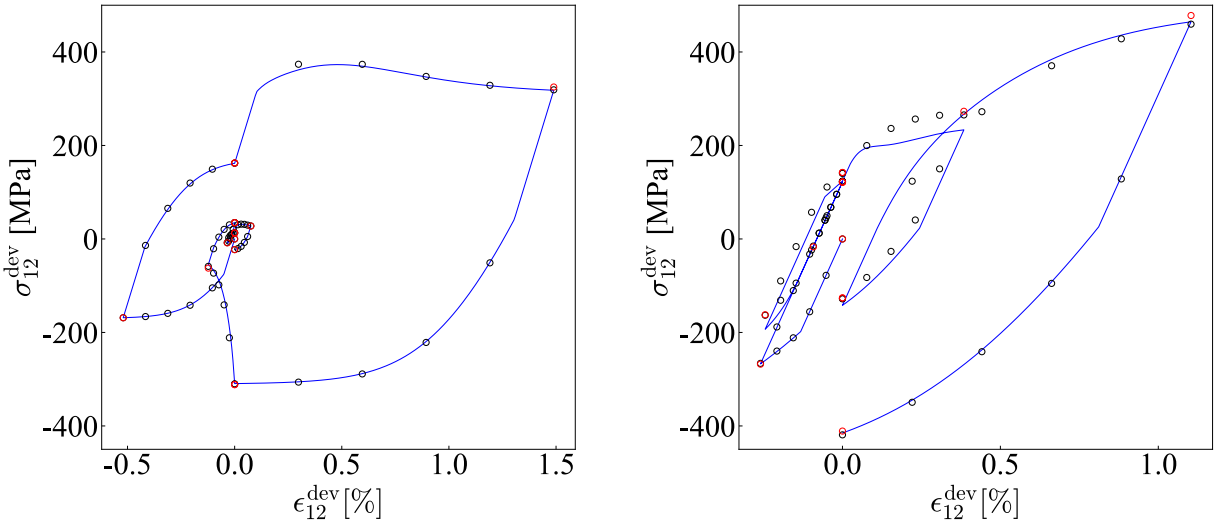


Figure 17: The distribution of the computed angles, θ, between the target and predicted $\Delta\boldsymbol{b}$ considering all training data for four sets of evolution directions: set 1: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$, set 2: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, \boldsymbol{\sigma}^{\mathrm{lim,\,dev}}\right\}$, set 3: $\left\{\boldsymbol{\sigma}^{\mathrm{tr,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$, and set 4: $\left\{\boldsymbol{\sigma}^{\mathrm{lim,\,dev}}, {}^{n}\boldsymbol{\beta}\right\}$.

Figure 18a shows the deviatoric shear responses over the first 5 cycles for the reference and NN-based models, considering the training data. The corresponding responses for the test data are presented in Figure 18b.



(a) Training data

(b) Test data

Figure 18: The shear stress-strain responses from the reference material model (blue) and the NN-based model. The black and red circles correspond to the results from the NN-based model with 5 and 1 strain increments per half-cycle, respectively. Only the first 5 cycles are shown.

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, International Journal of Mechanical Sciences 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, International Journal of 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 Dynamics and Earthquake Engineering 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, International Journal of Plasticity 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, Computer Methods in Applied Mechanics and Engineering 295 (2015) 414–445.

[8] D. Leidermark, K. Simonsson, Procedures for handling computationally heavy cyclic load cases with application to a disc alloy material, Materials at High Temperatures 36 (5) (2019) 447–458.

[9] N. Talebi, K. A. Meyer, M. Ekh, Cycle-domain plasticity modeling using neural networks and symbolic regression, Computers & Structures 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, Archives of Computational Methods in Engineering (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., Archives of computational methods in engineering 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, International Journal for numerical methods in engineering 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, International Journal of Plasticity 120 (2019) 205–219.

[14] D. Huang, J. N. Fuhg, C. Weißenfels, P. Wriggers, A machine learning based plasticity model using proper orthogonal decomposition, Computer Methods in Applied Mechanics and Engineering 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, Computer Methods in Applied Mechanics and Engineering 437 (2025) 117743.

[16] 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, Journal of Computational physics 378 (2019) 686–707.

[17] E. Haghighat, S. Abouali, R. Vaziri, Constitutive model characterization and discovery using physics-informed deep learning, Engineering Applications of Artificial Intelligence 120 (2023) 105828.

[18] F. Foroughi Boroujeni, A. Faramarzi, W. Cha, B. Ghiassi, A physics-informed evolutionary regression framework for constitutive modelling of sand, Computers and Geotechnics 197 (2026) 108179.

[19] P. Weber, W. Wagner, S. Freitag, Physically enhanced training for modeling rate-independent plasticity with feedforward neural networks, Computational Mechanics 72 (4) (2023) 827–857.

[20] P. Weber, J. Geiger, W. Wagner, Constrained neural network training and its application to hyperelastic material modeling, Computational Mechanics 68 (5) (2021) 1179–1204.

[21] K. A. Meyer, F. Ekre, Thermodynamically consistent neural network plasticity modeling and discovery of evolution laws, Journal of the Mechanics and Physics of Solids 180 (2023) 105416.

[22] J. N. Fuhg, C. M. Hamel, K. Johnson, R. Jones, N. Bouklas, Modular machine learning-based elastoplasticity: Generalization in the context of limited data, Computer Methods in Applied Mechanics and Engineering 407 (2023) 115930.

[23] A. Zhang, D. Mohr, Using neural networks to represent von mises plasticity with isotropic hardening, International Journal of Plasticity 132 (2020) 102732.

[24] D. P. Jang, P. Fazily, J. W. Yoon, Machine learning-based constitutive model for j2-plasticity, International Journal of Plasticity 138 (2021) 102919.

[25] P. Fazily, J. W. Yoon, Machine learning-driven stress integration method for anisotropic plasticity in sheet metal forming, International Journal of Plasticity 166 (2023) 103642.

[26] W. G. Dettmer, E. J. Muttio, R. Alhayki, D. Perić, A framework for neural network based constitutive modelling of inelastic materials, Computer Methods in Applied Mechanics and Engineering 420 (2024) 116672.

[27] M. Eghbalian, M. Pouragha, R. Wan, A physics-informed deep neural network for surrogate modeling in classical elasto-plasticity, Computers and Geotechnics 159 (April) (2023).

[28] S. Rezaei, A. Moeineddin, A. Harandi, Learning solutions of thermodynamics-based nonlinear constitutive material models using physics-informed neural networks, Computational Mechanics 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, Computers and Geotechnics 196 (2026) 108170.

[30] P. Zhang, Neural network based numerical integration for elastoplastic constitutive relations, Computers and Geotechnics 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 Elements in Analysis & Design 251 (July) (2025) 104431.

[32] K. Runesson, S. Sture, K. Willam, Integration in computational plasticity, Computers & Structures 30 (1) (1988) 119–130.

[33] H. L. Schreyer, R. F. Kulak, J. M. Kramer, Accurate numerical solutions for elastic-plastic models, Journal of Pressure Vessel Technology 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] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, Journal of Computational Physics 318 (2016) 22–35.

[37] K. Garanger, J. Kraus, J. J. Rimoli, Symmetry-enforcing neural networks with applications to constitutive modeling, Extreme Mechanics Letters 71 (2024) 102188.

[38] J. P. Boehler, On Irreducible Representations for Isotropic Scalar Functions, ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik 57 (6) (1977) 323–327.

[39] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, J. Han, On the variance of the adaptive learning rate and beyond, arXiv:1908.03265 (2019).

[40] A. Paszke, Pytorch: An imperative style, high-performance deep learning library, arXiv:1912.01703 (2019).

[41] F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, Springer, 2019.

[42] M. B. Gorji, M. Mozaffar, J. N. Heidenreich, J. Cao, D. Mohr, On the potential of recurrent neural networks for modeling path dependent plasticity, Journal of the Mechanics and Physics of Solids 143 (2020) 103972.

[43] M. Rosenkranz, K. A. Kalina, J. Brummund, M. Kästner, A comparative study on different neural network architectures to model inelasticity, International Journal for Numerical Methods in Engineering 124 (21) (2023) 4802–4840.

[44] L. Herrmann, S. Kollmannsberger, Deep learning in computational mechanics: a review, Computational Mechanics 74 (2) (2024) 281–331.

[45] S. L. Brunton, J. N. Kutz, Neural Networks and Deep Learning, Cambridge University Press, 2019, p. 195–226.

[46] X. Li, M. Ekh, J. C. Nielsen, Three-dimensional modelling of differential railway track settlement using a cycle domain constitutive model, International Journal for Numerical and Analytical Methods in Geomechanics 40 (12) (2016) 1758–1770.

[47] A. Bernasconi, M. Filippini, S. Foletti, D. Vaudo, Multiaxial fatigue of a railway wheel steel under non-proportional loading, International Journal of Fatigue 28 (5-6) (2006) 663–672.

[48] Dassault Systèmes Simulia Corp., Abaqus Analysis User's Manual, Providence, Rhode Island, USA, 2025th Edition, available from Dassault Systèmes (2025).