Version of Record: https://www.sciencedirect.com/science/article/pii/S0021999120308469
Manuscript_406242331f463a9786b3978b05bdbd44

Learning Constitutive Relations using Symmetric Positive Definite Neural Networks

Kailai Xu[a,*], Daniel Z. Huang[a,*], Eric Darve[a,b]

[a] *Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, 94305*
[b] *Mechanical Engineering, Stanford University, Stanford, CA, 94305*

Abstract

We present a new neural-network architecture, called the Cholesky-factored symmetric positive definite neural network (SPD-NN), for modeling constitutive relations in computational mechanics. Instead of directly predicting the stress of the material, the SPD-NN trains a neural network to predict the Cholesky factor of the tangent stiffness matrix, based on which the stress is calculated in incremental form. As a result of this special structure, SPD-NN weakly imposes convexity on the strain energy function, satisfies the second order work criterion (Hill's criterion) and time consistency for path-dependent materials, and therefore improves numerical stability, especially when the SPD-NN is used in finite element simulations. Depending on the types of available data, we propose two training methods, namely direct training for strain and stress pairs and indirect training for loads and displacement pairs. We demonstrate the effectiveness of SPD-NN on hyperelastic, elasto-plastic, and multiscale fiber-reinforced plate problems from solid mechanics. The generality and robustness of SPD-NN make it a promising tool for a wide range of constitutive modeling applications.

Keywords: Neural Networks, Plasticity, Hyperelasticity, Finite Element Method, Multiscale Homogenization

*Both authors contributed equally to this work.
Email addresses: kailaix@stanford.edu (Kailai Xu), zhengyuh@stanford.edu (Daniel Z. Huang), darve@stanford.edu (Eric Darve)

1. Introduction

One of the goals of material modeling is to construct constitutive models to describe the relationship between strain and stress. The mapping between strain and stress (constitutive relations) can be high dimensional and history-dependent, and these complexities make constitutive modeling very challenging. Traditionally, the constitutive relations are derived from microscopic interactions between multiscale structures or between atoms. However, first-principles simulations (e.g., molecular dynamics based simulations), which can resolve these interactions, remain prohibitively expensive. The computational difficulty motivates the construction of constitutive models with simplified assumptions that still capture the essential physical constraints of constitutive relations, such as isotropicity or thermodynamics principles of material. Parameters in the models are then calibrated on limited and coarse-scale loading test data. These constitutive models lead to affordable and robust simulations, and thus the models are very important for large-scale engineering and scientific applications. However, the functional forms representing these constitutive models are often limited to global exponents and polynomials, which are insufficient to represent complicated multiscale constitutive relations.

Since the development of deep learning techniques, deep neural networks (DNN) have emerged as a promising technique for constitutive modeling. For example, pioneering work has demonstrated the feasibility of constitutive modeling using neural networks in a wide variety of applications, such as Ghaboussi et al. [1] for modeling concrete, Ellis et al. [2] for modeling sands, Shen et al. [3], Liang et al. [4] for modeling hyper-elastic materials, and Furukawa et al. [5] for modeling viscoelastic materials.

Recently, recurrent neural networks (RNNs), which are effective for history-dependent phenomena, have been applied to model multiscale multi-permeability poroplasticity [6], multiscale-plasticity [7], and multiscale one-dimensional bars [8].

Additionally, neural networks have been used to address more complex material behaviors, such as microcracking, brittle fracture, and crack propagation [9, 10, 11, 12]. Neural networks that take as input microscopic structure parameters were used to design of new materials [13, 14, 15, 16].

The cited literature shows that DNNs offer a powerful framework to *represent* complex constitutive relations. There are two important aspects of neural-network-based constitutive relations:

- Firstly, depending on the nature and availability of data, the neural-network-based constitutive relations can either be trained directly or indirectly (Figure 1). Most of the cited literature adopt the **direct** training method, where neural-network-based constitutive models and dynamic structural equations are eventually decoupled: the neural networks, which accept strain (or strain increments, etc.) as input and yield stress as outputs, are trained using input-output pairs. To model complex material behavior, this method requires a voluminous amount of strain-stress data, which can be expensive or impossible to collect. Our proposed approach enables training the neural network with either direct input-output data or **indirect** full-field data [17, 18, 19, 20], such as displacement and external load data, by coupling the neural network with a dynamic structural equation solver (Figure 1); the indirect training approach alleviates the demanding requirement for strain-stress data.

- Secondly and more importantly, even DNNs that appear to make accurate predictions may be numerically unstable when we plug them into a dynamic structural equation time integrator. For example, in benchmarks that we run, accurate DNNs in fact led to unstable models when coupled with dynamic structural equations. For predictive modeling, issues like numerical stability, generalization, and global accuracy have not been sufficiently explored. In the

present work, the numerical stability of the coupled system is studied, and several stability-preserving neural network architectures are proposed.

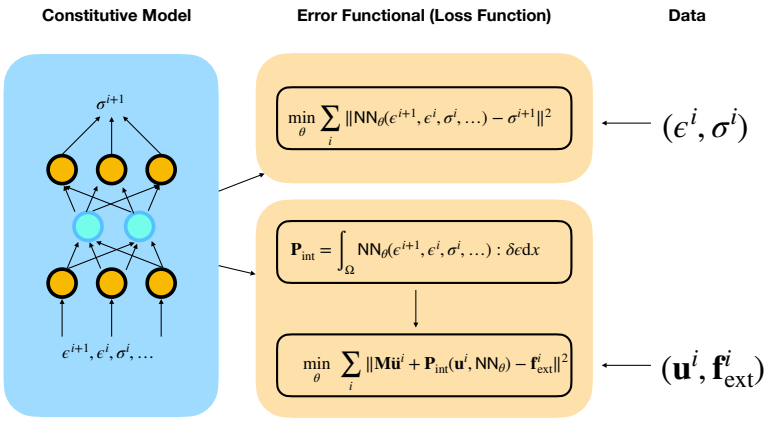


Figure 1: Training neural-network-based constitutive relations NN_θ, based on different types of data. When the strain and stress pairs are available, we can train the neural network using the typical least-square regression. However, when only the displacement and external load (experimental) data are available, we need to couple a neural-network-based constitutive model with the dynamic structural equation. In this work, we demonstrate training neural-network-based constitutive relations with both approaches.

If one solves the structural governing equations embedded with general DNN constitutive models, numerical instabilities are observed (see Section 4.1 for an example). We found that the numerical instability is actually due to the violation of physical constraints in the constitutive models, such as the time consistency and the second order work criterion, which we will discuss in the following. We show that by designing DNNs that satisfy these physical constraints, we can dramatically improve numerical stability and prediction accuracy.

The idea of adding physics-based constraints to the neural network is not new. For example, Ling et al. [21] enforce isotropicity an orthotropy on the predicted strain-energy of the crystal elastic materials through building a basis of invariant inputs. Liu et al. [22, 23] design deep material networks using a composition of simple building blocks inspired from the two-phase linearly elastic model. Heider et al. [24] introduced coordinate-free invariant metrics for the objective frame-independent functions to model anisotropic elastoplastic materials. However, despite better accuracy for predicting the stress as a result of satisfying these constraints, the performance of the resulting structural governing equations coupled with these physics constrained neural networks is not necessarily improved. For example, a trained neural-network-based constitutive model may predict stress relatively accurately, but it may not yield a symmetric positive definite stiffness matrix. Because of this, the numerical solver, where the neural network is embedded, may produce nonphysical solutions or suffer from numerical instability.

In the present work, we focus on the numerical stability aspect of the resulting hybrid model—the conservation equations embedded with NN-based constitutive models. We propose a novel neural network architecture with customized output layers to fulfill two objectives.

- The first objective is related to the convexity of the strain-energy, which plays a significant role in the numerical stability. To this end, we propose to predict the tangent stiffness matrix

and enforce that it be symmetric positive definite.[1] The symmetric positive definiteness guarantees the weak convexity of the strain-energy and fulfills the the second order work criterion (Hill's criterion) for path-dependent materials

- Another objective is ensuring time consistency Equation (13), in particular for path-dependent materials, such as elasto-plastic materials.

To achieve these objectives, instead of training a neural network to identify a nonlinear map directly between strain and stress, we train a neural network (with weights and biases $\boldsymbol{\theta}$) that maps the strain (and possibly other relevant quantities) to a **lower triangular matrix** $\mathsf{L}_{\boldsymbol{\theta}}$ (a Cholesky factor). We then construct the constitutive model in the following incremental form:

$$\Delta\boldsymbol{\sigma} = \mathsf{L}_{\boldsymbol{\theta}}\mathsf{L}_{\boldsymbol{\theta}}^{T}\Delta\boldsymbol{\epsilon} \tag{1}$$

where $\Delta\boldsymbol{\epsilon}$ and $\Delta\boldsymbol{\sigma}$ are the incremental strain and stress in Voigt notation.[2] In our formulation, the tangent stiffness matrix $\mathsf{L}_{\boldsymbol{\theta}}\mathsf{L}_{\boldsymbol{\theta}}^{T}$ is automatically symmetric positive semidefinite. If we assume that $\mathsf{L}_{\boldsymbol{\theta}}$ is bounded, i.e., $\|\mathsf{L}_{\boldsymbol{\theta}}\|_2 \leq C < \infty$ for a constant independent of $\boldsymbol{\theta}$, this incremental form leads to a time consistent scheme.

In the present work, results based on neural networks trained on both direct input-output data and indirect full-field data [17, 18, 19] are presented. The robustness of our approach is demonstrated in different numerical applications, including a hyperelastic material, an elasto-plastic material, and a multiscale material. We have developed a software library that seamlessly combines traditional finite element methods and neural networks. The code is accessible online:

`https://github.com/kailaix/NNFEM.jl`

The remainder of this paper is organized as follows. We first introduce the background in Section 2, including the governing equations, different classical constitutive relations and their associated constraints. In Section 3, we present our constraint-embedded neural network architecture—SPD-NN, and the training procedures. Finally, we apply the learned NN-based constitutive relations to several solid mechanics problems, including a one-dimensional truss coupon, and several two-dimensional thin plate problems with hyperelastic material, elasto-plastic material and fiber-reinforced multiscale material. Several issues related to neural networks are discussed in Section 5. We discuss a possible generalization of the approach in Section 6.

2. Background

2.1. Governing Equations

The governing equation of a solid undergoing infinitesimal deformations can be written as

$$\begin{aligned} \rho\ddot{\mathbf{u}} &= \mathtt{div}\ \boldsymbol{\sigma} + \rho\mathbf{b} && \text{in } \Omega \\ \mathbf{u} &= \bar{\mathbf{u}} && \text{on } \Gamma_u \\ \boldsymbol{\sigma}\boldsymbol{n} &= \bar{\boldsymbol{t}} && \text{on } \Gamma_t \end{aligned} \tag{2}$$

where ρ is the mass density, u is the displacement vector, $\boldsymbol{\sigma}$ is the stress tensor, and $\rho\mathbf{b}$ is the body force vector; Ω denotes the computational domain. The prescribed displacement $\bar{u}$ and the surface

[1] Execptions exist, i.e., non-associate elasto-plasticity, where both symmetry and definite positiveness are lost, and bifurcations and possible instabilities appear inside the plastic limit surface.

[2] For brevity, $\boldsymbol{\epsilon}$ and $\boldsymbol{\sigma}$ represent both tensor form and vector form in Voigt notation depending on the context.

traction $\bar{\boldsymbol{t}}$ are imposed on the domain boundaries Γ_u and Γ_t with the outward unit normal $\boldsymbol{n}$, where $\Gamma_u \cap \Gamma_t = \emptyset$ and $\Gamma_u \cup \Gamma_t = \partial\Omega$.

To solve for the displacement u from Equation (2), we also need the constitutive relations, which maps the deformation history of the structure to the stress:

$$\boldsymbol{\sigma}(t) = \mathcal{M}(\boldsymbol{\epsilon}(t), \mathcal{I}(t)) \tag{3}$$

Here $\boldsymbol{\epsilon}(t)$ is the strain tensor at time t related to the displacement vector and $\mathcal{I}(t)$ denotes all other quantities related to material states during time $\tau = [0, t)$, such as $\boldsymbol{\epsilon}(\tau)$, $\boldsymbol{\sigma}(\tau)$, etc. The infinitesimal strain tensor is

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]$$

When we apply the finite element method to solve Equation (2) numerically, we have the following semi-discrete equation at time t

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{P}(\mathbf{u}, \mathcal{M}(\boldsymbol{\epsilon}(\mathbf{u}), \mathcal{I})) = \mathbf{f}(\mathbf{u}, x, p) \tag{4}$$

where we use the same notation $\mathbf{u}$ to denote the spatial discretization of the displacement vector $\mathbf{u}$ in Equation (2), $\mathbf{M}$ is the discrete mass matrix, $\mathbf{P}$ and $\mathbf{f}$ are the discrete internal and external force vectors, x is the coordinate vector, and p is the parameter vector of external loads. We adopt the generalized α-method [25] with $\alpha_m = -1$ and $\alpha_f = 0$ for temporal discretization of Equation (4). This generalized α-method allows for dissipating high frequency energy to damp high frequency modes, which is crucial for the robustness of the numerical solver when an approximate constitutive relation is used.

Remark 1. *For structure undergoing finite or large deformations, the finite strain tensor reads*

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}[\nabla\mathbf{u} + (\nabla\mathbf{u})^T + (\nabla\mathbf{u})^T\nabla\mathbf{u}] \tag{5}$$

$\boldsymbol{\sigma}$ in Equation (2) represents the first Piola-Kirchhoff stress tensor, the constitutive relation generally relates the finite strain tensor with the symmetric second Piola-Kirchhoff stress tensor

$$\boldsymbol{S}(t) = \mathcal{M}(\boldsymbol{\epsilon}(t), \mathcal{I}(t)) \tag{6}$$

The first Piola-Kirchhoff tensor and the second Piola-Kirchhoff tensor are related by

$$\boldsymbol{\sigma} = \boldsymbol{F}\boldsymbol{S}$$

here $\boldsymbol{F} = \nabla\mathbf{u} + \mathbb{I}$ is the deformation gradient tensor, where $\mathbb{I}$ is the identity matrix.

Our method works for both infinitesimal and finite deformations. Numerical examples of structures undergoing finite deformations are reported in Section 4.1 and Section 4.2.1, and examples using infinitesimal deformations are reported in other numerical examples.

2.2. Constitutive Relations and Associated Constraints

Equation (3) represents one possible form of constitutive relations. Even when restricted to this specific form, the constitutive relations can describe a great variety of material properties. Nevertheless, most of the constitutive relations share extra constraints (i.e., objectivity, convex strain energy function, and second law of thermodynamics) that play significant roles in material stability and numerical stability. In what follows, several stability- and consistency-related constraints of constitutive relations are discussed. These constraints should be considered in any data-driven constitutive models.

2.2.1. Hyperelastic Materials

The constitutive relations of hyperelastic materials are path-independent and are related to the strain-energy density function $\omega(\boldsymbol{\epsilon})$,

$$\boldsymbol{\sigma} = \frac{\partial \omega(\boldsymbol{\epsilon})}{\partial \boldsymbol{\epsilon}} \tag{7}$$

In general, the strain-energy density function is assumed to be convex, namely, the tangent stiffness matrix $\frac{\partial^2 \omega(\boldsymbol{\epsilon})}{\partial^2 \boldsymbol{\epsilon}}$ is symmetric positive definite (SPD). The assumption in one-dimensional is equivalent to that the strain-stress is monotonically increasing[3]. The convexity is crucial to both the stability of the material and also the numerical scheme.

2.2.2. Elasto-plastic Materials

The constitutive relations of elasto-plastic materials are rate-independent but path-dependent, and feature transition between elastic and plastic behaviors. Namely, under loading, the material behaves elastically until the initial yield stress σ_Y is attained, and then undergoes permanent irreversible plastic deformations with further loading. The onset and continuance of plastic deformation is governed by a yield function

$$f(\boldsymbol{\sigma}) \leq 0 \tag{8}$$

For plastic deformation, the stress state must remain on the yield surface $f = 0$, thanks to hardening parameters; and for elastic deformation, the yield function satisfies $f < 0$.

The strain is assumed to be additively decomposed into elastic $\boldsymbol{\epsilon}^e$ and plastic $\boldsymbol{\epsilon}^p$ parts, as follows,

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^e + \boldsymbol{\epsilon}^p \tag{9}$$

The constitutive relation relates the stress and the elastic part of the strain:

$$\boldsymbol{\sigma} = \mathsf{C}\boldsymbol{\epsilon}^e \tag{10}$$

here C is the tangent stiffness tensor. The plastic strain rate is given by a flow rule, i.e., the associative flow rule as follows,

$$\dot{\boldsymbol{\epsilon}}^p = \dot{\lambda}\frac{\partial f}{\partial \boldsymbol{\sigma}}$$

where $\dot{\lambda}$ is called the plastic rate parameter or the consistency parameter, which is non-zero only if $f = 0$.

Naturally, the constitutive relation is written in the rate form

$$\dot{\boldsymbol{\sigma}} = \mathsf{H}\dot{\boldsymbol{\epsilon}} = \begin{cases} \mathsf{C}\dot{\boldsymbol{\epsilon}} & \text{if } \ f < 0 \\ \left[\mathsf{C} - \dfrac{(\mathsf{C}\frac{\partial f}{\partial \boldsymbol{\sigma}})(\mathsf{C}\frac{\partial f}{\partial \boldsymbol{\sigma}})^T}{\left(\frac{\partial f}{\partial \boldsymbol{\sigma}}\right)^T \mathsf{C}\frac{\partial f}{\partial \boldsymbol{\sigma}}}\right]\dot{\boldsymbol{\epsilon}} & \text{if } \ f = 0 \end{cases} \tag{11}$$

It is worth mentioning the tangent stiffness matrix H is symmetric positive semidefinite, and when strain hardening is considered, it becomes SPD. The definite positiveness leads to the second order work criterion, proposed by Hill [27]:

$$\delta^2 W := \frac{1}{2}\dot{\boldsymbol{\sigma}}^T \cdot \dot{\boldsymbol{\epsilon}} \geq 0 \tag{12}$$

[3]Exceptions exist, i.e., strain-softening [26], which is beyond the scope of the present work.

which is also a sufficient condition for uniqueness of the elasto-plastic boundary value problem (see [28, 29]). Moreover, the form in Equation (11) implies time consistency, i.e., as $\Delta\boldsymbol{\epsilon} \to 0$, $\Delta\boldsymbol{\sigma} \to 0$, and also rate-independent[4], i.e., the tangent stiffness matrix is independent of the strain or stress rate.

2.2.3. Associated Constraints

Based on the discussion above, the following properties are crucial to be incorporated in data-driven constitutive models

(1) Symmetry positive definiteness of the tangent stiffness matrix (i.e., strain energy convexity), which leads to non-singular stiffness matrix. We also found that the SPD property is crucial for numerical stability.
(2) Time consistency, which is formulated as follows,

$$\lim_{\Delta\boldsymbol{\epsilon}\to 0} \Delta\boldsymbol{\sigma} = 0 \tag{13}$$

It is crucial for the convergence of the numerical approximation when $\Delta t \to 0$.

3. Methodology

In this section, we describe our method for learning the constitutive relations Equations (3) and (6). Our discussion will be divided into two parts:

1. the neural network architecture for approximating the mapping between the strain and the stress;
2. the direct and indirect training methods based on the types of available data.

3.1. Neural Network Architectures

We note that the neural network has already been used to approximate the constitutive relations. The neural network architectures in many literatures output stress directly ($\boldsymbol{\sigma}$-NN) with the following form (recall that $\mathcal{I}$ stands for other relevant information up to the current time)

$$\boldsymbol{\sigma}\text{-NN:} \qquad \boldsymbol{\sigma} = \mathsf{NN}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathcal{I}) \tag{14}$$

or output stress increment directly ($\Delta\boldsymbol{\sigma}$-NN):

$$\Delta\boldsymbol{\sigma}\text{-NN:} \qquad \Delta\boldsymbol{\sigma} = \mathsf{NN}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathcal{I}) \tag{15}$$

Note that $(\boldsymbol{\epsilon}, \mathcal{I})$ is an abstract description of the neural network dependencies, which may include the strain rate information $\dot{\boldsymbol{\epsilon}}$, or directional information $\dfrac{\Delta\boldsymbol{\epsilon}}{\|\Delta\boldsymbol{\epsilon}\|}$.

These architectures are suitable for learning the constitutive relations when the strain and the stress data are available. However, these strain-stress relations expressed by Equations (14) and (15) do not satisfy certain physical constraints and thus may break numerical solvers (see Section 4.1), when we plug them into a numerical solver.

[4]Strictly speaking, rate independency implies homogeneity, namely the stiffness tensor must be homogeneous of degree 0 and be dependent only on the direction of strain rate or stress rate.

We propose an alternative architecture (SPD-NN) based on the incremental form,

$$\begin{aligned} \Delta\boldsymbol{\sigma} &= \mathsf{H}_{\boldsymbol{\theta}}\Delta\boldsymbol{\epsilon} = \mathsf{L}_{\boldsymbol{\theta}}\mathsf{L}_{\boldsymbol{\theta}}^{T}\Delta\boldsymbol{\epsilon} \\ \mathsf{L}_{\boldsymbol{\theta}} &= \mathsf{NN}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}, \mathcal{I}) \end{aligned} \tag{16}$$

Here instead of outputting the stress or stress increment directly, the neural network outputs a lower triangular matrix $\mathsf{L}_{\boldsymbol{\theta}}$, which is the Cholesky factor of the tangent stiffness matrix $\mathsf{H}_{\boldsymbol{\theta}}$. The numerical approximation to Equation (16) in the dynamic simulations has the following form

$$\boldsymbol{\sigma}^{n+1} = \mathsf{L}_{\boldsymbol{\theta}}\mathsf{L}_{\boldsymbol{\theta}}^{T}(\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^{n}) + \boldsymbol{\sigma}^{n} \tag{17}$$

here, the superscript n indicates the time step.

An obvious advantage of Equation (16) is the guarantee that the tangent stiffness matrix is a symmetric semidefinite matrix, which is true for commonly used constitutive relations (i.e., hyper-elasticity and associate elasto-plasticity). Additionally, when $\mathsf{NN}_{\boldsymbol{\theta}}$ is bounded, we have

$$\lim_{\Delta\boldsymbol{\epsilon}\to 0} \Delta\boldsymbol{\sigma} = 0 \tag{18}$$

which proves the time-consistency for both path-dependent/independent constitutive relations. In contrast, both the $\boldsymbol{\sigma}$-NN with $\boldsymbol{\sigma}^{n+1} = \mathsf{NN}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1}, \boldsymbol{\epsilon}^{n}, \boldsymbol{\sigma}^{n})$ and the $\Delta\boldsymbol{\sigma}$-NN with

$$\boldsymbol{\sigma}^{n+1} = \mathsf{NN}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1}, \boldsymbol{\epsilon}^{n}, \boldsymbol{\sigma}^{n}) + \boldsymbol{\sigma}^{n}$$

fail to satisfy the time-consistency condition Equation (18) for path-dependent constitutive relations.

In the following, we discuss how to adapt SPD-NN for different materials.

- **Linear Elasticity**

  For linear elastic material, the tangent stiffness matrix is independent of the strain or stress. Therefore, no neural network is needed, and the constitutive relation reads

  $$\boldsymbol{\sigma}^{n+1} = \mathsf{C}_{\theta}\boldsymbol{\epsilon}^{n+1} \tag{19}$$

  where C_{θ} is the parametric tangent stiffness tensor and the unknowns are simply entries in the tensor (no neural network).

- **Nonlinear Elasticity**

  For nonlinear elastic material, the tangent stiffness matrix Equation (7) depends only on the strain at the current time step. The constitutive relation Equation (17) can be formulated as

  $$\boldsymbol{\sigma}^{n+1} = \mathsf{L}_{\theta}(\boldsymbol{\epsilon}^{n+1})\mathsf{L}_{\theta}(\boldsymbol{\epsilon}^{n+1})^{T}(\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^{n}) + \boldsymbol{\sigma}^{n} \tag{20}$$

  Note the input $\boldsymbol{\epsilon}^{n+1}$ must be evaluated at the current time step $n+1$ because it is impossible to determine the stress at time step $n+1$ given only the information at time step n.

- **Elasto-Plasticity**

  For elasto-plastic material, the material behavior features transition from elastic behavior to plastic behavior. To model this effect, we consider two types of constitutive relations:

  1. In the linear elasticity region, the constitutive relation is approximated by

     $$\boldsymbol{\sigma}^{n+1}_{\text{elasticity}} = \mathsf{C}_{\theta}(\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^{n}) + \boldsymbol{\sigma}^{n} \tag{21}$$

2. In the plasticity region, the constitutive relation is approximated by

$$\boldsymbol{\sigma}^{n+1}_{\text{plasticity}} = \mathsf{L}_\theta(\boldsymbol{\epsilon}^{n+1}, \boldsymbol{\epsilon}^n, \boldsymbol{\sigma}^n)\mathsf{L}_\theta(\boldsymbol{\epsilon}^{n+1}, \boldsymbol{\epsilon}^n, \boldsymbol{\sigma}^n)^T(\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^n) + \boldsymbol{\sigma}^n \tag{22}$$

However, since we do not know when the transition occurs (the yield strength σ_Y is not available and strain hardening could strength the material), we can relax the constitutive relation using a differentiable[5] transition function $D(\boldsymbol{\sigma}^n, \tilde{\sigma}_Y)$ (see Figure 2), whose value is between 0 and 1, as follows,

$$\boldsymbol{\sigma}^{n+1} = (1 - D(\boldsymbol{\sigma}^n, \tilde{\sigma}_Y))\boldsymbol{\sigma}^{n+1}_{\text{elasticity}} + D(\boldsymbol{\sigma}^n, \tilde{\sigma}_Y)\boldsymbol{\sigma}^{n+1}_{\text{plasticity}} \tag{23}$$

here $\tilde{\sigma}_Y$ is the estimated yield strength, which does not need to be accurate. When the equivalent stress estimated from $\boldsymbol{\sigma}^n$ (such as von Mises stress)[6] is smaller than $\tilde{\sigma}_Y$, the material behavior is assumed to be linear and described by Equation (21); otherwise, it is assumed to be described by the plastic form Equation (22). It is worth noting the plastic form Equation (22) can degenerate to the linear elastic form Equation (21), but not vice versa. Therefore, the estimated yield strength $\tilde{\sigma}_Y$ should be smaller than the yield strength σ_Y. In the present study, historical data for plastic form Equation (22) span only one time step, more historical data might be needed for materials with strong hysteresis.

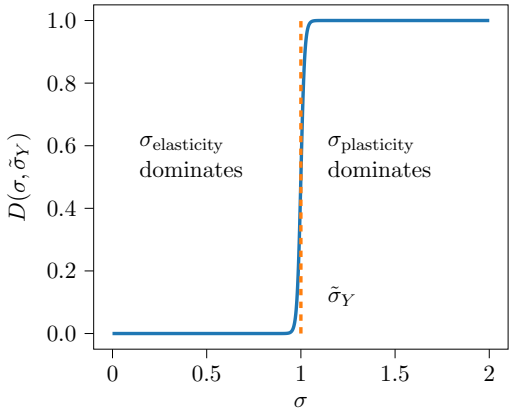


Figure 2: An exemplary transition function $D(\sigma, \tilde{\sigma}_Y) = \texttt{sigmoid}\left(100(\sigma^2 - \tilde{\sigma}_Y^2)\right)$ and $\tilde{\sigma}_Y = 1$. Here $\texttt{sigmoid}$ is the sigmoid function $\texttt{sigmoid}(x) = (1 + e^{-x})^{-1}$.

Remark 2. *Strictly speaking, the stiffness tensor $\mathsf{L}_{\boldsymbol{\theta}}\mathsf{L}_{\boldsymbol{\theta}}^T$ must be homogeneous of degree 0 and be dependent only on the direction of strain rate or stress rate:*

$$\frac{\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^n}{\|\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^n\|}, \quad \frac{\boldsymbol{\sigma}^{n+1} - \boldsymbol{\sigma}^n}{\|\boldsymbol{\sigma}^{n+1} - \boldsymbol{\sigma}^n\|}$$

In the implementation, we use the formulation Equation (22) *for simplicity.*

[5]The differentiability of the transition function is necessary since we need to evaluate the gradient of $\boldsymbol{\sigma}^{n+1}$ with respect to $\boldsymbol{\sigma}^n$ during the training with indirect data.

[6]In general, D should depend on $\boldsymbol{\sigma}^{n+1}$ instead of $\boldsymbol{\sigma}^n$. Because it is difficult to express $\boldsymbol{\sigma}^{n+1}$ explicitly using this form, we assume $\boldsymbol{\sigma}^{n+1} \approx \boldsymbol{\sigma}^n$ and thus obtain Equation (23).

Remark 3. *Generally, we can write the strain-stress relations for a linear elastic material in Voigt notation, as follows:*

$$\begin{bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{bmatrix} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1123} & C_{1113} & C_{1112} \\ & C_{2222} & C_{2233} & C_{2223} & C_{2213} & C_{2212} \\ & & C_{3333} & C_{3323} & C_{3313} & C_{3312} \\ & & & C_{2323} & C_{2313} & C_{2312} \\ & symm & & & C_{1313} & C_{1312} \\ & & & & & C_{1212} \end{bmatrix} \begin{bmatrix}\epsilon_{11}\\ \epsilon_{22}\\ \epsilon_{33}\\ 2\epsilon_{23}\\ 2\epsilon_{13}\\ 2\epsilon_{12}\end{bmatrix}$$

The corresponding Cholesky factor $\mathsf{L}_{\boldsymbol{\theta}}$ *of* $\mathsf{C}_{\boldsymbol{\theta}}$ *is a full lower triangular matrix. For the materials considered in the present work, they are orthotropic, i.e., they have three mutually orthogonal planes of reflection symmetry. Thus the tangent stiffness matrix can be expressed by a block diagonal matrix (Orth-NN), when orthotropy axes are used to form an orthogonal frame:*

$$\mathsf{C}_{\boldsymbol{\theta}} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & & & \\ & C_{2222} & C_{2233} & & & \\ & & C_{3333} & & & \\ & & & C_{2323} & & \\ & symm & & & C_{1313} & \\ & & & & & C_{1212} \end{bmatrix}$$

Therefore, the associated Cholesky factor $\mathsf{L}_{\boldsymbol{\theta}}$ *has the following form*

$$\mathsf{L}_{\boldsymbol{\theta}} = \begin{bmatrix} L_{1111} & & & & & \\ L_{2211} & L_{2222} & & & & \\ L_{3311} & L_{3322} & L_{3333} & & & \\ & & & L_{2323} & & \\ & & & & L_{1313} & \\ & & & & & L_{1212} \end{bmatrix} \tag{24}$$

Consequently, we can further simplify the neural network outputs to only the nonzero entries in Equation (24)*. Equation* (24) *is the form of the Cholesky factor used in the present work.*

Remark 4. *Bringing the rate form of Equation* (16) *into Equation* (12) *leads to*

$$\frac{1}{2}\dot{\boldsymbol{\sigma}}^T \cdot \dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{\epsilon}}^T \mathsf{L}_{\boldsymbol{\theta}} \mathsf{L}_{\boldsymbol{\theta}}^T \dot{\boldsymbol{\epsilon}} \geq 0$$

Therefore, our proposed SPD-NN architecture Equation (16) *is fully consistent with the second order work criterion.*

3.2. Training Methods

Based on the types of available data, i.e., direct and indirect, we can employ different training methods. For the following discussion, we summarize the neural network based constitutive relations as follows

$$\boldsymbol{\sigma}^{n+1} = \mathsf{M}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1}, \boldsymbol{\epsilon}^n, \boldsymbol{\sigma}^n) := \begin{cases} \mathsf{C}_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^{n+1} & \text{Linear Elasticity} \\ \mathsf{L}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1}) \mathsf{L}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1})^T (\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^n) + \boldsymbol{\sigma}^n & \text{Nonlinear Elasticity} \\ (1 - D(\boldsymbol{\sigma}^n, \tilde{\sigma}_Y)) \boldsymbol{\sigma}^{n+1}_{\text{elasticity}} + D(\boldsymbol{\sigma}^n, \tilde{\sigma}_Y) \boldsymbol{\sigma}^{n+1}_{\text{plasticity}} & \text{Elasto-Plasticity} \end{cases}$$

3.2.1. Direct Data

Direct data consist of the input and the output of the NN-based constitutive relations such as strain-stress pairs or strain-stress increments pairs. These data points come from experimental measurements and numerical simulation results. The comprehensive strain-stress data measurement relying on simple mechanical tests, such as tensile or bending tests, might be challenging. However, comprehensive strain-stress data generated from sub-scale simulations, such as representative volume element (RVE) simulations [30, 31, 32, 33, 34] or post-processed from direct numerical simulations, are widely used to train neural networks [13, 14, 21, 6].

Mathematically, the direct data are given in terms of N sequences of strain-stress pairs at n time snapshots.

$$(\boldsymbol{\epsilon}_j^1, \boldsymbol{\sigma}_j^1),\ (\boldsymbol{\epsilon}_j^2, \boldsymbol{\sigma}_j^2),\ \cdots,\ (\boldsymbol{\epsilon}_j^n, \boldsymbol{\sigma}_j^n) \quad j = 1, 2, 3, \ldots, N$$

Here the superscripts indicate time and the subscripts indicate the sequential number. We train the neural network by solving a minimization problem

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := \sum_{j=1}^{N} \sum_{i=2}^{n} \left(\boldsymbol{\sigma}_j^i - \mathsf{M}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}_j^i, \boldsymbol{\epsilon}_j^{i-1}, \boldsymbol{\sigma}_j^{i-1}) \right)^2 \tag{25}$$

3.2.2. Indirect Data

Indirect data consist of deformation data from structure coupons under different load conditions. Deformations are measured by techniques such as digital image correlation or grid method [35]. These techniques can record complete heterogeneous fields, which are rich in the constitutive relations. The virtual fields method [17, 36, 37, 38] has been designed to apply the finite element method (FEM) to bridge the full-field data with parametric constitutive relations. Recently, the method is generalized to an end-to-end training procedure to learn neural-network (or its counterparts) based constitutive relations from the full-field data [18].

The indirect data are given by N full-field deformation-load sequential data at n time snapshots

$$(\mathbf{u}_j^1, \mathbf{f}_j^1),\ (\mathbf{u}_j^2, \mathbf{f}_j^2),\ \cdots,\ (\mathbf{u}_j^n, \mathbf{f}_j^n) \quad j = 1, 2, 3, \ldots, N$$

Here the superscripts indicate time and the subscripts indicate the sequential number.

We can compute the acceleration and the stress using the formulas

$$\ddot{\mathbf{u}}_j^i := \frac{\mathbf{u}_j^{i+1} - 2\mathbf{u}_j^i + \mathbf{u}_j^{i-1}}{\Delta t^2} \tag{26}$$

$$\boldsymbol{\sigma}_j^i(\boldsymbol{\theta}) := \mathsf{M}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}(\mathbf{u}_j^i), \boldsymbol{\epsilon}(\mathbf{u}_j^{i-1}), \boldsymbol{\sigma}_j^{i-1}(\boldsymbol{\theta})), \quad i = 2, \ldots, n-1 \tag{27}$$

$$\ddot{\mathbf{u}}_j^1 = 0,\ \mathbf{u}_j^1 = 0,\ \boldsymbol{\sigma}_j^1(\theta) = 0 \tag{28}$$

Here Δt is the time step, which is assumed to be constant. We train the neural network by solving a minimization problem

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) := \sum_{j=1}^{N} \sum_{i=2}^{n-1} \left(\mathbf{M}\ddot{\mathbf{u}}_j^i + \mathbf{P}(\mathbf{u}_j^i, \boldsymbol{\sigma}_j^i(\theta)) - \mathbf{f}_j^i \right)^2 \tag{29}$$

Here i is the index for time and j is the index for spacial degrees of freedom. It is worth mentioning, in Equation (29), the predicted stress $\boldsymbol{\sigma}_j^i(\boldsymbol{\theta}) = \mathsf{M}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}(\mathbf{u}_j^i), \boldsymbol{\epsilon}(\mathbf{u}_j^{i-1}), \boldsymbol{\sigma}_j^{i-1}(\boldsymbol{\theta}))$ depends on the *predicted* stress $\boldsymbol{\sigma}_j^{i-1}(\boldsymbol{\theta})$ from the last time step. The procedure of evaluating the residual $\mathcal{L}^i = \left(\mathbf{M}\ddot{\mathbf{u}}_j^i + \mathbf{P}(\mathbf{u}_j^i, \boldsymbol{\sigma}_j^i(\theta)) - \mathbf{f}_j^i \right)^2$ at each time step in Equation (29) is depicted in Figure 3. The

procedure resembles the recurrent neural network in deep learning, where the state variables are the stresses. Like the recurrent neural network, the stress predictions at different time steps must be sequentially computed and so does the back-propagated gradients, and thus the computation is hard to be parallelized. Additionally, the training of recurrent neural network suffers from exploding and vanishing gradients problem [39], which poses a challenge for indirect data training of SPD-NNs as well. For example, when we train recurrent neural networks using gradient back-propagation, we need to back-propagate the gradients layer by layer. When the `tanh` activation function is used, its derivative is close to zero when the input to `tanh` is large. Therefore, the back-propagated gradients will possibly shrink layer after layer. The small gradient makes the training process challenging. In other cases, when the RELU activation function is used, its derivative is not contracted; therefore, back-propagation gradients can accumulate exponentially. This phenomenon is called gradient explosion. There are many efforts for alleviating these problems. One notable technique is to simply back-propagate the gradients for only a few layers instead of all of them. This technique results in an inaccurate gradient and thus its effectiveness may be case dependent. Another approach is to carefully initialize the neural network. We employ the latter approach in our present work.

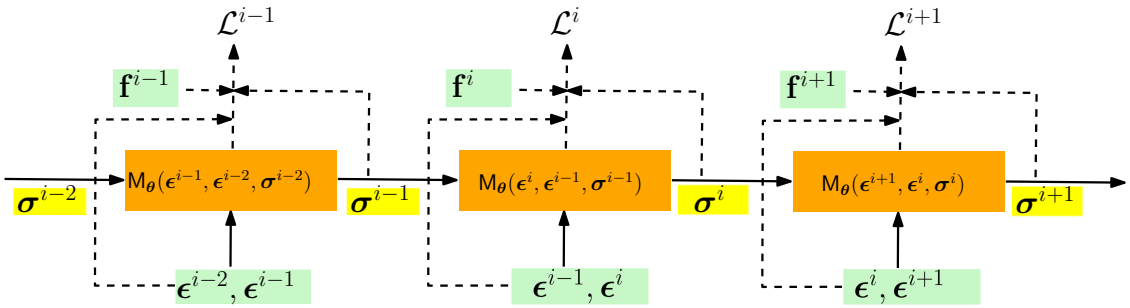


Figure 3: The procedure of evaluating the residual $\mathcal{L}^i$ at each time step for training with indirect data approach in Equation (29).

3.3. Initialization for Indirect Data Training

Another notable challenge in training the recurrent neural network based constitutive relation is the existence of many local minima, since the training data set is small in the present work, and different constitutive relations under the same external loads may produce similar displacements. A common strategy to find a good local minimum is to start from multiple random weights and biases. However, this strategy is expensive and does not guarantee a good initial guess. We thereby propose an initialization technique that yields a set of reasonably good initial weights and biases.

The key idea is that according to Equation (4), we have the relation between the internal force and the stress field

$$\mathbf{P}(\mathbf{u}_j^i, \boldsymbol{\sigma}_j^i) = \mathbf{f}_j^i - \mathbf{M}\ddot{\mathbf{u}}_j^i \tag{30}$$

However, solving the stress field $\boldsymbol{\sigma}_j^i$ from Equation (30) is generally an underdetermined problem, since the number of equations is fewer than the number of stress unknowns. To alleviate this, quadratic elements are used to represent the displacement field $\mathbf{u}$, and in each quadratic element, a linear stress field is assumed and approximated (see Figure 4).

The number of stress unknowns is roughly $3(n_x+1)(n_y+1)$ for a 2D plate, and $6(n_x+1)(n_y+1)(n_z+1)$ for a 3D cube, the number of equations is roughly $2(2n_x+1)(2n_y+1)$ for a 2D plate, and $3(2n_x+1)(2n_y+1)(2n_y+1)$ for a 3D cube. Here n_x, n_y, and n_z represent the number of quadratic elements in each direction. Therefore, the number of equations outnumbers the number

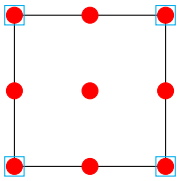


Figure 4: Schematic of the 2D quadratic element, with quadratic nodes (red circle) for displacements and linear nodes (blue empty square) for stress components.

of unknowns, and Equation (30) becomes an overconstrained problem, the least-square fitting is applied for solving it. And then the stress field is approximated linearly at each Gaussian point in each quadratic element.

Once we solve for $\boldsymbol{\sigma}_j^i$ from Equation (30), we can use the technique in Section 3.2.1 to pre-train the neural network. Although the least square approximation of the stress field is poor (the error can be larger than 100%), but the approximation is qualitatively correct and sufficient for obtaining a good initial guess.

4. Applications

In this section, we present numerical results from solid mechanics for the proposed NN based constitutive relations:

- Problem with a 1D truss coupon made of elasto-plastic materials under dynamic loading, which compares the proposed SPD-NN (17) and other neural network architectures, including $\boldsymbol{\sigma}$-NN (14) and $\Delta\boldsymbol{\sigma}$-NN (15).
- Problems with 2D thin plates made of hyperelastic materials, elasto-plastic materials, and multiscale fiber-reinforced materials under dynamic loading, which demonstrate the effectiveness of the proposed SPD-NN (17) for learning path-independent, path dependent (hysteresis), and multiscale constitutive relations.

In all problems, the training data and test data of the strain/stress fields and the displacement fields are generated numerically.

4.1. 1D Trusses with Elasto-plasticity

In this section, we consider 1D truss elements with a length $L_x = 1$ m, a cross section area $A = 0.005$ m^2 and a density 8000 kg/m^3. The truss coupon, clamped on the left end, is tested under 5 loading conditions (see Figure 5-left). The setting corresponds to a uniaxial tensile test. The prescribed time-dependent load force $\bar{\boldsymbol{t}}$ consists of both loading and unloading parts and takes the form

$$\bar{\boldsymbol{t}} = p \sin\left(\frac{t\pi}{T}\right), \; p = (0.4\,\mathrm{tid} + 1.6) \times 10^6 \text{ N}$$

here tid $= 1, 2, 3, 4$ and 5 are the test indices. The total simulation time is $T = 0.2$ s. The case tid $= 3$ is used as test set and all the other tests are used as training sets.

The trusses are made of elasto-plastic materials. The Young's modulus is

$$E = 200 \text{ GPa}$$

The yield function with isotropic hardening has the form

$$f = |\sigma| - \sigma_Y - K\alpha$$

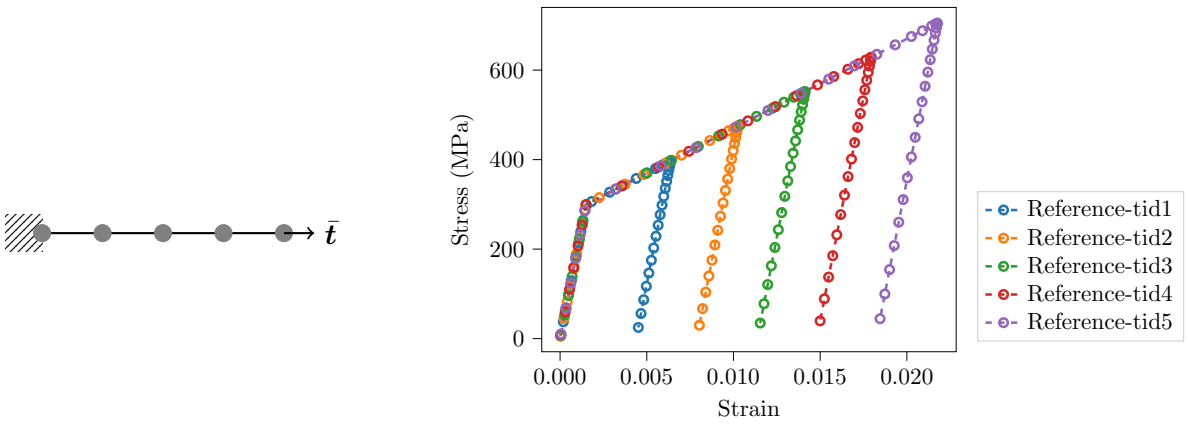


Figure 5: 1D truss problem setup (left): the truss coupon consists of 4 elements with left end fixed and force load on the right end; Extracted strain-stress curves for different loading conditions (right).

The yield strength is $\sigma_Y = 0.3$ GPa, the plastic modulus is $K = \frac{200}{9}$ GPa, the internal hardening variable follows the simplest evolutionary equation

$$\dot{\alpha} = \dot{\lambda}$$

The truss coupon consists of 4 truss elements, which are modeled as geometric nonlinear truss elements [40, p. 63]. The time step size is $\Delta t = 0.001$ s.

As for the SPD-NN Equation (23), the estimated Young's modulus is assumed to be known, which can be separately calibrated using small external loads (e.g., using the method proposed in [18]), and the estimated yield strength is $\tilde{\sigma}_Y = 0.1$ GPa. The transition function is

$$D(\sigma^n, \tilde{\sigma}_Y) = \texttt{sigmoid}\left(\frac{(\sigma^n)^2 - \tilde{\sigma}_Y^2}{d\tilde{\sigma}_Y^2}\right) \tag{31}$$

where $d = 0.1$ is the nondimensional parameter.

Because the strain and stress pairs can be extracted from the uniaxial tensile tests, we can apply the direct input-output data training method Section 3.2.1 to train the neural network. Figure 5-right shows the strain-stress curves exhibiting plastic loading and elastic unloading phenomena, which the neural networks are trained to learn. These curves start from the origin and rise with the slope E in the linear elastic region, until the stress reaches the yield stress. Then the curves enter the strain hardening region, where plastic deformations happen, until the elastic unloading. The slope of the unloading curve is typically equal to the slope in the elastic (initial) region of the stress-strain curve. The plasticity deformations result in permanent strains, which cannot be recovered by the elastic unloading. Since training neural networks involve highly non-convex optimization problems, we start from 10 different initial weights for all NN training.

To evaluate the quality of approximation, we propose two kinds of tests

1. NN test: extract the sequential strain-stress data (ϵ^i, σ^i) at each Gaussian quadrature point from the test data, and compare the predicted stress and the reference stress for each tuple $(\epsilon^{i+1}, \epsilon^i, \sigma^i; \sigma^{i+1})$.
2. NN-FEM test: embed the learned constitutive relation $\mathsf{M}_{\boldsymbol{\theta}}$ into the finite element framework, solve the governing equation of the truss coupon under the corresponding loading condition, and finally compare the predicted strain-stress paths and the truss deformations.

Because the optimization results depend on the initial guess for the neural network weights and biases, we pick the neural network with the minimal loss on the training set for the NN test and NN-FEM test. Other choices for selecting the candidate neural networks such as cross-validation [41] can also be used. It is worth noting that NN-FEM test is more challenging than the NN test, but is more relevant for predictive modeling. Indeed, the NN test does not take the numerical stability in the predictive modeling into account and only evaluates the NN's ability to fit the strain-stress curve. As we will see in the following examples, although SPD-NNs and other NN methods fit the strain-stress curves equally well in the NN test, SPD-NNs are significantly preferable due to their predicting power in new scenarios.

4.1.1. Comparison of SPD-NN, $\boldsymbol{\sigma}$-NN, and $\Delta\boldsymbol{\sigma}$-NN

The performance of the aforementioned neural network architectures, including the proposed SPD-NN, $\boldsymbol{\sigma}$-NN and $\Delta\boldsymbol{\sigma}$-NN, with different hyper-parameters are compared. The neural networks considered contain 1, 2, 3 and 4 hidden layers with 20 neurons in each layer, and `tanh` as the activation function is used. Both input and output are 1-dimensional. The losses at each training step are reported in Figure 6. Different initial weights lead to different local minima, as with any neural network based data-driven approaches. And deeper neural networks perform better in terms of the training error. SPD-NN achieves significantly smaller training errors, compared with $\boldsymbol{\sigma}$-NN and $\Delta\boldsymbol{\sigma}$-NN.

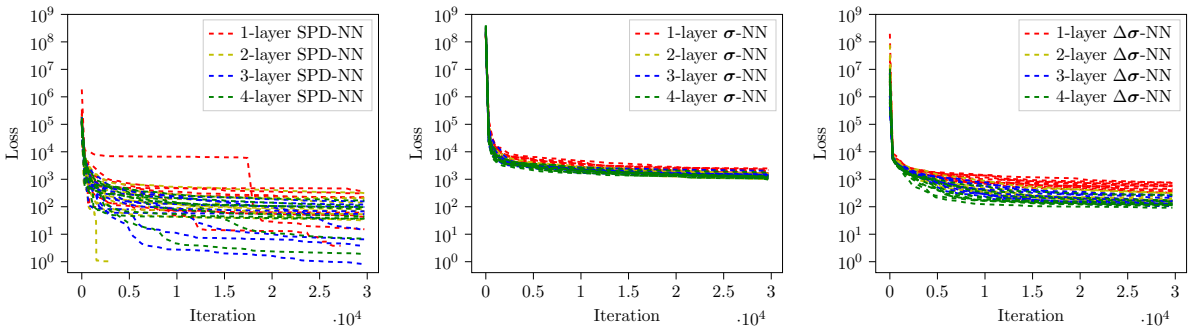


Figure 6: The losses evaluated on the training set at each training iteration with SPD-NN (left), $\boldsymbol{\sigma}$-NN (middle), and $\Delta\boldsymbol{\sigma}$-NN (right) with different number of layers for the 1D truss problem. Different curves correspond to different initial guesses.

For the NN test, the predicted strain-stress relations are reported in Figure 7. All deep neural networks (DNN) architectures give good results. The predictions of different neural network architectures are all very accurate and the errors are indiscernible in the plots.

In the NN-FEM test, the finite-element code uses the DNN predictions for integrating the equations of motion. The predicted displacement trajectories of the right end point and the predicted strain-stress curves for one Gaussian quadrature point on the right end element are reported in Figure 8 and Figure 9. Most of $\boldsymbol{\sigma}$-NN and $\Delta\boldsymbol{\sigma}$-NN architectures are found to be numerically unstable. We attribute the instability to the deviation of the strain and stress pairs in the test loading condition from the training set, where the numerical errors are accumulated during the prediction process. More specifically, the **violation of SPD constraints** on the tangent stiffness matrix for $\boldsymbol{\sigma}$-NN or $\Delta\boldsymbol{\sigma}$-NN makes them vulnerable to these errors and less robust to even slight extrapolations. In contrast, the proposed SPD-NN, which is supposed to be stable inside the plastic limit surface according to the second order work criterion (See Equation (12)), delivers numerically stable results and the predicted displacements and strain-stress curves overlap the exact ones.

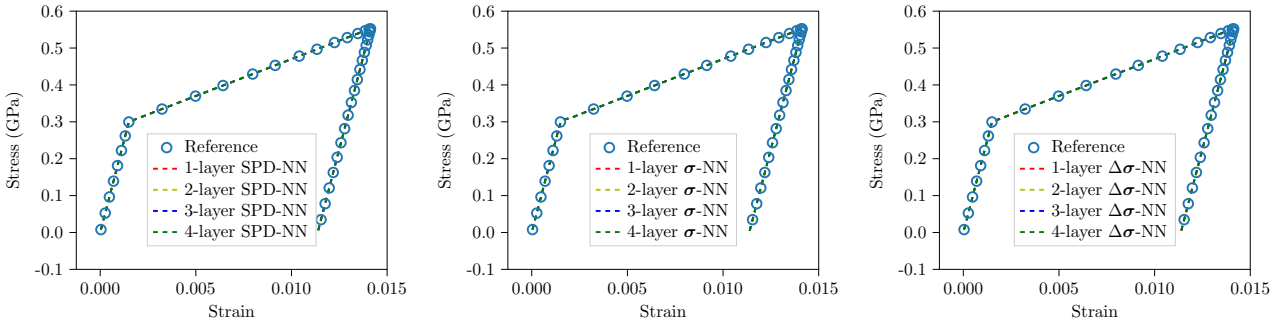


Figure 7: The strain stress curve results for SPD-NN (left), $\boldsymbol{\sigma}$-NN (middle), and $\Delta\boldsymbol{\sigma}$-NN (right) in the NN test.

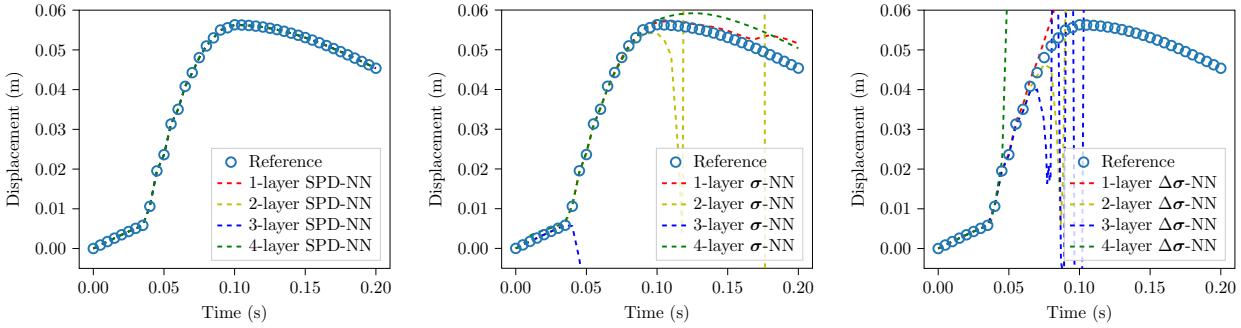


Figure 8: Displacement trajectories of the right end point for SPD-NN (left), $\boldsymbol{\sigma}$-NN (middle), and $\Delta\boldsymbol{\sigma}$-NN (right) in the NN-FEM test. As can be seen, some of the DNNs lead to unstable models as predicted by the second order work criterion.

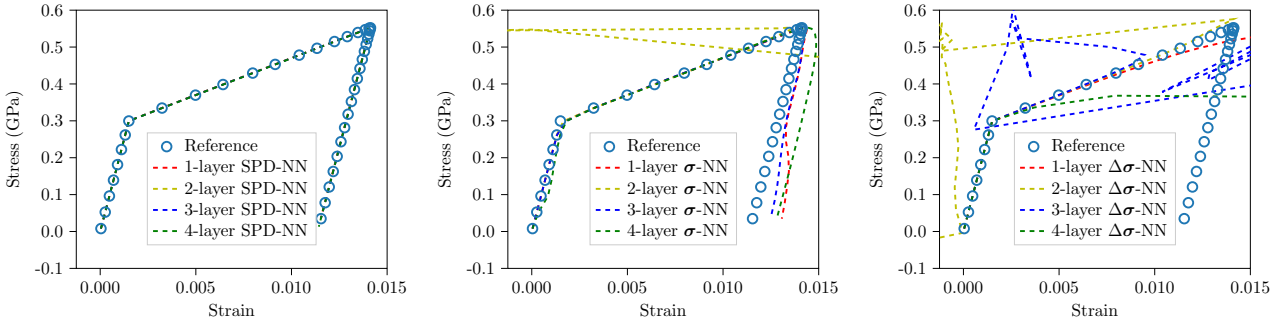


Figure 9: The strain stress curve of the NN-FEM test results obtained by using SPD-NN (left), $\boldsymbol{\sigma}$-NN (middle), and $\Delta\boldsymbol{\sigma}$-NN (right). The middle and right models are unstable as predicted by the second order work criterion.

	Depth\Width	2	10	20	40
`tanh`	1	1.1×10^{-5}	4.2×10^{-5}	2.0×10^{-5}	3.0×10^{-5}
	3	4.7×10^{-5}	1.8×10^{-6}	1.3×10^{-6}	NaN
	8	5.5×10^{-4}	1.3×10^{-5}	NaN	NaN
	20	8.8×10^{-6}	5.9×10^{-5}	NaN	NaN
ReLU	1	9.3×10^{-3}	3.7×10^{-5}	6.1×10^{-5}	4.2×10^{-3}
	3	9.3×10^{-3}	3.1×10^{-3}	4.7×10^{-3}	5.3×10^{-3}
	8	9.1×10^{-3}	1.8×10^{-4}	6.1×10^{-3}	1.0×10^{-2}
	20	3.5	5.9×10^{-6}	3.3×10^{-3}	NaN
leaky ReLU	1	4.6×10^{-3}	4.8×10^{-3}	4.3×10^{-3}	1.2×10^{-2}
	3	8.9×10^{-3}	1.2×10^{-2}	1.0×10^{-2}	2.1×10^{-3}
	8	5.3×10^{-3}	5.0×10^{-3}	6.6×10^{-3}	5.4×10^{-3}
	20	9.3×10^{-3}	4.4×10^{-3}	8.6×10^{-3}	1.8×10^{-2}
SELU	1	1.0×10^{-2}	6.6×10^{-4}	2.2×10^{-4}	2.2×10^{-4}
	3	9.6×10^{-3}	9.8×10^{-5}	4.3×10^{-4}	3.8×10^{-4}
	8	9.2×10^{-5}	7.6×10^{-3}	4.8×10^{-5}	NaN
	20	8.3×10^{-6}	5.8×10^{-5}	NaN	NaN
ELU	1	4.2×10^{-3}	4.6×10^{-3}	5.0×10^{-3}	6.6×10^{-3}
	3	6.8×10^{-5}	5.7×10^{-3}	4.7×10^{-4}	4.1×10^{-3}
	8	8.0×10^{-6}	8.4×10^{-6}	NaN	NaN
	20	6.6×10^{-5}	NaN	NaN	NaN

Table 1: Mean squared errors of the predicted displacements at the final time T for different neural network architectures, including `tanh`, ReLU (rectified linear unit), ELU (exponential linear unit), SELU (scaled exponential linear unit), and leaky ReLU (leaky rectified linear unit) with leakage 0.1. The width is the number of activation nodes in each hidden layer. The depth is the number of hidden layers. "NaN" denotes the numerical simulation fails (e.g., due to numerical instability) using the trained NN-based constitutive relations.

4.1.2. Comparison of Different Neural Network Architectures for SPD-NN

In this experiment, we consider different neural network architectures. We vary widths, depths and activation functions of the neural network used in SPD-NN, while keeping other settings the same as Section 4.1.1. The errors in the NN-FEM test in terms of mean squared errors of the predicted displacements at the final time T are shown in Table 1. We see that the `tanh` activation function is in general more accurate than others if appropriate widths and depths of the neural network are chosen (For this case, the training set consists of 800 data, the number of neural network parameters is not supposed to outnumbers it too much). However, ReLU and leaky ReLU are more robust for deep and wide neural networks, despite being less accurate.

4.1.3. Choice of $\tilde{\sigma}_Y$ and d

We consider the impact of $\tilde{\sigma}_Y$ and d in the transition function D (see Equation (31))

$$D(\sigma^n, \tilde{\sigma}_Y) = \texttt{sigmoid}\left(\frac{(\sigma^n)^2 - \tilde{\sigma}_Y^2}{d\,\tilde{\sigma}_Y^2}\right)$$

on the accuracy of the SPD-NN. As mentioned before, $\tilde{\sigma}_Y$ controls where the transition from the elastic form (Equation (21)) to the plastic form (Equation (22)) happens and d controls the sharpness of the transition (see Figure 2). In this numerical experiment, we use a fully connected

neural network with 3 hidden layers, 20 neurons in each layer, and the `tanh` activation function. We vary $\tilde{\sigma}_Y$ and d in Equation (31).

The error plot is shown in Figure 10, where the error metric is the same as Section 4.1.2. The reported errors are the average errors of 10 simulations with different initial guesses. We can see that the accuracy of the SPD-NN is less sensitive to d. Additionally, as long as we choose a small enough $\tilde{\sigma}_Y$ so that the corresponding plastic form (Equation (22)) covers the plasticity regime, the SPD-NN is sufficiently expressive to approximate the constitutive relation. This justifies our choices $d = 0.1$ and $\tilde{\sigma}_Y = 0.1$ GPa.

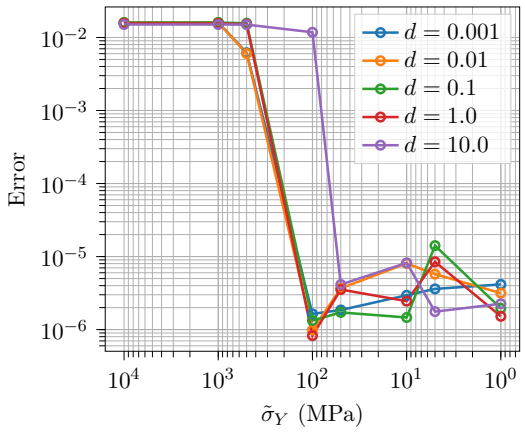


Figure 10: Impacts of $\tilde{\sigma}_Y$ and d on the approximation accuracy of SPD-NNs.

4.1.4. The Effect of the Training Dataset

In this experiment, we investigate the effect of the training dataset, including its size and distance to the test dataset, on the performance of the aforementioned neural network models. The prescribed time-dependent load force $\bar{\boldsymbol{t}}$ consists of both loading and unloading parts and takes the form

$$\bar{\boldsymbol{t}} = p \sin\left(\frac{t\pi}{T}\right), \quad p = \left(\frac{\text{tid} - 1}{12} + 1.6\right) \times 10^6 \text{ N}$$

here $\text{tid} = 1, 2, \ldots, 25$ are the test indices. The total simulation time is $T = 0.2$ s. The case $\text{tid} = 13$ is used as the test dataset. For fairness, we start from the same deep neural network architecture (3 hidden layers, 20 neurons per layer, `tanh` activation function) and weight initialization. We run the optimization for 30,000 iterations using the L-BFGS-B optimizer.

To understand the effect of the training dataset size, we train σ-NN, $\Delta\sigma$-NN, and SPD-NN using n datasets. In each case, we use the following tids (test IDs)

- $n = 1$; $\text{tid} = 12$
- $n = 2$; $\text{tid} = 10, 15$
- $n = 4$; $\text{tid} = 1, 9, 17, 25$
- $n = 9$; $\text{tid} = 3k + 1, k = 0, 1, \ldots, 8$
- $n = 14$; $\text{tid} = 1, 3, 5, 7, 8, 10, 12, 14, 16, 18, 19, 21, 23, 25$
- $n = 24$; $\text{tid} = k, k \neq 13$

The errors in the NN test (σ-NN and $\Delta\sigma$-NN suffer instability for the NN-FEM test) in terms of the mean squared errors of the predicted stresses are presented in Figure 11. See page 14 for a definition of these tests. Since tid = 13 does not appear in the training dataset, the mean squared error can be interpreted as the test error. From Figure 11, we can see that SPD-NN outperforms both σ-NN and $\Delta\sigma$-NN for moderate dataset sizes. For small dataset sizes, SPD-NN can achieve reasonable accuracy, except for the case where only 1 dataset is used. This illustrates that potentially our approach does not require a large number of data to have good prediction power. This is beneficial for engineering applications because, in practice, data may be scarce or expensive to collect.

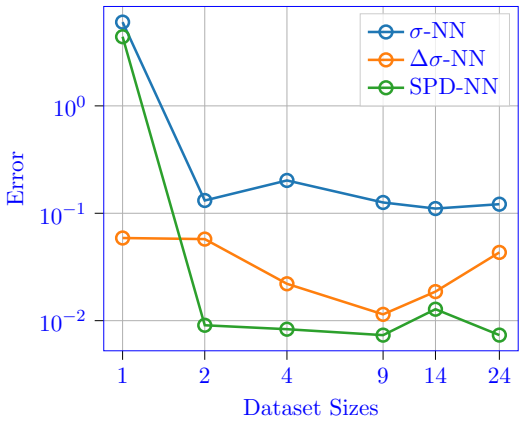


Figure 11: Mean squared errors for different neural network models and training dataset sizes. We see that SPD-NN achieves best accuracy compared with other models.

To understand the effect of the distance between the training data and test data (or the quality of the training data) on the performance of the aforementioned neural network models, we conduct the following experiments with small training dataset sizes. The tids are indicated for each n:

- $n = 1$; tid $= 12$
- $n = 1$; tid $= 1$. Note this dataset is far from the test dataset tid $= 13$.
- $n = 2$; tid $= 10, 15$
- $n = 2$; tid $= 1, 25$. Note these two datasets are far from the test dataset tid $= 13$.

The mean squared errors are shown in Table 2. We can see that for all cases, using datasets that are closer to the test dataset yields better test results. Therefore, we conclude that although more data help train SPD-NN better, the quality of the training data also matters. And the SPD-NN is more robust with respect to the quality of the training dataset when a moderate dataset size is used.

4.2. 2D Thin Plate with Different Materials

In this section, we consider thin plate coupons of size $L_x = 10$ cm by $L_y = 5$ cm with the plane stress assumption ($L_z = 0.1$ cm). These plates are made of different materials, including hyperelastic material (finite deformation), elasto-plastic material (infinitesimal deformation), and fiber-reinforced multiscale elasto-plastic material (infinitesimal deformation). For each case, the plate is tested under 13 loading conditions as depicted in Figure 12. The prescribed time-dependent load force $\bar{\boldsymbol{t}} \in \mathcal{R}^2$ consists of both loading and unloading parts and takes the form

$$\bar{\boldsymbol{t}} = p \sin\left(\frac{t\pi}{T}\right)$$

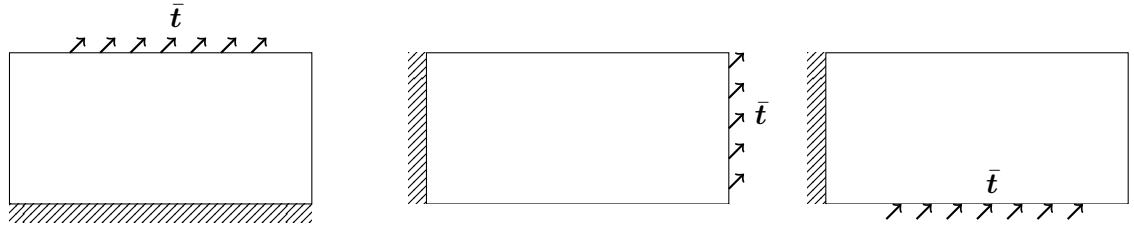
	$n=1$		$n=2$	
Training Dataset	tid $=12$	tid $=1$	tid $=10,15$	tid $=1,25$
σ-NN	6.02	106365.00	0.13	86.50
$\Delta\sigma$-NN	0.06	658.85	0.06	1.00
SPD-NN	4.40	914.90	0.01	0.06

Table 2: Mean squared errors for small dataset sizes. Given the dataset size n, when the training dataset is close to the test dataset, the test errors are in general smaller.

here $p \in \mathcal{R}^2$ is the loading parameter vector, as follows,

- A) clamp on the bottom edge and impose force load on the top edge.
  A1: $(0, p_1)$, A2: $(0, -p_1)$, A3: $(p_3, 0)$, A4: $(-p_3, 0)$, A5: $(p_3/\sqrt{2}, p_1/\sqrt{2})$, and A6: $(0.75p_3, 0)$.
- B) clamp on the left edge and impose force load on the right edge.
  B1: $(p_1, 0)$, B2: $(-p_1, 0)$, B3: $(0, p_2)$, B4: $(0, -p_2)$, B5: $(p_1/\sqrt{2}, p_2/\sqrt{2})$, and B6: $(0, 0.75p_2)$.
- C) clamp on the left edge and impose force load on the bottom edge.
  C1: $\left(0, \frac{p_2 L_x}{\sqrt{2\pi}\sigma_X} \exp\left(\frac{-(x-x_0)^2}{\sigma_X^2}\right)\right)$, with $x_0 = \frac{5L_x}{6}$ and $\sigma_X = 0.2L_x$.

The total simulation time is $T = 0.2s$. We use A1-A5 and B1-B5 as training data and A6, B6, and C1 as test data. Both training procedures discussed in Sections 3.2.1 and 3.2.2 are applied. For the direct input-output data training, the strain-stress sequential data are extracted from all Gaussian points in the training sets. For the indirect data training, the full-filed displacement fields on the 21 by 11 grid (0.5 cm interval) from the training data are extracted. Therefore, this approach is potentially applicable to experimental data. The pre-training is required to obtain good initial guesses.



(a) A (b) B (c) C

Figure 12: Schematic of the boundary conditions of the thin plate tests.

4.2.1. Hyperelasticity

The plate is made of the incompressible Rivlin-Saunders material [42, 43] with the density $\rho = 800$ kg/m^3 and the energy density function

$$w = c_1(I_1 - 3) + c_2(I_2 - 3)$$

Here $c_1 = 0.1863$ MPa and $c_2 = 0.00979$ MPa, and I_1, I_2, I_3 are three scalar invariants of the right Cauchy-Green stretch tensor $\boldsymbol{C} = \boldsymbol{F}\boldsymbol{F}^T = 2\boldsymbol{\epsilon} + 1$, where

$$I_1 = \mathrm{tr}\boldsymbol{C} \qquad I_2 = \frac{1}{2}[(\mathrm{tr}\boldsymbol{C})^2 - \mathrm{tr}\boldsymbol{C}^2] \qquad I_3 = J^2 = \det \boldsymbol{C}$$

The incompressibility implies that $J = 1$. The plate is assumed to undergo finite deformations (see Remark 1), and the second Piola-Kirchhoff stress tensor reads

$$\boldsymbol{S} = \frac{\partial w}{\partial \boldsymbol{\epsilon}} + \lambda_J \frac{\partial J}{\partial \boldsymbol{\epsilon}} \tag{32}$$

here λ_J is the Lagrangian multiplier, which can be calculated based on the plane stress assumption $\boldsymbol{S}_{33} = 0$. The plate domain is discretized by 20×10 quadratic quadrilateral elements. The time step size is $\Delta t = 0.001$ s. The data sets are generated with load parameters $(p_1, p_2, p_3) = (44800, 4480, 16800)$ N/m.

Both the direct data training approach in [2] Section 3.2.1 and the indirect data training approach in [2] Section 3.2.2 are applied to train a SPD-NN:

$$\boldsymbol{S}^{n+1} = \mathsf{L}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1})\mathsf{L}_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}^{n+1})^T(\boldsymbol{\epsilon}^{n+1} - \boldsymbol{\epsilon}^n) + \boldsymbol{S}^n \tag{33}$$

where the neural network consists of 4 hidden layers and 20 neurons in each layer.

The predicted trajectories of displacements at the top-right and top-middle points as a function of time and the references for all test cases are depicted in Figure 13. All predicted results are in good agreement with the references, and the direct input-output data training approach leads to slightly better results for case C1. The predicted von Mises stress fields at $t = \frac{T}{2}$ for all test cases and the references are depicted in Figure 14.

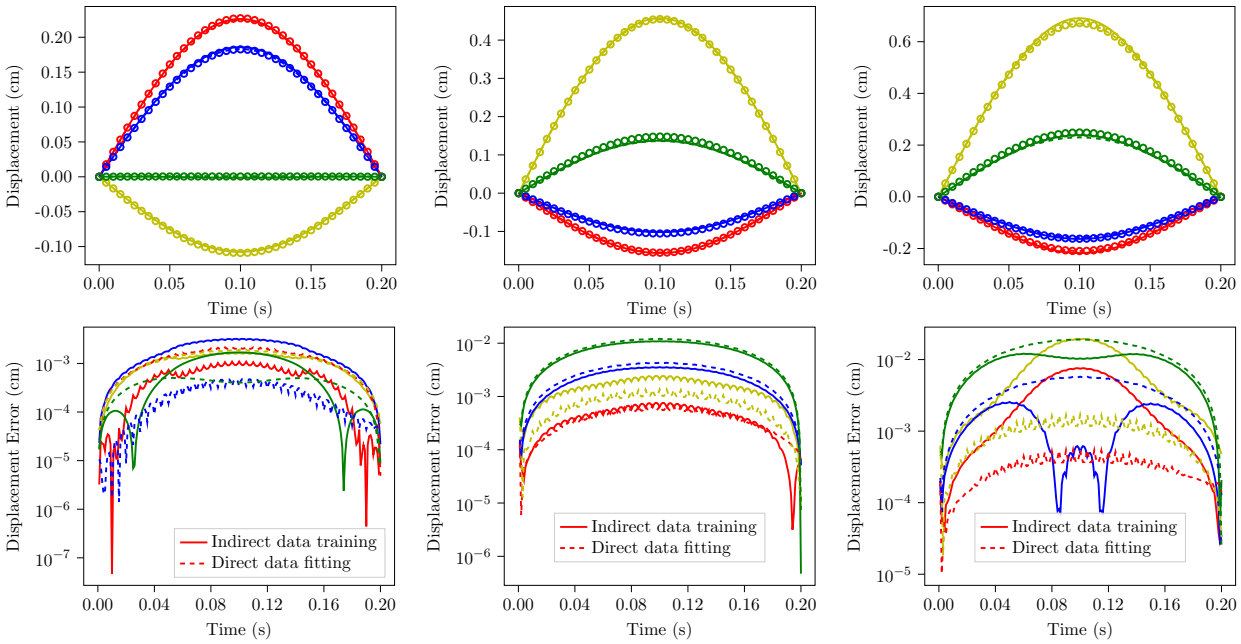


Figure 13: Top: Trajectories of displacement at top-right (red: u_x, yellow: u_y) and top-middle points (blue: u_x, green: u_y) of the 2D hyperelastic plate for test A6 (left), B6 (middle) and C1 (right), defined on page 19. The reference solutions are marked by empty circles, the solutions obtained by the SPD-NN trained using indirect data are marked by solid lines, and the solutions obtained by SPD-NN trained with direct data are marked by dashed lines. Bottom: The absolute errors of displacements for each cases.

4.2.2. Elasto-Plasticity

The plate is made of titanium, which is assumed to be elasto-plastic material with density $\rho = 4200$ kg/m^3. The constitutive relation is

$$\boldsymbol{\sigma} = \mathsf{C}\boldsymbol{\epsilon}$$

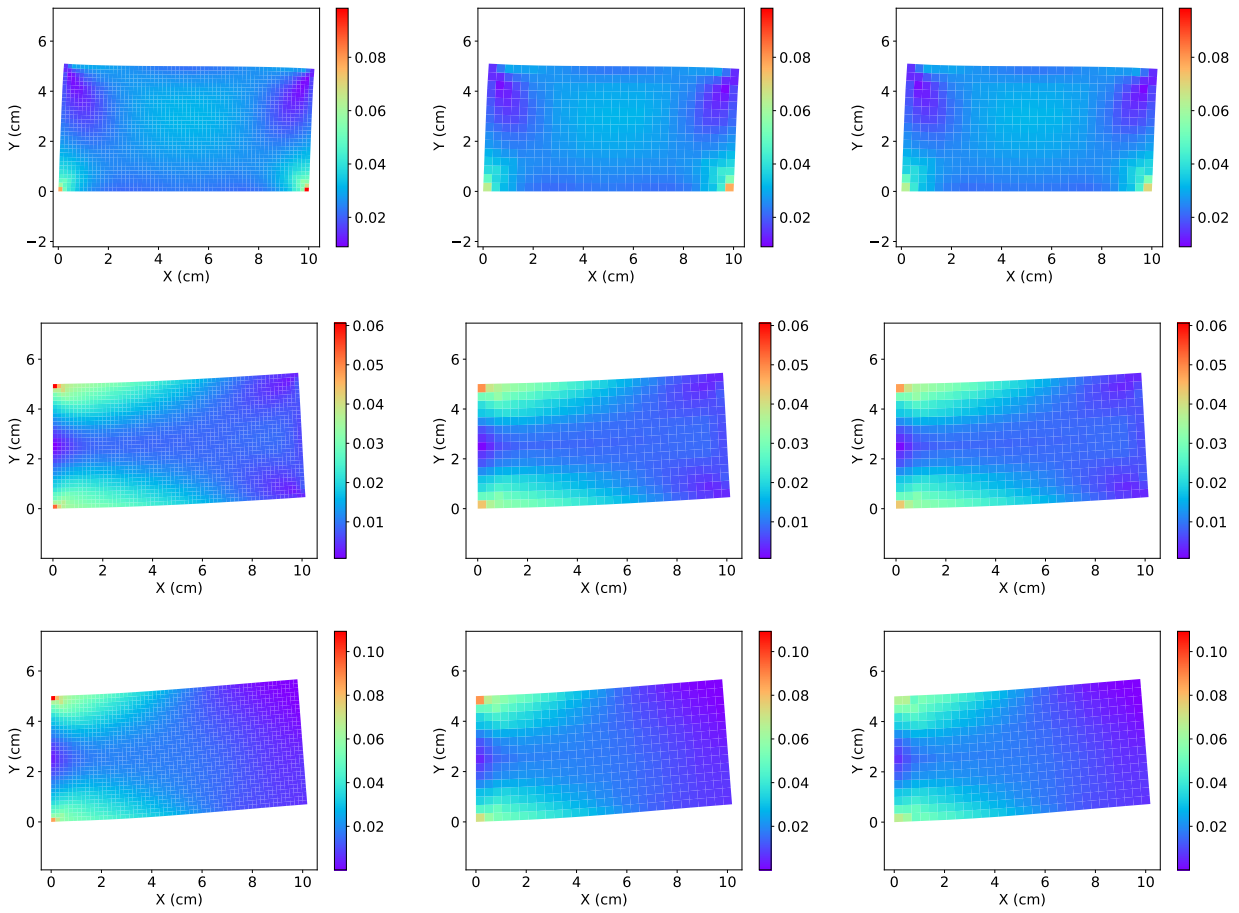


Figure 14: The von Mises stress (MPa) fields at $t = \frac{T}{2}$ for the 2D hyperelastic plate for test A6 (top), B6 (middle) and C1 (bottom), defined on page 19. From left to right: reference solutions (on a fine mesh), solutions obtained by SPD-NN trained with direct data, solutions obtained by SPD-NN trained with indirect data.

here C denotes the isotropic plane stress stiffness tensor with Young's modulus $E = 100$ GPa and Poisson's ratio $\nu = 0.35$. The von Mises yield function with isotropic hardening has the form

$$f = \sqrt{\sigma_{11}^2 - \sigma_{11}\sigma_{22} + \sigma_{22}^2 + 3\sigma_{12}^2} - \sigma_Y - K\alpha \tag{34}$$

The yield strength $\sigma_Y = 0.97$ GPa and the plastic modulus $K = 10$ GPa, the internal hardening variable α follows the simplest evolutionary equation

$$\dot{\alpha} = \dot{\lambda} \tag{35}$$

This plate domain is discretized by 20×10 quadratic quadrilateral elements. And the time step size is $\Delta t = 0.001s$.

As for the SPD-NN Equation (23), the estimated yield strength is $\tilde{\sigma}_Y = 0.32$GPa, the transition function is

$$D(\sigma_{\text{vm}}^n, \tilde{\sigma}_Y) = \texttt{sigmoid}\left(\frac{{\sigma_{\text{vm}}^n}^2 - \tilde{\sigma}_Y^2}{d\tilde{\sigma}_Y^2}\right)$$

here σ_{vm}^n is the computed von Mises stress at the previous time step and $d = 0.1$ denotes the nondimensional parameter. The tangent stiffness matrix $\mathsf{C}_{\boldsymbol{\theta}}$ in the linear region is first estimated as following, and then used as constant in Equation (23).

Linear region. The data sets are generated with load parameters

$$(p_1, p_2, p_3) = (0.16, 0.016, 0.06) \text{ GN/m}$$

which are small enough to maintain the deformations in the linear region.

The indirect data training approach in Section 3.2.2 is applied to extract the tangent stiffness matrix, and we obtain the following estimation

$$\mathsf{C}_\theta = \begin{bmatrix} 1.04064 \times 10^6 & 2.09077 \times 10^5 & 0.0 \\ 2.09077 \times 10^5 & 1.041146 \times 10^6 & 0.0 \\ 0.0 & 0.0 & 4.19057 \times 10^5 \end{bmatrix} \tag{36}$$

Based on Young's modulus and Poisson's ratio of the material, the tangent stiffness matrix is

$$\mathsf{C}_{\text{ref}} = \begin{bmatrix} 1.04167 \times 10^6 & 2.08333 \times 10^5 & 0.0 \\ 2.08333 \times 10^5 & 1.04167 \times 10^6 & 0.0 \\ 0.0 & 0.0 & 4.16667 \times 10^5 \end{bmatrix} \tag{37}$$

For each components, the relative error is less than one percent. These errors are introduced by the discretization, including the interpolation of the displacement field on the observation grid and the estimation of the acceleration Equation (26). It is worth mentioning, the direct input-output data training delivers exactly the same stiffness matrix as the reference.

Nonlinear region. The data sets are generated with load parameters

$$(p_1, p_2, p_3) = (1.6, 0.16, 0.6) \text{ GN/m}$$

which are 10 times larger than these in the linear region. Both direct input-output data training (Section 3.2.1) and indirect data training (Section 3.2.2) are applied to train a SPD-NN with 5 hidden layers and 20 neurons in each layer. The predicted trajectories of the displacements at top-right and top-middle points for all test cases are depicted in Figure 15, along with the references. SPD-NNs trained with both methods are able to predict the initial elastic behavior, the strain-hardening region, and the unloading behavior. The SPD-NN obtained by the direct input-output data training performs better especially for the prediction of the yield strength the strain-hardening behavior. The predicted von Mises stress fields at $t = \frac{T}{2}$ and the references for all test cases are depicted in Section 4.2.2. Reasonable agreements are achieved.

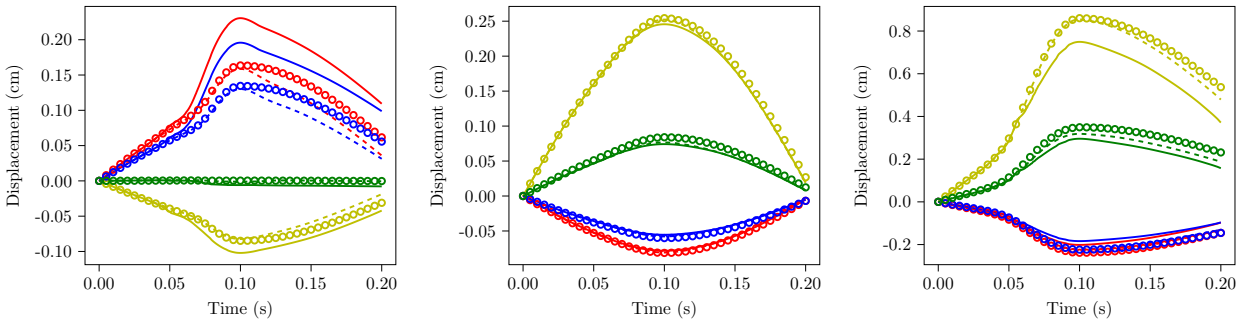


Figure 15: Trajectories of displacement at top-right (red: u_x, yellow: u_y) and top-middle points (blue: u_x, green: u_y) of the 2D elasto-plastic plate for test A6 (left), B6 (middle) and C1 (right), defined on page 19. The reference solutions are marked by empty circles, the solutions obtained by the CholNN trained using indirect data are marked by solid lines, and the solutions obtained by CholNN trained with direct data are marked by dashed lines. Note that the plasticity model takes into account the full history of the deformation while the DNN models take as input only the value from the last step.

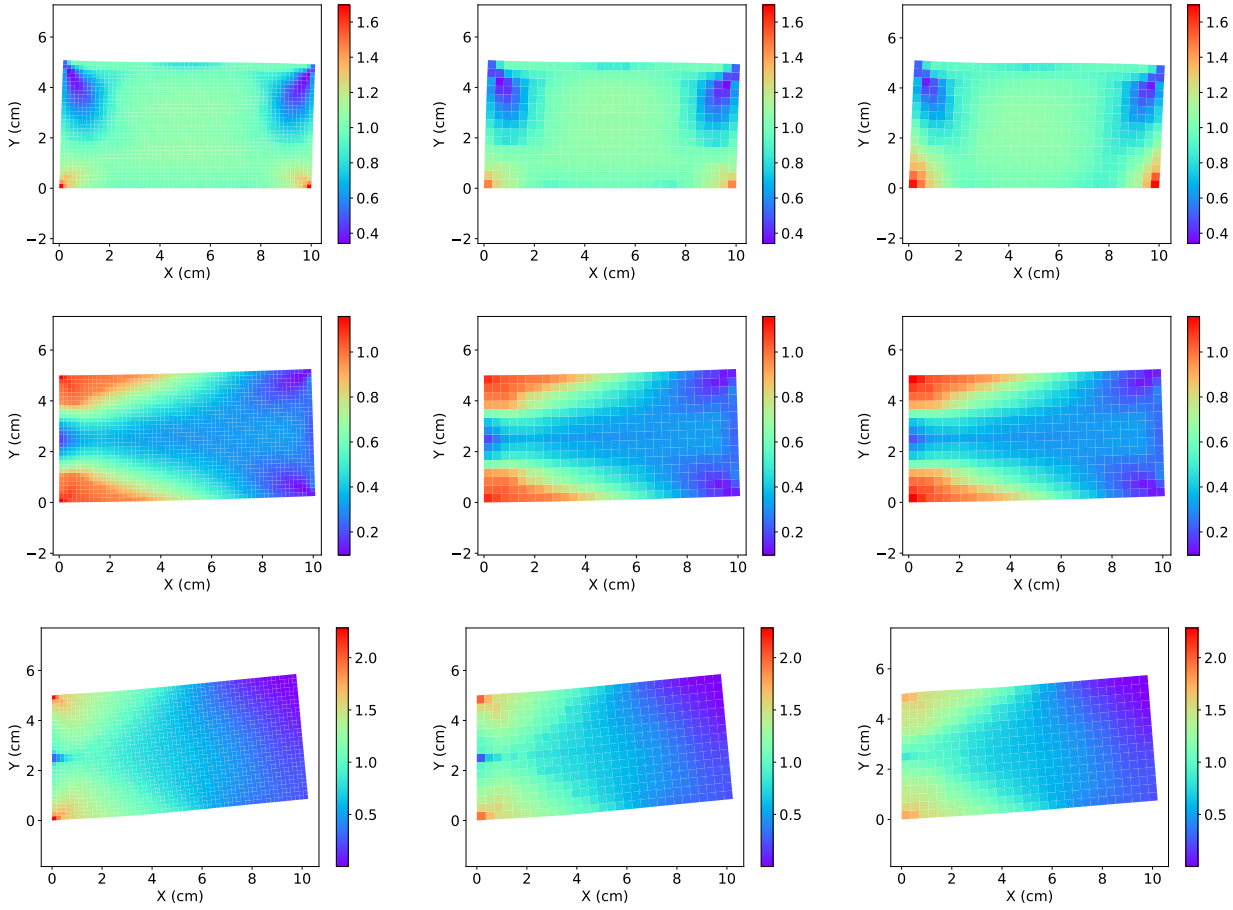


Figure 16: The von Mises stress (GPa) fields at $t = \frac{T}{2}$ for the 2D elasto-plastic plate for test A6 (top), B6 (middle) and C1 (bottom), defined on page 19. From left to right: reference solutions (on a fine mesh), solutions obtained by SPD-NN trained with direct data, solutions obtained by SPD-NN trained with indirect data.

4.2.3. Multiscale Fiber Reinforced Elasto-plasticity

The plate is made of the titanium—the same as Section 4.2.2—but reinforced by fibers made of SiC, which are assumed to be isotropic and elastic with

$$\rho = 3200 \text{ kg/m}^3, \ E = 400 \text{ GPa, and } \nu = 0.35$$

These fibers are square shaped and uniformly distributed in the plate, with a diameter $d = 0.25$ cm and a fraction 25%. There are in total 800 fibers, as shown in Figure 17. This plate domain is discretized by 200×400 quadratic quadrilateral elements, and 25 elements for each fiber. And the time step size is $\Delta t = 0.001s$.

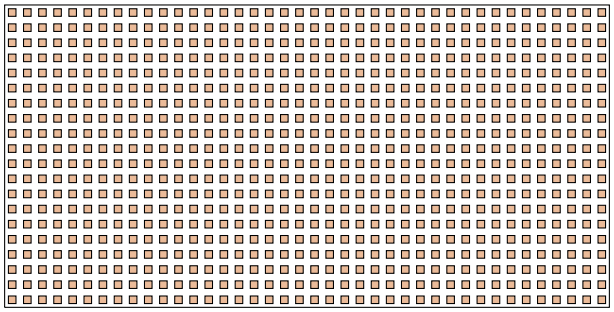


Figure 17: Schematic of the fiber (orange) reinforced thin plate.

As for training SPD-NNs, the estimated yield strength and the transition function are the same as these in Section 4.2.2. The tangent stiffness matrix $\mathsf{C}_{\boldsymbol{\theta}}$ in the linear region is first calibrated and then fixed as constant in Equation (23) when training the SPD-NN.

Linear region. The data sets are generated with load parameters

$$(p_1, p_2, p_3) = (0.16, 0.016, 0.06) \text{ GN/m}$$

which are small enough to maintain the deformations in the linear region. The indirect data training approach in Section 3.2.2 is applied to extract the following predicted tangent stiffness matrix,

$$\mathsf{C}_\theta = \begin{bmatrix} 1.335174 \times 10^6 & 3.26448 \times 10^5 & 0.0 \\ 3.26448 \times 10^5 & 1.326879 \times 10^6 & 0.0 \\ 0.0 & 0.0 & 5.26955 \times 10^5 \end{bmatrix} \tag{38}$$

The predicted linear constitutive relation Equation (19) is verified on the test set. The predicted displacements at top-right and top-middle points as a function of time and the references for all test cases are depicted in Section 4.2.3. The corresponding von Mises stress fields at $t = \frac{T}{2}$ are reported in Section 4.2.3. The SPD-NN based homogenized model delivers similar results as the high-resolution multiscale model.

Nonlinear region. The data sets are generated with load parameters

$$(p_1, p_2, p_3) = (1.6, 0.16, 0.6) \text{ GN/m}$$

which are 10 times larger than those in the linear region. Only the indirect data training approach is applied to train a SPD-NN with 5 hidden layers and 20 neurons in each layer. To enable direct input-output data training, homogenization is required to generate strain-stress data from RVE simulations [13, 14, 21, 6], or extract strain-stress data from direct numerical simulations. This

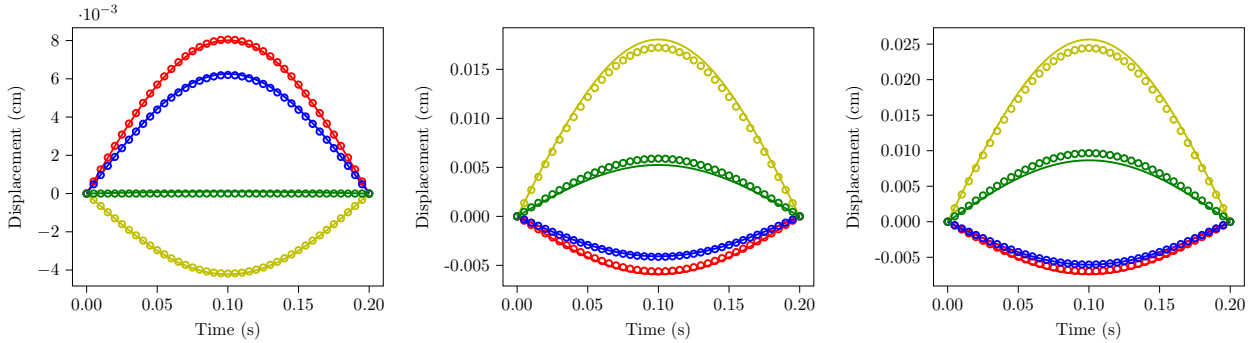


Figure 18: Trajectories of displacement at top-right (red: u_x, yellow: u_y) and top-middle points (blue: u_x, green: u_y) of the 2D multiscale plate in the linear region for test A6 (left), B6 (middle) and C1 (right), defined on page 19. The reference solutions are marked by empty circles and the solutions obtained by the SPD-NN trained with indirect data are marked by solid lines.

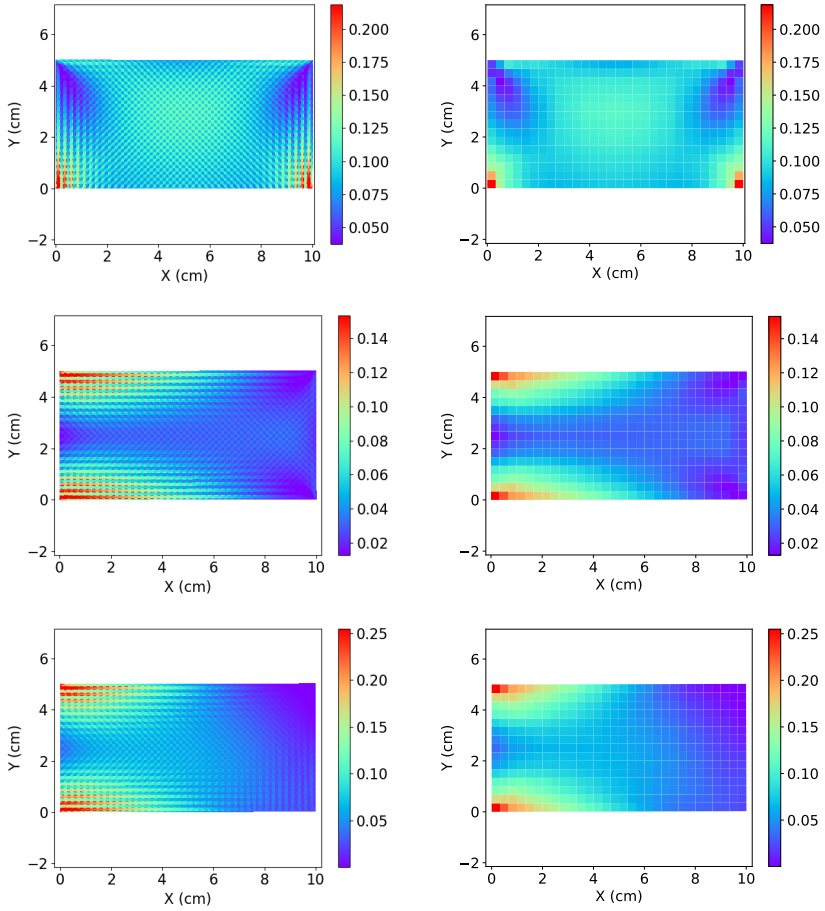


Figure 19: The von Mises stress (GPa) fields at $t = \frac{T}{2}$ for the 2D multiscale plate in the linear region for test A6 (top), B6 (middle) and C1 (bottom), defined on page 19. From left to right: reference solutions (on a fiber-resolved mesh) and solutions obtained by SPD-NN trained with indirect data.

may be challenging in the context of experimental data, which necessitates the indirect training approach.

The predicted trajectories of displacements at top-right and top-middle points for all test cases are depicted in Figure 20, along with the references. Comparing with the previous elasto-plasticity case, all displacements are smaller due to the SiC fiber reinforcement. The proposed SPD-NN gives a satisfactory approximation of the initial elastic behavior, the strain-hardening region, and the following unloading behavior. The predicted von Mises stress fields at $t = \frac{T}{2}$ and the references for all test cases are depicted in Figure 21. The predicted and simulated homogenized stress fields are in reasonably good agreement. Although the solutions obtained by SPD-NNs do not capture local large stress concentrations near each fiber (at the level of the microstructure), the local recovery techniques [44] can be applied to estimate these local stress concentrations. For example, a RVE simulation can be conducted to estimate local stresses using the local strain from the coarse homogenized solution.

Accelerating Simulations with Neural Network Surrogates for Constitutive Modeling. It is also worth noting that each SPD-NN-based simulation is several order magnitude faster than the corresponding fiber resolved simulations. The CPU time for both the SPD-NN-based simuations and fiber resolved simulations are shown in Figure 22. Note as we increase the external load, the CPU time increases because more elements undergo plastic deformations, which requires more expensive Raphson-Newton iterations in the numerical simulations. Still, the dramatic acceleration from around 24 hours to just a few minutes is impressive. However, the acceleration should be carefully interpreted in the context of surrogate models. First, the current benchmarks are based on serial execution, where the state-of-the-art FEM simulations usually involve parallelization. Nevertheless, these parallelization techniques for fiber resolved simulations are also directly applicable to SPD-NNs. Second, the speed of the fiber resolved simulations depend on the required resolution (e.g., the resolution on each fiber). If we use very coarse grids, the speed of the fiber resolved simulations may be comparable to or even faster than the SPD-NN-based simulations, though the coarse grids raise concerns for accuracy. Third, we have not compared the present model with other state-of-the-art accelerating techniques for multiscale modeling, for example projection-based model order reduction [45] and self-consistent clustering [46]. Finally, as with any NN-based surrogate models, the SPD-NN only works on test data that does not deviate too much from the training data, while fiber resolved simulations are usually considered to be general and applicable in a much wider context.

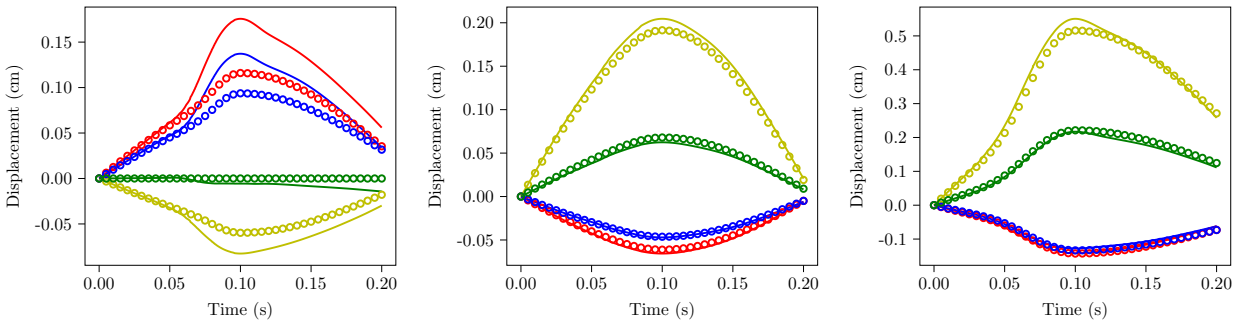


Figure 20: Trajectories of displacement at top-right (red: u_x, yellow: u_y) and top-middle points (blue: u_x, green: u_y) of the 2D multiscale plate in nonlinear region for test A6 (left), B6 (middle) and C1 (right), defined on page 19. The reference solutions are marked by empty circles and the solutions obtained by the SPD-NN trained with indirect data are marked by solid lines.

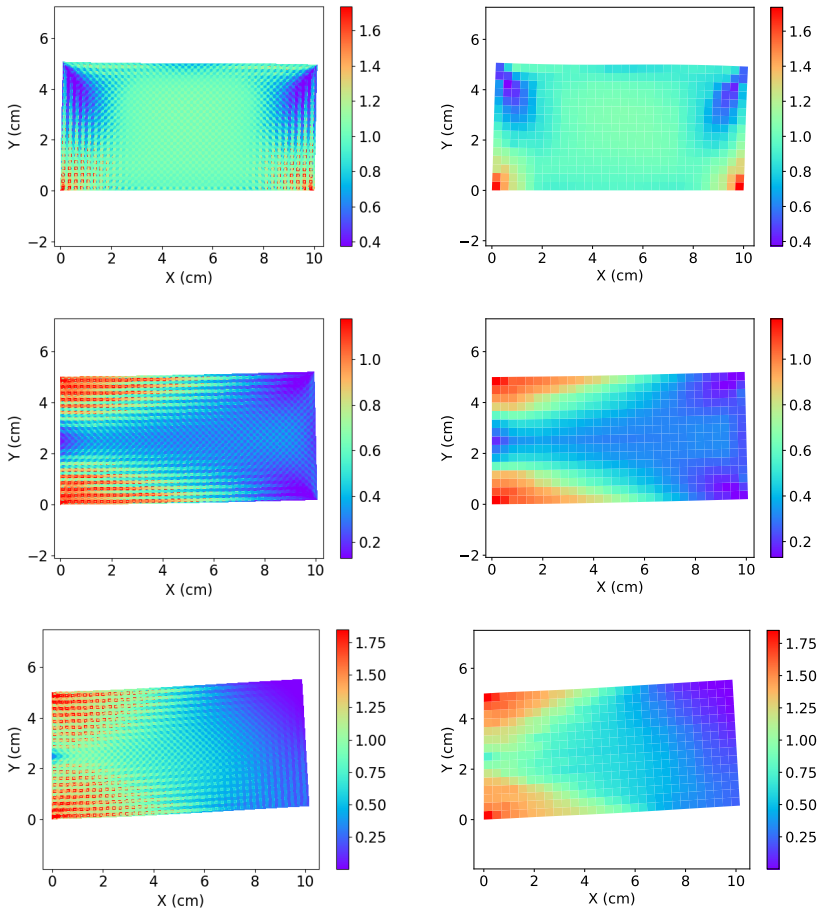


Figure 21: The von Mises stress (GPa) fields at $t = \frac{T}{2}$ for the 2D multiscale plate in nonlinear region for test A6 (top), B6 (middle) and C1 (bottom), defined on page 19. From left to right: reference solutions (on a fiber-resolved mesh) and solutions obtained by SPD-NN trained with indirect data.

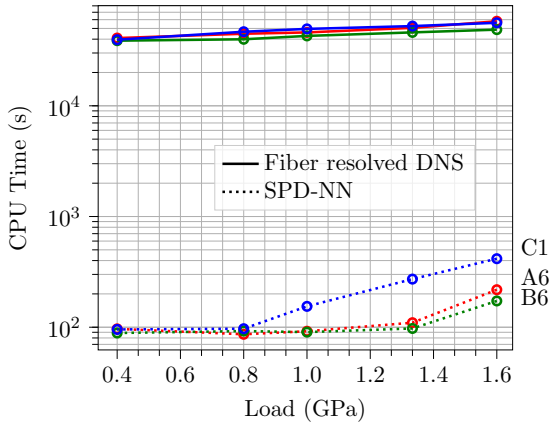


Figure 22: CPU time for simulations using fiber resolved DNS (direct numerical simulation) and the SPD-NN surrogate model. The same color denotes the same test case (C1, A6, or B6).

5. Discussion of Neural Networks

The introduction and benchmarking of SPD-NNs raise many questions, some of which are partially answered in this paper while some others are worthwhile to be investigated further. We believe that the following discussion is important and central to the application of NN-based constitutive modeling and should be careful dealt for the development of SPD-NNs and other NN-based approaches.

Function approximators. In the present work, neural network is used as a basis function to approximate a complex constitutive relation, the strain-stress relation. Many other basis functions, including piecewise linear functions, radial basis functions, and radial basis function networks, can also be applied for the approximation of the Cholesky factor. The choice of neural networks is well-versed and justified by the following three reasons.

- First, the distribution of the strain data—the input to the neural network—is determined by experiment records, which in general does not evenly spread over the domain. The comparison study presented in [18] illustrates that neural network outperforms the other basis functions in terms of regularization and generalization properties when the data distribution is ill-behaved.
- Second, the strain-stress curves are non-smooth in the context of plastic deformations and elastic unloading. As a result of the non-smoothness, the approximation efficiency of typical basis functions is usually compromised. Nevertheless, neural networks exhibit the potential to capture the sharp transitions in the strain-stress relations and performs reasonably well for the non-smooth data.
- Third, the input and the output dimensions are relatively high (e.g., the input is 9D and the output is 4D in the elasto-plasticity and the multi-scale cases), which poses a challenge for traditional basis functions. For example, if we were to use the linear basis functions, to discretize the high dimensional input space, even if we only have 10 grid points per dimension, the total degrees of freedom is 10^9, which is too costly regarding both the computation and the storage. Yet, neural networks are particularly convenient and useful for expressing mappings between high dimensional spaces [47] and requires no mesh in the input parameter domain.

All the aforementioned reasons motivate us to use neural networks in this work.

Optimization method. Most of neural networks in literature, especially in computer science communities, are trained with stochastic gradient methods (SGD). In our present work and the previous work [18], the optimization of the loss functions Equations (25) and (29) is done by the Limited-memory BFGS (L-BFGS-B) method [48] with the line search routine in [49], which attempts to enforce the Wolfe conditions [48] by a sequence of polynomial interpolations. Note BFGS is applicable in our case since the data sets are typically small and the neural network is reasonably deep and wide; otherwise, the memory requirement of L-BFGS-B is so high that SGD or other similar first-order methods for training neural networks should be adopted. It is worth mentioning that the choice of BFGS optimizer is well-motivated and has long been adopted for scientific and engineering applications due to its fast convergence and robustness.

Data scaling. We observed that input data scaling significantly helps train SPD-NNs faster, reduces overfitting, and makes better predictions. In the present work, the inputs and outputs of the neural network are scaled to a similar magnitude. Specifically, we introduce a strain reference ϵ_{ref} and a

stress reference σ_{ref} to scale strains and stresses. We also scale the tangent stiffness matrix by $\frac{\sigma_{\text{ref}}}{\epsilon_{\text{ref}}}$. For example, in the elasto-plasticity case, the Cholesky factor has the form

$$\mathsf{L}_\theta\left(\frac{\boldsymbol{\epsilon}^i}{\epsilon_{\text{ref}}}, \frac{\boldsymbol{\epsilon}^{i-1}}{\epsilon_{\text{ref}}}, \frac{\boldsymbol{\sigma}^{i-1}}{\sigma_{\text{ref}}}\right)$$

The most important thing is that $\frac{\sigma_{\text{ref}}}{\epsilon_{\text{ref}}} \sim O(E)$, here E is the estimated Young's modulus. This guarantees the inputs for SPD-NN, especially for elasto-plasticitiy, $(\boldsymbol{\epsilon}^i, \boldsymbol{\epsilon}^{i-1}, \boldsymbol{\sigma}^{i-1})$ have similar magnitudes.

Local minima. Local minima are observed in Section 4.1, since training neural networks involves highly non-convex optimization problems. Although neural networks with minimal losses on the training set from 10 different initial weights perform well, neural networks with median losses are less satisfactory. This reveals the uncertainty with respect to the initial weights for most NN-based data-driven approaches. Wider neural networks will be considered in the future, since some theoretical and computational results [50, 51] show quality of local minima tends to improve toward the global minimum value as depths and widths increase. As for the indirect data training approach, pre-training approach (see Section 3.2.2) produces acceptable initial weights and thus mitigates this concern. For all these training processes, the optimization is terminated when the objective function is called 3,000 times and 50,000 times for pre-training and training, respectively. The direct input-output data training is about 4 times faster than the indirect data training.

6. Conclusion

Data-driven approaches continue to gain popularity for constructing constitutive models from high-fidelity simulations and high-resolution experiments. The incorporation of data-driven constitutive models into conservation equations leads to a hybrid model, namely a coupled system with differential equations to describe conservation laws and thermodynamic principles, and neural networks to describe the material properties.

To make these hybrid models numerically more robust, we introduced a novel neural network architecture, SPD-NN, in which the neural network outputs the Cholesky factor of the tangent stiffness matrix instead of the stress or the stress increment. This neural network architecture weakly imposes convexity on the strain energy function (i.e., SPD tangent stiffness matrix). The incremental form of SPD-NN also preserves the time consistency and fulfills the second order work criterion. We tested SPD-NN-based constitutive relations on a 1D elasto-plastic truss problem and several 2D plate problems in which the plate is made of hyperelastic, elasto-plastic, and multiscale fiber-reinforced materials. When contrasting the SPD-NN with two other neural network architectures, we showed that SPD-NN exhibits better numerical stability in the resulting hybrid models. The general training approach and the improved numerical stability will allow extending SPD-NNs to other time-dependent physical systems, such as viscoelastic materials, where the constitutive relations are rate-dependent, and plastic materials with stronger hysteresis, where more history-dependent variables are required.

However, one limitation of the current approach is that the training process requires full-field data, either for strain-stress pairs or displacement measurements. This is especially challenging for a 3D solid body, where the measurements may only be made on the surface (although by using techniques such as X-ray computer tomograph [52], we can obtain full field displacements or strain fields induced by loading). NN-based constitutive modeling with incomplete data remains to be investigated in the future.

Acknowledgements

This work is supported by the Applied Mathematics Program within the Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR), through the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems Research Center (DE-SC0019453).

References

[1] J Ghaboussi, JH Garrett Jr, and Xiping Wu. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

[2] GW Ellis, C Yao, Rui Zhao, and Df Penumadu. Stress-strain modeling of sands using artificial neural networks. *Journal of geotechnical engineering*, 121(5):429–435, 1995.

[3] Yuelin Shen, K Chandrashekhara, WF Breig, and LR Oliver. Finite element analysis of v-ribbed belts using neural network based hyperelastic material model. *International Journal of Non-Linear Mechanics*, 40(6):875–890, 2005.

[4] Guanghui Liang and K Chandrashekhara. Neural network based constitutive model for elastomeric foams. *Engineering structures*, 30(7):2002–2011, 2008.

[5] Tomonari Furukawa and Genki Yagawa. Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, 43(2):195–219, 1998.

[6] Kun Wang and WaiChing Sun. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 334:337–380, 2018.

[7] M Mozaffar, R Bostanabad, W Chen, K Ehmann, J Cao, and MA Bessa. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420, 2019.

[8] F Ghavamian and A Simone. Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. *Computer Methods in Applied Mechanics and Engineering*, 357:112594, 2019.

[9] Panagiotis G Asteris and Vagelis Plevris. Anisotropic masonry failure criterion using artificial neural networks. *Neural Computing and Applications*, 28(8):2207–2229, 2017.

[10] Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, page 102447, 2019.

[11] Jakub Gajewski and Tomasz Sadowski. Sensitivity analysis of crack propagation in pavement bituminous layered structures using a hybrid system integrating artificial neural networks and finite element method. *Computational Materials Science*, 82:114–117, 2014.

[12] Xin Liu, Fei Tao, and Wenbin Yu. A neural network enhanced system for learning nonlinear constitutive relation of fiber reinforced composites. In *AIAA Scitech 2020 Forum*, page 0396, 2020.

[13] BA Le, Julien Yvonnet, and Q-C He. Computational homogenization of nonlinear elastic materials using neural networks. *International Journal for Numerical Methods in Engineering*, 104(12):1061–1084, 2015.

[14] MA Bessa, R Bostanabad, Z Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320:633–667, 2017.

[15] Chun-Teh Chen and Grace X Gu. Generative deep neural networks for inverse materials design using backpropagation and active learning. *Advanced Science*, page 1902607, 2020.

[16] Xiaoxuan Zhang and Krishna Garikipati. Machine learning materials physics: Multi-resolution neural networks learn the free energy and nonlinear elastic response of evolving microstructures. *arXiv preprint arXiv:2001.01575*, 2019.

[17] Michel Grediac, Fabrice Pierron, Stéphane Avril, and Evelyne Toussaint. The virtual fields method for extracting constitutive parameters from full-field measurements: a review. *Strain*, 42(4):233–253, 2006.

[18] Daniel Z Huang, Kailai Xu, Charbel Farhat, and Eric Darve. Predictive modeling with learned constitutive laws from indirect observations. *arXiv preprint arXiv:1905.12530*, 2019.

[19] Jie Yang, Rui Xu, Heng Hu, Qun Huang, and Wei Huang. Structural-genome-driven computing for composite structures. *Composite Structures*, 215:446–453, 2019.

[20] Jamshid Ghaboussi, David A Pecknold, Mingfu Zhang, and Rami M Haj-Ali. Autoprogressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering*, 42(1):105–126, 1998.

[21] Julia Ling, Reese Jones, and Jeremy Templeton. Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35, 2016.

[22] Zeliang Liu, CT Wu, and M Koishi. A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials. *Computer Methods in Applied Mechanics and Engineering*, 345:1138–1168, 2019.

[23] Zeliang Liu and CT Wu. Exploring the 3d architectures of deep material network in data-driven multiscale mechanics. *Journal of the Mechanics and Physics of Solids*, 127:20–46, 2019.

[24] Yousef Heider, Kun Wang, and WaiChing Sun. So(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials. *Computer Methods in Applied Mechanics and Engineering*, 345:1138–1168, 2020.

[25] Jintai Chung and GM1223971 Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method. *Journal of Applied Mechanics*, 60(2):371–375, 06 1993.

[26] Zdenek P Bazant, Ted B Belytschko, Ta-Peng Chang, et al. Continuum theory for strain-softening. *Journal of Engineering Mechanics*, 110(12):1666–1692, 1984.

[27] Rodney Hill. *The mathematical theory of plasticity*, volume 11. Oxford university press, 1998.

[28] Juan C Simo and Thomas JR Hughes. *Computational inelasticity*, volume 7. Springer Science & Business Media, 2006.

[29] François Nicot, Jean Lerbet, and Félix Darve. Second-order work criterion: from material point to boundary value problems. *Acta Mechanica*, 228(7):2483–2498, 2017.

[30] Zvi Hashin. Analysis of composite materials—a survey. *Journal of Applied Mechanics*, 50(3):481–505, 1983.

[31] CT Sun and RS Vaidya. Prediction of composite properties from a representative volume element. *Composites Science and Technology*, 56(2):171–179, 1996.

[32] Frédéric Feyel and Jean-Louis Chaboche. Fe2 multiscale approach for modelling the elasto-viscoplastic behaviour of long fibre sic/ti composite materials. *Computer methods in applied mechanics and engineering*, 183(3-4):309–330, 2000.

[33] T Kanit, S Forest, Ia Galliet, Va Mounoury, and D Jeulin. Determination of the size of the representative volume element for random composites: statistical and numerical approach. *International Journal of solids and structures*, 40(13-14):3647–3679, 2003.

[34] Zheng Yuan and Jacob Fish. Toward realization of computational homogenization in practice. *International Journal for Numerical Methods in Engineering*, 73(3):361–380, 2008.

[35] Yves Surrel. Moiré and grid methods: a signal-processing approach. In *Interferometry'94: photomechanics*, volume 2342, pages 118–128. International Society for Optics and Photonics, 1994.

[36] Stéphane Avril, Marc Bonnet, Anne-Sophie Bretelle, Michel Grédiac, François Hild, Patrick Ienny, Félix Latourte, Didier Lemosse, Stéphane Pagano, Emmanuel Pagnacco, et al. Overview of identification methods of mechanical parameters based on full-field measurements. *Experimental Mechanics*, 48(4):381, 2008.

[37] Giuseppe Geymonat, François Hild, and Stéphane Pagano. Identification of elastic parameters by displacement field measurement. *Comptes Rendus Mecanique*, 330(6):403–408, 2002.

[38] Xia-Ting Feng and Chengxiang Yang. Genetic evolution of nonlinear material constitutive models. *Computer Methods in Applied Mechanics and Engineering*, 190(45):5957–5973, 2001.

[39] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[40] René De Borst, Mike A Crisfield, Joris JC Remmers, and Clemens V Verhoosel. *Nonlinear finite element analysis of solids and structures*. John Wiley & Sons, 2012.

[41] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer series in statistics New York, 2nd edition, 2016.

[42] João Paulo Pascon. Large deformation analysis of plane-stress hyperelastic problems via triangular membrane finite elements. *International Journal of Advanced Structural Engineering*, 11(3):331–350, 2019.

[43] RS Rivlin. Rheology theory and applications. *ed. FR Eirich, Academic Books, London*, page 531, 1956.

[44] P Kanouté, DP Boso, JL Chaboche, and BA Schrefler. Multiscale methods for composites: a review. *Archives of Computational Methods in Engineering*, 16(1):31–75, 2009.

[45] Matthew J Zahr, Philip Avery, and Charbel Farhat. A multilevel projection-based model order reduction framework for nonlinear dynamic multiscale problems in structural and solid mechanics. *International Journal for Numerical Methods in Engineering*, 112(8):855–881, 2017.

[46] Zeliang Liu, MA Bessa, and Wing Kam Liu. Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials. *Computer Methods in Applied Mechanics and Engineering*, 306:319–341, 2016.

[47] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[48] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[49] Jorge J Moré and David J Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

[50] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pages 855–863, 2014.

[51] Kenji Kawaguchi, Jiaoyang Huang, and Leslie Pack Kaelbling. Effect of depth and width on local minima in deep learning. *Neural computation*, 31(7):1462–1498, 2019.

[52] Eric Maire and Philip John Withers. Quantitative x-ray tomography. *International materials reviews*, 59(1):1–43, 2014.