RECURRENT NEURAL NETWORK BASED MODEL DISCOVERY OF NONLINEAR VISCOELASTICITY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SAIM MASOOD

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

JULY 2024

Approval of the thesis:

RECURRENT NEURAL NETWORK BASED MODEL DISCOVERY OF NONLINEAR VISCOELASTICITY

submitted by **SAIM MASOOD** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Date:22.07.2024

Name, Surname: Saım Masood

Signature :

ABSTRACT

RECURRENT NEURAL NETWORK BASED MODEL DISCOVERY OF NONLINEAR VISCOELASTICITY

Masood, Saım
M.S., Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. Serdar Göktepe

July 2024, 83 pages

This thesis introduces a novel framework for automated model discovery in the context of nonlinear viscoelasticity. The framework leverages a recurrent neural network (RNN) model representing the stress update procedure that inherently satisfies necessary physical constraints. We trained the model on data comprising temporal sequences of applied deformation and resulting stresses. We use gradient-based optimization for the parameter identification, with gradients evaluated analytically via a recurrent derivative update algorithm.

The loss function comprises two terms: one quantifying the accuracy of the predictions and another inducing sparsity. The sparsity term encourages some learnable parameters to be zero, resulting in interpretable models with a few meaningful parameters. We demonstrate the ability of the framework to produce interpretable sparse models on synthetically generated data and experimental datasets of VHB 4910 and HNBR50 polymers. Further validation and applicability of the framework are illustrated through a finite element simulation using the model learned from the experimental dataset of HNBR50 polymer.

Keywords: Recurrent neural network, Model discovery, Nonlinear viscoelasticity

ÖZ

DOĞRUSAL OLMAYAN VİSKOELASTİSİTE İÇİN TEKRARLAYAN SİNİR AĞI TABANLI MODEL KEŞFİ

Masood, Saım
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Serdar Göktepe

Temmuz 2024 , 83 sayfa

Bu tez, doğrusal olmayan viskoelastisite bağlamında otomatik model keşfi için yenilikçi bir çerçeve sunmaktadır. Çerçeve, gerekli fiziksel kısıtlamaları doğal olarak karşılayan bir gerilme güncelleme prosedürünü temsil eden bir tekrarlayan sinir ağı (RNN) modelinden yararlanmaktadır. Model, uygulanan deformasyonların ve sonuçta ortaya çıkan gerilmelerin zaman serilerini içeren veriler üzerinde eğitilmiştir. Parametre tanımlama için gradyan tabanlı optimizasyon kullanılmakta, gradyanlar bir tekrarlayan türev güncelleme algoritması aracılığıyla analitik olarak değerlendirilmektedir.

Kayıp fonksiyonu iki terimden oluşmaktadır: Biri tahminlerin doğruluğunu ölçmekte, diğeri ise seyreklik sağlamaktadır. Seyreklik terimi, bazı öğrenilebilir parametrelerin sıfır olmasını teşvik ederek, az sayıda anlamlı parametreye sahip yorumlanabilir modellerin elde edilmesini sağlar. Çerçevenin, sentetik olarak üretilen veriler ve VHB 4910, HNBR50 polimerlerine ait deneysel veri kümeleri üzerinde yorumlanabilir seyrek modeller üretebilme yeteneğini gösterilmektedir. Çerçevenin daha fazla doğrulama ve uygulanabilirliği, HNBR50 polimerine ait deneysel veri kümesinden öğrenilen model kullanılarak yapılan bir sonlu eleman simülasyonu ile gösterilmek-tedir.

Anahtar Kelimeler: Tekrarlayan sinir ağı, Model keşfi, Doğrusal olmayan viskoelas-tisite

To my parents

ACKNOWLEDGMENTS

First, I would like to express my heartfelt gratitude to my advisor, Prof. Serdar Göktepe, for his guidance and mentorship over the past two years. His deep knowledge, dedication, and passion for computational mechanics have been truly inspirational. I am deeply grateful for his constant availability, attention to detail, and the time and effort he invested in my development.

I am also deeply indebted to Prof. Kağan Tuncay for his engaging lectures and invaluable advice and to Prof. Nejan Huvaj Sarıhan for her constant friendly support throughout my years at METU. In addition, I am thankful to Prof. Afşin Sarıtaş, Prof. Ahmet Hakan Argeşo, and Prof. Ercan Gürses for their evaluation and service on my thesis examining committee.

I am genuinely grateful for the companionship of my close friends: Ali Mahmood, Muhammad Suleman, Nael Haseeb, Saadat Raza, and Zaid Sabbah. Their understanding, encouragement, and friendship have been a constant source of strength, especially during challenging times. I would also like to thank my office mates, Erman Tunçer, Hossein Mazaheri, Tolga Tamer, and Sepehr Seyedian Choubi, for creating a friendly and pleasant working atmosphere.

Most importantly, I would like to express my most profound appreciation to my parents, Nuzhat Yasmeen and Masood Syed, for being exceptional role models and for their unwavering love and support. I am also deeply grateful to my sisters, Azka, Maryam, and Rabeea Masood, for their constant encouragement and belief in me.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

This thesis aims to develop a framework that converts experimental data of a material exhibiting finite viscoelastic behavior into a physically interpretable and numerically efficient constitutive model. We build a neural network architecture based on a custom recurrent unit to represent a generalized model of finite nonlinear viscoelasticity. For the parameter identification of the model, we supply a recurrent derivative update function alongside the stress update to analytically calculate the derivatives needed for the gradient-based optimization. To validate the effectiveness of the framework, we test the framework on three distinct datasets: one synthetically generated and two obtained from real-world experimental data.

In this chapter, we outline the motivation for this study and provide a concise literature review of related subjects. We introduce the theory of finite nonlinear viscoelasticity and examine existing data-driven methods for constitutive modeling. Furthermore, we describe the specific aim of this work and list the contributions. Lastly, we present the outline of the subsequent chapters.

1.1 Motivation

Materials that exhibit finite viscoelastic behavior, characterized by large deformation while displaying both viscous and elastic behavior, are crucial for many applications in various engineering fields. Polymers, a prevalent example, play a vital role in the production of everyday items like rubber bands and tires, as well as in the development of high-performance materials for aerospace and biomedical applications. Similarly, the finite viscoelastic properties of many biological tissues - including tendons, cartilage, and skin - are critical to their functional roles in the human body, making them a subject of great interest in medical research. Despite the importance of these materials, developing accurate and reliable constitutive models to predict their behavior remains a challenge. Existing literature offers a variety of finite-viscoelastic models to describe their deformation response. Broadly, these models fall under two approaches: purely phenomenological [1–8] and micromechanically-motivated [9–13].

Selecting an appropriate model or developing a new one for a given material requires expertise and careful consideration. Considering this requirement, this thesis develops a framework that automatically discovers a model of finite viscoelasticity from a material's experimental data. The framework generates a physically constrained constitutive model while accurately identifying a set of material parameters. By streamlining the traditionally complex processes of model formulation and calibration, this tool aims to alleviate a significant bottleneck faced by engineers who routinely work with new materials. This advancement significantly reduces the time and specialized knowledge required to develop functional and reliable models for materials exhibiting finite viscoelastic behavior, accelerating their implementation and evaluation in practical settings. Ultimately, this thesis is a contribution to the research aimed at enhancing the efficiency and precision with which new viscoelastic materials are integrated and utilized across various industries.

1.2 Finite Viscoelasticity

Finite viscoelasticity refers to the behavior of solids exhibiting both elastic and viscous characteristics when subject to large deformations. The word "finite" pertains to considering large deformations that require equations of nonlinear continuum mechanics for characterization. Elasticity is the property of materials that store and release energy without dissipation; a perfectly elastic material immediately returns to its original shape after removing stress. The stress calculation of an elastic response depends only on the deformation's current value without considering the path taken to reach that state of deformation or strain. On the other hand, the viscous part of a material's response is time- and rate-dependent—the viscous response results from internal energy dissipative mechanisms that evolve even after stopping external loading. A viscoelastic material's combined viscous and elastic response is commonly observed and explained through creep and relaxation. Creep refers to the time-dependent increase in strain observed when subjecting a body to constant stress. Relaxation is the decrease in stress over time to its underlying elastic response at constant strain, as shown in Figure 1.1.

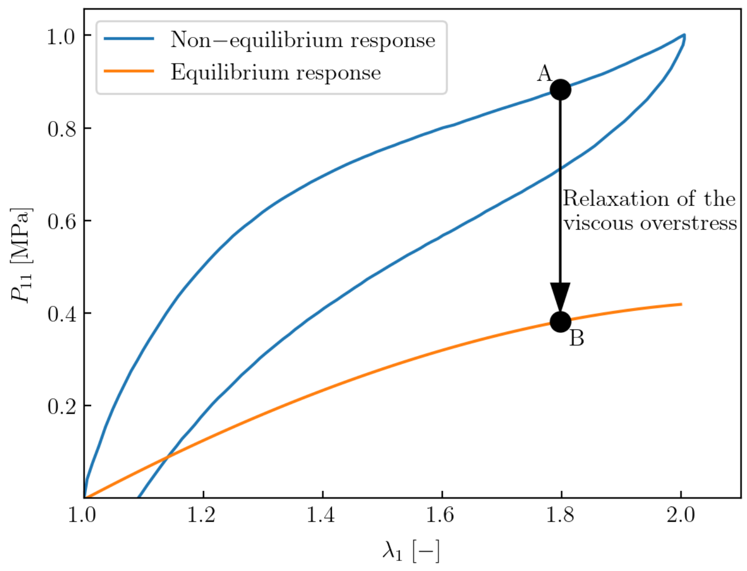


Figure 1.1: Uniaxial stress-stretch response of an HNBR50 polymer subjected to loading and unloading at a constant stretch rate of $|\dot{\lambda}| = 5\ (1/\mathrm{min})$ (Non-equilibrium response). The orange curve represents the underlying elastic response using data points obtained by relaxation periods during the cyclic loading in both the loading and unloading paths (Equilibrium response). The arrow represents the full relaxation of the viscous overstress from Point A to Point B.

From a macroscopic constitutive modeling viewpoint, one needs two equations to characterize viscoelastic behavior: one relating to the storage of energy in the material, known as the free energy function, and the other describing the dissipative viscous phenomena. Existing standard models account for the viscous phenomena arising from the internally evolving mechanisms by employing internal variables and supplying equations describing their evolution. The first category of formulations [3–5] defines stress-like internal variables by additively dividing the stress into an equilibrium part and several non-equilibrium components. Linear ordinary differential equations (ODEs) govern the evolution of the non-equilibrium overstresses, for which closed-form solutions exist in the form of convolution integrals. Although this approach is computationally efficient, it limits dissipation phenomena to small deviations from thermodynamic equilibrium, known as finite linear viscoelasticity. To account for large deviations, the second category [1, 2, 6, 7, 9, 10, 14] employs a multiplicative decomposition of the deformation gradient into elastic and inelastic parts. Quantities derived from the inelastic parts constitute the internal variables, and nonlinear ODEs generally define the evolution of these variables. We use the latter approach, employing the multiplicative decomposition, as it is more comprehensive in characterizing the so-called finite nonlinear viscoelasticity and is necessary to fit the data used in this study. We provide the details of the approach to model finite nonlinear viscoelasticity in Chapter 2.

1.3 Data Driven Approaches

The literature presents a range of models for both finite linear and nonlinear viscoelasticity. We can broadly divide these models into two categories. The first category comprises purely phenomenological models [1–8] that are based on empirical observations and designed to accurately describe macroscopic behavior without delving into micromechanics. The second category consists of micromechanically-motivated models [9–13] that are formulated based on the interactions and properties of the material's microstructure. Despite extensive research and the availability of numerous models, there is yet to be a consensus on a single best model for characterizing the behavior of any specific material. This lack of consensus poses a challenge for engineers when selecting an appropriate model and accurately identifying its parameters for novel materials. Researchers have turned to data-driven methods as a potential solution to this challenge, leading to developments in two fundamentally distinct research directions.

The first direction led to a new computing paradigm that eliminates the need for material modeling by directly utilizing raw data in simulations [15, 16]. These approaches, often called the "model-free" methods, solve boundary value problems by assigning a stress-strain pair from an experimental dataset to each material point that most closely satisfies the physical constraints and conservation laws. The underlying motive is to address solid mechanics by adhering to the definitive conservation laws while replacing the uncertain material laws with actual experimental data. This method efficiently bypasses the material (constitutive) modeling step in cases where the stress-strain relationship is bijective, such as in hyperelastic materials. However, this approach faces significant challenges when extended to behavior with non-unique, history-dependent relationships in inelasticity [17–19] like viscoelasticity, often requiring a large dataset to model arbitrary inelastic loading paths accurately.

In a different direction, researchers have utilized machine learning methods to develop generalized models. Flaschel et al. [20] introduced the EUCLID (Efficient Unsupervised Constitutive Law Identification and Discovery) framework, which relies solely on global force and displacement data to "discover" a hyperelastic model from a large catalog of functions through sparsity-inducing regularization while minimizing residual forces. Subsequent research extended the framework to include plasticity models [21] by incorporating a Fourier series representation of the yield function, and linear viscoelastic models [22] by employing a generalized Maxwell model represented by a Prony series to describe viscoelastic behavior. Additionally, Flaschel et al. [23] expanded the framework to handle materials with unknown constitutive behavior by identifying two scalar thermodynamic potentials: the Helmholtz free energy and the dissipation potential. However, to our knowledge, the EUCLID framework has yet to be extended to nonlinear viscoelasticity or incorporate finite deformations in the case of inelasticity.

Research has also explored using neural networks in constitutive modeling due to their ability to approximate any continuous function, as outlined in the Universal Approximation Theorem [24]. These networks are adept at capturing complex, nonlinear relationships in data, making them ideal for modeling intricate material behaviors. The early applications of neural networks in this field include the work by Ghaboussi et al. [25]. Recurrent Neural Networks (RNNs), known for handling temporal dependencies, are particularly suitable for modeling the history-dependent behavior of viscoelastic materials. For example, Ghavamian and Simone [26] used RNNs as surrogates for history-dependent micromechanical models to accelerate multiscale simulations. Gorji et al. [27] employed a Gated Recurrent Unit (GRU)-based RNN for modeling path-dependent plasticity, while Chen [28] used a Long-Short Term Memory (LSTM)-based RNN to capture the viscoelastic response of a Maxwell model. However, employing neural networks as black boxes in material modeling can violate kinematical, thermodynamical, and physical constraints when tested beyond their training regime [29, 30].

To integrate necessary constraints into deep-learning models, Raissi et al. [31] introduced Physics-Informed Neural Networks (PINNs) that integrate physical principles into the architecture by embedding these principles as penalty terms in the loss function. Following this innovation, He and Chen [32] and Danoun et al. [33] applied this methodology by including the violation of the second law of thermodynamics in the loss function to ensure thermodynamic consistency. Additionally, Haghighat et al. [34] utilized this approach by incorporating inequality constraints of elastoplasticity into the loss function, thus ensuring adherence to the fundamental laws of elastoplasticity, such as yield conditions and plastic flow directions. Further applying PINNs, Rezaei et al. [35] trained the networks to learn the solutions of constitutive equations, effectively bypassing the repetitive Newton iterations required for solving nonlinear material equations. However, this method "weakly" enforces constraints, which does not guarantee adherence to these constraints during inference on unseen data [30].

As an alternative, designing neural network architectures to satisfy physical constraints inherently from the outset can "strongly" enforce these constraints. Thakolkaran et al. [36] adapted the EUCLID framework to discover hyperelastic models by utilizing Input-Convex Neural Networks (ICNNs) specifically tailored to adhere to the physical laws of hyperelastic materials. Similarly, Rosenkranz et al. [37] and Asad and Farhat [38] used ICNNs, while Tac et al. [39] leveraged the so-called Neural Ordinary Differential Equations (NODEs) to define the potentials required for modeling nonlinear viscoelasticity, thus ensuring the network's compliance with essential constraints from the beginning. Moreover, Linka et al. [40, 41] developed Constitutive Artificial Neural Networks (CANNs). These networks leverage existing constitutive models to create generalized, invariant-based models that are intrinsically physics-informed, allowing the parameters learned to be physically interpretable. Researchers have extended the application of CANNs to include quasi-linear viscoelasticity [42], nonlinear finite viscoelasticity [43], and general inelastic behavior [44].

The works of Tac et al. [39], Holthusen et al. [44], and Abdolazizi et al. [43] build strongly physics-informed neural network architectures that are capable of model discovery of finite nonlinear viscoelasticity. Tac et al. [39] modeled anisotropic finite viscoelasticity by replacing the Helmholtz free energy function and the dissipation potential with NODEs, successfully leveraging the flexibility of NODEs to capture the complex viscoelastic behavior and achieving superior predictive accuracy compared to traditional closed-form models. However, the learned parameters lack physical interpretability, and the model is complex and computationally inefficient compared to conventional models. On the other hand, Holthusen et al. [44] developed inelastic CANNs (iCANNs) using feed-forward networks for the Helmholtz free energy and pseudo potential combined with a recurrent neural network setup, while Abdolazizi et al. [43] introduced viscoelastic CANNs (vCANNs) based on the generalized Maxwell model using a hereditary integral approach with the time-dependent kernel function influenced by the strain and strain rate to model nonlinear viscoelasticity. Both iCANNs and vCANNs feature physically interpretable parameters. However, iCANNs lack sparsity induction, and both rely on explicit update equations for automatic differentiation, enforcing a maximum time step size in the temporal datasets requiring comparatively finer data points to train.

1.4 Aim and Contributions

In light of the reviewed literature on data-driven approaches for constitutive modeling, this thesis aims to build a recurrent neural network-based framework for the model discovery of finite viscoelasticity. We highlight the key aspects and features of the framework below:

- We build the recurrent neural network architecture in a way that a priori satisfies necessary physical constraints.
- We specify equations to build a generalized constitutive model with physically interpretable parameters. Additionally, we suggest a novel polynomial-type evolution equation for the internal variables. The developed model is capable of characterizing finite nonlinear viscoelastic behavior.

- The model uses implicit update equations, allowing model training with scarce data. We supply a derivative update function to calculate the derivatives instead of using automatic differentiation.
- The framework uses sparsity-inducing regularization. Subsequently, the discovered model has a few non-zero parameters, which gives us insight into the physical nature of the material.

We train and validate the framework using three distinct datasets. We generate the first dataset synthetically, characterized by highly nonlinear viscoelastic behavior, and obtain the other two experimental datasets from published studies on the VHB 4910 [45] and HNBR50 polymer [11]. Notably, previous model discovery efforts have not addressed the dataset related to the HNBR50 polymer. Previous efforts to characterize HNBR50 material have relied exclusively on complex, micromechanically-motivated models [11,12,46,47]. Our framework effectively characterizes this material using a significantly simplified model approach.

1.5 Outline

The subsequent chapters are structured as follows:

Chapter 2 presents the theory of macroscopic finite viscoelasticity. Firstly, we present the decoupled volumetric-isochoric response and then the further decoupled isochoric equilibrium-overstress response. Secondly, we present details relating to the thermodynamical consistency and the form of the evolution law. Additionally, we give the algorithmic update procedure for the stress update. Finally, we specify constitutive equations of the generalized constitutive model we use in the framework.

In Chapter 3, we delineate the entire model discovery framework. We explain the custom recurrent unit we use in the RNN architecture. Furthermore, we develop the loss function along with the sparsity-inducing term. Finally, we provide details of the parameter identification algorithm, derivative calculation, and hyperparameter tuning.

In Chapter 4, we assess the framework's effectiveness across the three datasets and present the numerical results. We discuss each dataset's features and the insights gained regarding the material's physical properties from the results. We also implement the discovered model of the HNBR50 polymer in a finite element program and compare the results with actual experimental data.

Chapter 5 offers concluding remarks, addressing the framework's limitations and suggesting potential future research directions based on our findings.

CHAPTER 2

FINITE NONLINEAR VISCOELASTICITY

In this chapter, we present the fundamental geometric and balance equations that govern a displacement boundary value problem and outline the theory of macroscopic finite nonlinear viscoelasticity we utilize. We derive the general form for the Helmholtz free energy function and derive an expression for the evolution equation of the internal variables that satisfies thermodynamical consistency. We also provide the algorithmic update equations using the general form of the Helmholtz free energy function and the evolution equation. Finally, we propose a specific constitutive model consistent with the general constitutive equations for use in the framework.

2.1 Fundamental geometric and balance equations

We define the deformation map $\boldsymbol{\varphi} : \boldsymbol{X} \mapsto \boldsymbol{x}$ of a deformable body that maps points $\boldsymbol{X} \in \mathcal{B}$ in the reference configuration $\mathcal{B} \subset \mathbb{R}^3$ to points $\boldsymbol{x}(\boldsymbol{X}, t) \in \mathcal{S}$ in the spatial configuration $\mathcal{S} \subset \mathbb{R}^3$ at time $t \in \mathbb{R}_+$. Let $\boldsymbol{F} := \nabla \boldsymbol{\varphi}(\boldsymbol{X}; t)$ denote the deformation gradient that maps referential tangent vectors $\boldsymbol{T} \in T_X\mathcal{B}$ to spatial tangent vectors $\boldsymbol{t} = \boldsymbol{F}\boldsymbol{T} \in T_X\mathcal{S}$, and let the Jacobian $J := \det(\boldsymbol{F})$ denote the volume map that is restricted to positive real numbers $\mathbb{R}_+$ ensuring non-interpenetrability of the solid. Furthermore, $\boldsymbol{G} = G_{AB}$ and $\boldsymbol{g} = g_{ab}$ are the reference and spatial configurations metrics, respectively. These metrics are equal to Kronecker's deltas, $\boldsymbol{G} = \delta_{AB}$ and $\boldsymbol{g} = \delta_{ab}$, for Cartesian coordinate systems. The balance of linear momentum governs the boundary value problem of a deformable body

$$\rho_0 \ddot{\boldsymbol{\varphi}} = \text{Div}\left[\boldsymbol{\tau}\boldsymbol{F}^{-T}\right] + \bar{\gamma} \quad \text{in } \mathcal{B} \tag{2.1}$$

along with the displacement boundary conditions $\boldsymbol{\varphi} = \boldsymbol{\varphi}\left(\bar{\boldsymbol{X}};t\right)$ on $\partial\mathcal{B}_{\boldsymbol{\varphi}}$ and traction boundary conditions $\left[\boldsymbol{\tau}\boldsymbol{F}^{-T}\right]\mathrm{N} = \bar{\boldsymbol{t}}\left(\boldsymbol{X};t\right)$ on $\partial\mathcal{B}_{\boldsymbol{t}}$ with the referential outward unit normal N. Moreover, ρ_0 is the reference density, and $\bar{\gamma}$ is the body force for a unit volume in the reference configuration. For a viscoelastic body, we assume the Kirchhoff stress $\boldsymbol{\tau}$ to be a function of the deformation gradient $\boldsymbol{F}$ and internal variables $\boldsymbol{\mathcal{I}}$ to account for the viscous phenomena

$$\boldsymbol{\tau} = \hat{\boldsymbol{\tau}}(\boldsymbol{g}, \boldsymbol{\mathcal{I}}, \boldsymbol{F})\,. \tag{2.2}$$

2.2 Free energy function

Let Ψ denote the Helmholtz free energy function of a viscoelastic body. Staying consistent with Equation (2.2), we define Ψ to be a function of the deformation gradient $\boldsymbol{F}$ and internal variables $\boldsymbol{\mathcal{I}}$ as well

$$\Psi = \hat{\Psi}(\boldsymbol{g}, \boldsymbol{\mathcal{I}}, \boldsymbol{F})\,. \tag{2.3}$$

We assume the nearly incompressible behavior of the material and decouple the free energy function into volumetric and isochoric parts. The unimodular part of the deformation gradient

$$\bar{\boldsymbol{F}} := J^{-1/3}\boldsymbol{F} \tag{2.4}$$

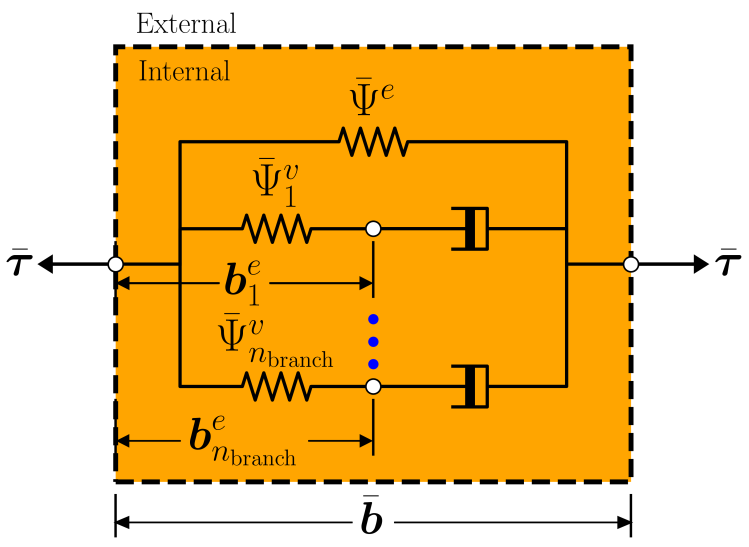
is assumed to govern the isochoric response. Additionally, we assume the viscous response to be purely isochoric. The Helmholtz free energy function takes the particular form

$$\Psi = U(J) + \hat{\Psi}^{\text{iso}}(\boldsymbol{g}, \boldsymbol{\mathcal{I}}; \bar{\boldsymbol{F}})\,, \tag{2.5}$$

where the potential U acts as a penalty function to enforce the assumed incompressibility.

Based on the idea of a generalized Maxwell model, we consider the further decomposition of the isochoric part into an elastic equilibrium branch and an arbitrary number of viscous branches, as shown in Figure 2.1. For any given viscous branch, we multiplicatively decompose the unimodular part of the deformation gradient into an elastic and a viscous part

$$\bar{\boldsymbol{F}} = \boldsymbol{F}^e_k \boldsymbol{F}^v_k\,, \tag{2.6}$$



External
Internal
$\bar{\Psi}^e$
$\bar{\Psi}^v_1$
$\bar{\boldsymbol{\tau}}$
$\bar{\boldsymbol{\tau}}$
$\boldsymbol{b}^e_1$
$\bar{\Psi}^v_{n_{\text{branch}}}$
$\boldsymbol{b}^e_{n_{\text{branch}}}$
$\bar{\boldsymbol{b}}$

Figure 2.1: Rheological representation of a generalized Maxwell model for the isochoric response of a viscoelastic body. The response is decomposed into a single elastic branch ($\bar{\Psi}^e$, $\bar{\boldsymbol{b}}$) and an arbitrary number n_{branch} of viscous branches ($\bar{\Psi}^v_k$, $\boldsymbol{b}^e_k$).

where the subscript k denotes the branch number. The isochoric part of the free energy function takes the form

$$\bar{\Psi}^{\text{iso}}(\boldsymbol{g}, \boldsymbol{F}^e_k, \bar{\boldsymbol{F}}) = \bar{\Psi}^e(\boldsymbol{g}, \bar{\boldsymbol{F}}) + \sum_{k=1}^{n_{\text{branch}}} \bar{\Psi}^v_k(\boldsymbol{g}, \boldsymbol{F}^e_k)\,, \tag{2.7}$$

where the elastic part of the deformation gradient $\boldsymbol{F}^e_k$ enters the formulation as the internal variable and is assumed to govern the viscous response of the k^{th} branch.

Furthermore, we assume the response to be isotropic and thus represent $\bar{\Psi}^e$ and $\bar{\Psi}^v_k$ as functions of the unimodular part of the left Cauchy-Green deformation tensor $\bar{\boldsymbol{b}} := \bar{\boldsymbol{F}}\boldsymbol{G}^{-1}\bar{\boldsymbol{F}}^T$ and the elastic left Cauchy-Green deformation tensor $\boldsymbol{b}^e_k := \boldsymbol{F}^e_k\tilde{\boldsymbol{G}}_k^{-1}\boldsymbol{F}^{eT}_k$, respectively, where $\tilde{\boldsymbol{G}}_k = \delta_{\tilde{A}\tilde{B}}$ is the metric of the intermediate configuration of the k^{th} branch coming from the multiplicative decomposition

$$\bar{\Psi}^{\text{iso}} = \bar{\Psi}^e(\boldsymbol{g}, \bar{\boldsymbol{b}}) + \sum_{k=1}^{n_{\text{branch}}} \bar{\Psi}^v_k(\boldsymbol{g}, \boldsymbol{b}^e_k)\,. \tag{2.8}$$

The principle of material frame invariance demands $\Psi(\boldsymbol{g}, \boldsymbol{\mathcal{I}}, \boldsymbol{F}) = \Psi(\boldsymbol{g}, \boldsymbol{\mathcal{I}}, \boldsymbol{QF})$ for all rotations $\boldsymbol{Q} \in SO(3)$. For free energy functions defined using $\bar{\boldsymbol{b}}$ and $\boldsymbol{b}^e_k$, this principle further reduces the above equation to be an isotropic function of the tensors.

To this end, we express $\bar{\Psi}^e$ and $\bar{\Psi}^v_k$ in terms of the first two invariants of $\bar{\boldsymbol{b}}$ and $\boldsymbol{b}^e_k$

$$\begin{aligned}\bar{\Psi}^{\text{iso}} &= \bar{\Psi}^e(\bar{I}_1, \bar{I}_2) + \sum_{k=1}^{n_{\text{branch}}} \bar{\Psi}^v_k(I^e_{1k}, I^e_{2k}) \qquad (2.9)\\ \text{with} \quad \bar{I}_1 &:= \operatorname{tr}(\bar{\boldsymbol{b}})\,,\\ \bar{I}_2 &:= \frac{1}{2}\big((\bar{I}_1)^2 - \operatorname{tr}(\bar{\boldsymbol{b}}^2)\big)\,,\\ I^e_{1k} &:= \operatorname{tr}(\boldsymbol{b}^e_k) \quad \text{and}\\ I^e_{2k} &:= \frac{1}{2}\big((I^e_{1k})^2 - \operatorname{tr}(\boldsymbol{b}^{e2}_k)\big)\,,\end{aligned}$$

where $\operatorname{tr}(\boldsymbol{A}) = \boldsymbol{A} : \boldsymbol{g}^{-1}$ represents the trace operation on tensor $\boldsymbol{A}$. The third invariant that describes the volume change is irrelevant as it is constant for the isochoric response.

2.3 Thermodynamical consistency

The second law of thermodynamics restricts the constitutive equations by the internal dissipation inequality for isothermal processes

$$\boldsymbol{S} : \frac{1}{2}\dot{\boldsymbol{C}} - \dot{\Psi} \geq 0\,, \qquad (2.10)$$

where $\boldsymbol{S}$ is the second Piola-Kirchhoff stress tensor and $\boldsymbol{C} := \boldsymbol{F}^T \boldsymbol{g} \boldsymbol{F}$ is the right Cauchy-Green deformation tensor. The invariants of $\bar{\boldsymbol{b}}$ and $\boldsymbol{b}^e_k$ are equal to the invariants of unimodular part of the right Cauchy-Green deformation tensor $\bar{\boldsymbol{C}} := \bar{\boldsymbol{F}}^T \boldsymbol{g} \bar{\boldsymbol{F}}$ and the elastic right Cauchy-Green deformation tensor $\boldsymbol{C}^e_k := \boldsymbol{F}^{eT}_k \boldsymbol{g} \boldsymbol{F}^e_k$, respectively. This allows us to define our free energy function in an alternative manner

$$\Psi = U(J) + \Psi^{\text{iso}} \quad \text{with} \quad \Psi^{\text{iso}} := \tilde{\Psi}^e(\bar{\boldsymbol{C}}) + \sum_{k=1}^{n_{\text{branch}}} \tilde{\Psi}^v_k(\boldsymbol{C}^e_k)\,. \qquad (2.11)$$

We can also obtain $\boldsymbol{C}^e_k$ by a push-forward operation on $\bar{\boldsymbol{C}}$ to the intermediate configuration

$$\boldsymbol{C}^e_k = \boldsymbol{F}^{v^{-T}} \bar{\boldsymbol{C}} \boldsymbol{F}^{v^{-1}}\,. \qquad (2.12)$$

Taking the material time derivative of Equation (2.11), and inserting it into Equation (2.10) gives us

$$\left[\boldsymbol{S} - 2\frac{\partial U}{\partial J}\frac{\partial J}{\partial \boldsymbol{C}} - \left(2\frac{\partial \tilde{\Psi}^e}{\partial \bar{\boldsymbol{C}}} + \sum_{k=1}^{n_{\text{branch}}} 2\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : \frac{\partial \boldsymbol{C}_k^e}{\partial \bar{\boldsymbol{C}}}\right) : \frac{\partial \bar{\boldsymbol{C}}}{\partial \boldsymbol{C}}\right] : \frac{1}{2}\dot{\boldsymbol{C}} - \sum_{k=1}^{n_{\text{branch}}} \frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : \frac{\partial \boldsymbol{C}_k^e}{\partial \boldsymbol{F}_k^v} : \dot{\boldsymbol{F}}_k^v \geq 0\,. \tag{2.13}$$

Using the argument of Coleman and Gurtin [48], the inequality in Equation (2.13) must be satisfied for an arbitrary value of $\dot{\boldsymbol{C}}$. This reasoning demands that the first term equal zero, giving us the following form for the second Piola-Kirchhoff stress

$$\boldsymbol{S} = \boldsymbol{S}^{\text{vol}} + \left(\bar{\boldsymbol{S}}^e + \sum_{k=1}^{n_{\text{branch}}} \bar{\boldsymbol{S}}_k^v\right) : \frac{\partial \bar{\boldsymbol{C}}}{\partial \boldsymbol{C}} \tag{2.14}$$

$$\begin{aligned}\text{with}\quad \boldsymbol{S}^{\text{vol}} &:= 2\frac{\partial U}{\partial J}\frac{\partial J}{\partial \boldsymbol{C}}\,,\\ \bar{\boldsymbol{S}}^e &:= 2\frac{\partial \tilde{\Psi}^e}{\partial \bar{\boldsymbol{C}}} \quad \text{and}\\ \bar{\boldsymbol{S}}_k^v &:= 2\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : \frac{\partial \boldsymbol{C}_k^e}{\partial \bar{\boldsymbol{C}}} = \boldsymbol{F}^{v^{-1}} 2\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} \boldsymbol{F}^{v^{-T}}\,.\end{aligned}$$

The dissipation inequality then reduces to

$$-\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : \frac{\partial \boldsymbol{C}_k^e}{\partial \boldsymbol{F}_k^v} : \dot{\boldsymbol{F}}_k^v \geq 0\,, \tag{2.15}$$

where we have dropped the summation over the branches as each branch's viscous behavior is independent, and the inequality must be satisfied for each branch separately, giving us a stronger constraint. Using Equation (2.12) in Equation (2.15) gives us

$$\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : \left(\boldsymbol{l}_k^{v^T} \boldsymbol{C}_k^e + \boldsymbol{C}_k^e \boldsymbol{l}_k^v\right) \geq 0\,, \tag{2.16}$$

where $\boldsymbol{l}_k^v := \dot{\boldsymbol{F}}_k^v \boldsymbol{F}^{v^{-1}}$ denotes the viscous part of the spatial velocity gradient of the k^{th} branch. Utilizing the symmetry of $\boldsymbol{C}_k^e$, we obtain

$$2\frac{\partial \tilde{\Psi}_k^v}{\partial \boldsymbol{C}_k^e} : (\boldsymbol{l}_k^v \boldsymbol{C}_k^e) \geq 0\,, \tag{2.17}$$

and further reduce it to

$$\left(\bar{\boldsymbol{\tau}}_k^v \boldsymbol{b}_k^{e^{-1}}\right) : \left(\boldsymbol{F}_k^e \boldsymbol{l}_k^v \boldsymbol{F}_k^{e^T}\right) \geq 0\,, \tag{2.18}$$

where we define $\bar{\boldsymbol{\tau}}_k^v := \bar{\boldsymbol{F}}\bar{\boldsymbol{S}}_k^v\bar{\boldsymbol{F}}^T$. The tensors $\bar{\boldsymbol{\tau}}_k^v$ and $\boldsymbol{b}_k^{e^{-1}}$ commute and are symmetric, which allows us to take the symmetric part of $\left(\boldsymbol{F}_k^e\boldsymbol{l}_k^v\boldsymbol{F}_k^{e^T}\right)$ and recast it as

$$\operatorname{sym}\left(\boldsymbol{F}_k^e\boldsymbol{l}_k^v\boldsymbol{F}_k^{e^T}\right) = -\frac{1}{2}\bar{\boldsymbol{F}}\overline{\boldsymbol{C}_k^{v^{-1}}}^{\boldsymbol{\cdot}}\bar{\boldsymbol{F}}^T = -\frac{1}{2}\boldsymbol{\pounds}_v\boldsymbol{b}_k^e\,. \tag{2.19}$$

Here, $\boldsymbol{\pounds}_v\boldsymbol{b}_k^e$ is the Lie-derivative of $\boldsymbol{b}_k^e$, and $\boldsymbol{C}_k^v$ denotes the viscous right Cauchy-Green deformation tensor of the k^{th} branch. Furthermore, $\boldsymbol{b}_k^e$ relates to $(\boldsymbol{C}_k^v)^{-1}$ through the push-forward operation

$$\boldsymbol{b}_k^e = \bar{\boldsymbol{F}}(\boldsymbol{C}_k^v)^{-1}\bar{\boldsymbol{F}}^T\,. \tag{2.20}$$

The dissipation inequality can now be written as

$$\bar{\boldsymbol{\tau}}_k^v : -\frac{1}{2}\left(\boldsymbol{\pounds}_v\boldsymbol{b}_k^e\right)\boldsymbol{b}_k^{e^{-1}} \geq 0\,. \tag{2.21}$$

We will use the following form of the evolution law proposed by Bergstrom and Boyce [9]

$$-\frac{1}{2}\left(\boldsymbol{\pounds}_v\boldsymbol{b}_k^e\right)\boldsymbol{b}_k^{e^{-1}} := \dot{\gamma}_k\mathbf{N}_k\,, \tag{2.22}$$

where

$$\mathbf{N}_k = \frac{\boldsymbol{\tau}_k^{v,\text{iso}}}{\|\boldsymbol{\tau}_k^{v,\text{iso}}\|} \tag{2.23}$$

$$\text{with}\quad \boldsymbol{\tau}_k^{v,\text{iso}} := \mathbb{P} : \bar{\boldsymbol{\tau}}_k^v \quad\text{and}$$

$$\|\boldsymbol{\tau}_k^{v,\text{iso}}\| := \sqrt{\boldsymbol{\tau}_k^{v,\text{iso}} : \boldsymbol{\tau}_k^{v,\text{iso}}}\,,$$

where $\mathbb{P}_{cd}^{ab} = [\delta_c^a\delta_d^b + \delta_d^a\delta_c^b]/2 - \delta^{ab}\delta_{cd}/3$ is the fourth-order deviatoric projection tensor. Moreover, $\dot{\gamma}_k$ denotes the effective creep rate and is a scalar quantity. We can insert Equation (2.22) into Equation (2.21) and utilize the deviatoric nature of the evolution law to replace $\bar{\boldsymbol{\tau}}_k^v$ with $\boldsymbol{\tau}_k^{v,\text{iso}}$ to obtain

$$\boldsymbol{\tau}_k^{v,\text{iso}} : \dot{\gamma}_k\mathbf{N}_k \geq 0\,. \tag{2.24}$$

We note that the thermodynamical consistency (2.21) is apriori satisfied for $\dot{\gamma}_k \geq 0$.

2.4 Algorithmic Update Equations

Here, we provide the algorithmic setting of the model for a finite element implementation. We will derive the stress and moduli terms using the general form of the Helmholtz free energy function and evolution equation developed in the previous section.

2.4.1 Decoupled volumetric-isochoric response

We use a push forward operation on Equation (2.14) of the second Piola-Kirchhoff stress tensor $\boldsymbol{S}$ to obtain the Kirchhoff stress tensor $\boldsymbol{\tau} = \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T$ in the following form

$$\boldsymbol{\tau} = p\boldsymbol{g}^{-1} + \mathbb{P} : \left(\bar{\boldsymbol{\tau}}^e + \sum_{k=1}^{n_{\text{branch}}} \bar{\boldsymbol{\tau}}_k^v \right) , \tag{2.25}$$

where the stress update equations for $\bar{\boldsymbol{\tau}}^e$ and $\bar{\boldsymbol{\tau}}_k^v$ will be derived in the subsequent sections. Furthermore, the moduli term can be additively decomposed into volumetric and isochoric parts.

$$\begin{aligned} &\mathbb{C} = \mathbb{C}^{\text{vol}} + \mathbb{C}^{\text{iso}} \\ &\text{with} \quad \mathbb{C}^{\text{vol}} := 4\partial^2_{\boldsymbol{gg}} U(J) \quad \text{and} \\ &\qquad\qquad \mathbb{C}^{\text{iso}} := 4\partial^2_{\boldsymbol{gg}} \hat{\Psi}^{\text{iso}} \left(\boldsymbol{g}; \bar{\boldsymbol{F}}\right) . \end{aligned} \tag{2.26}$$

For the volumetric part, we obtain

$$\mathbb{C}^{\text{vol}} = (p+s)\boldsymbol{g}^{-1} \otimes \boldsymbol{g}^{-1} - 2p\mathbb{I} , \tag{2.27}$$

where $\mathbb{I}^{abcd}_{\boldsymbol{g}^{-1}} = [\delta^{ac}\delta^{bd} + \delta^{ad}\delta^{bc}]/2$ is the fourth-order identity tensor. The volumetric stress p and modulus s in Equations (2.25) and (2.27) are defined as the following derivatives of the volumetric part of the free energy function $U(J)$

$$p := JU'(J) \quad \text{and} \quad s := J^2 U''(J) . \tag{2.28}$$

For the isochoric part of the moduli, we obtain

$$\begin{aligned} &\mathbb{C}^{\text{iso}} = \mathbb{P} : \left[\bar{\mathbb{C}} + \frac{2}{3}\left(\bar{\boldsymbol{\tau}} : \boldsymbol{g}\right)\mathbb{I} - \frac{2}{3}\left(\bar{\boldsymbol{\tau}} \otimes \boldsymbol{g}^{-1} + \boldsymbol{g}^{-1} \otimes \bar{\boldsymbol{\tau}}\right) \right] : \mathbb{P} \\ &\text{with} \quad \bar{\mathbb{C}} := \bar{\mathbb{C}}^e + \sum_{k=1}^{n_{\text{branch}}} \bar{\mathbb{C}}^v_{k,\text{algo}} , \end{aligned} \tag{2.29}$$

where $\bar{\mathbb{C}}^e$ and $\bar{\mathbb{C}}^v_{k,\text{algo}}$ will be derived in the subsequent sections.

2.4.2 Isochoric elastic response

For the elastic part of the isochoric response, we define

$$\bar{\boldsymbol{\tau}}^e := 2\partial_{\boldsymbol{g}} \bar{\Psi}^e(\bar{I}_1, \bar{I}_2) \quad \text{and} \quad \bar{\mathbb{C}}^e := 4\partial^2_{\boldsymbol{gg}} \bar{\Psi}^e(\bar{I}_1, \bar{I}_2) . \tag{2.30}$$

After taking the derivatives, we obtain the following for the stress and moduli

$$\bar{\boldsymbol{\tau}}^e = \bar{\Psi}_1^e \bar{\boldsymbol{b}} + \bar{\Psi}_2^e \left(\bar{I}_1 \bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2 \right) \quad \text{and} \tag{2.31}$$

$$\begin{aligned} \bar{\mathbb{C}}^e = & \left[\bar{\Psi}_{11}^e \bar{\boldsymbol{b}} + \bar{\Psi}_{21}^e \left(\bar{I}_1 \bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2 \right) \right] \otimes \bar{\boldsymbol{b}} + \\ & \left[\bar{\Psi}_{12}^e \bar{\boldsymbol{b}} + \bar{\Psi}_{22}^e \left(\bar{I}_1 \bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2 \right) \right] \otimes \left(\bar{I}_1 \bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2 \right) + \\ & 2\bar{\Psi}_2^e \left(\bar{\boldsymbol{b}} \otimes \bar{\boldsymbol{b}} - \mathbb{I}_{\bar{\boldsymbol{b}}} \right) , \end{aligned} \tag{2.32}$$

where we have defined the fourth order tensor $\mathbb{I}_{\bar{\boldsymbol{b}}}^{abcd} = [\bar{b}^{ac}\bar{b}^{bd} + \bar{b}^{ad}\bar{b}^{bc}]/2$ and the partial derivatives of $\bar{\Psi}^e$

$$\bar{\Psi}_i^e := 2\frac{\partial \bar{\Psi}^e}{\partial \bar{I}_i} \quad \text{and} \tag{2.33}$$

$$\bar{\Psi}_{ij}^e := 4\frac{\partial}{\partial \bar{I}_j} \left(\frac{\partial \bar{\Psi}^e}{\partial \bar{I}_i} \right) . \tag{2.34}$$

2.4.3 Isochoric viscous response

For the viscous part of the isochoric response of the kth branch, we define the stress

$$\bar{\boldsymbol{\tau}}_k^v := 2\partial_{\boldsymbol{g}} \bar{\Psi}_k^v (I_{1k}^e, I_{2k}^e) , \tag{2.35}$$

which results in

$$\bar{\boldsymbol{\tau}}_k^v = \bar{\Psi}_{k,1}^v \boldsymbol{b}_k^e + \bar{\Psi}_{k,2}^v \left(I_{1k}^e \boldsymbol{b}_k^e - \boldsymbol{b}_k^{e^2} \right) \tag{2.36}$$

$$\text{with} \quad \bar{\Psi}_{k,i}^v := 2\frac{\partial \bar{\Psi}_k^v}{\partial I_i^e} . \tag{2.37}$$

2.4.3.1 Integration of the evolution law

The evaluation of Equation (2.36) requires the value of the updated $\boldsymbol{b}_k^e$ at the current time step t_{n+1}. To obtain this value, we need to perform integration of the evolution law defined in Equation (2.22). The integration of the evolution law is based on the operator split of the material time derivative of $\boldsymbol{b}_k^e$ into an elastic predictor *E* and an inelastic corrector *I* as suggested by [6, 49, 50]

$$\dot{\overline{\boldsymbol{b}_k^e}} := \underbrace{\boldsymbol{l}^{\text{iso}} \boldsymbol{b}_k^e + \boldsymbol{b}_k^e \boldsymbol{l}^{\text{iso}^T}}_{E} + \underbrace{\pounds_v \boldsymbol{b}_k^e}_{I} , \tag{2.38}$$

where $\boldsymbol{l}^{\text{iso}} := \dot{\bar{\boldsymbol{F}}}\bar{\boldsymbol{F}}^{-1}$ is the unimodular part of the spatial velocity gradient.

In the elastic predictor step, the material time derivative of $(\boldsymbol{C}_k^v)^{-1}$ is set to zero, giving us the intermediate trial value of $\boldsymbol{b}_k^e$

$$(\boldsymbol{C}_k^{v,\text{tr}})^{-1} = (\boldsymbol{C}_k^v)_n^{-1} \qquad \Rightarrow \qquad \boldsymbol{b}_k^{e,\text{tr}} = \bar{\boldsymbol{F}}(\boldsymbol{C}_k^v)_n^{-1}\bar{\boldsymbol{F}}^T \tag{2.39}$$

The subscript denoting the current time step t_{n+1} is dropped for convenience. In the inelastic corrector step, the spatial velocity gradient is set to zero, which results in $\dot{\overline{\boldsymbol{b}_k^e}} = \boldsymbol{\pounds}_v \boldsymbol{b}_k^e$. Inputting the defined evolution law (2.22), we get the following

$$\dot{\overline{\boldsymbol{b}_k^e}} = \left[-2\dot{\gamma}_k \mathbf{N}_k\right] \boldsymbol{b}_k^e \,. \tag{2.40}$$

This equation is solved using the so-called exponential mapping, and the resultant integral is discretized to give us

$$\boldsymbol{b}_k^e \approx \exp\left[-2\Delta t \dot{\gamma}_k \mathbf{N}_k\right] \boldsymbol{b}_k^{e,\text{tr}} \,. \tag{2.41}$$

For our assumed isotropic case, $\boldsymbol{\tau}_k^{v,\text{iso}}$ and consequently $\mathbf{N}_k$ commute with both $\boldsymbol{b}_k^e$ and $\boldsymbol{b}_k^{e,\text{tr}}$. This allows us to write (2.41) in the principal stretch directions

$$\Lambda_{k,\alpha}^e \approx \exp\left[-\frac{\sqrt{2}\Delta t}{\tau_k^v}\dot{\gamma}_k \tau'_{k,\alpha}\right] \Lambda_{k,\alpha}^{e,\text{tr}} \,, \tag{2.42}$$

where $\tau_k^v := \|\boldsymbol{\tau}_k^{v,\text{iso}}\|/\sqrt{2}$. Here, we make use of the following spectral decompositions

$$\boldsymbol{F}^e = \sum_{\alpha=1}^{3} \lambda_{k,\alpha}^e \, \boldsymbol{n}_\alpha^k \otimes \boldsymbol{N}_\alpha^k \,, \quad \boldsymbol{F}^{e,\text{tr}} = \sum_{\alpha=1}^{3} \lambda_{k,\alpha}^{e,\text{tr}} \, \boldsymbol{n}_\alpha^k \otimes \boldsymbol{N}_\alpha^k \,, \tag{2.43}$$

$$\boldsymbol{b}_k^e = \sum_{\alpha=1}^{3} \Lambda_{k,\alpha}^e \, \boldsymbol{n}_\alpha^k \otimes \boldsymbol{n}_\alpha^k \,, \quad \boldsymbol{b}_k^{e,\text{tr}} = \sum_{\alpha=1}^{3} \Lambda_{k,\alpha}^{e,\text{tr}} \, \boldsymbol{n}_\alpha^k \otimes \boldsymbol{n}_\alpha^k \quad \text{and} \tag{2.44}$$

$$\boldsymbol{\tau}_k^{v,\text{iso}} = \sum_{\alpha=1}^{3} \tau'_{k,\alpha} \, \boldsymbol{n}_\alpha^k \otimes \boldsymbol{n}_\alpha^k \,, \tag{2.45}$$

where $\Lambda_{k,\alpha}^e := (\lambda_{k,\alpha}^e)^2$ and $\Lambda_{k,\alpha}^{e,\text{tr}} := (\lambda_{k,\alpha}^{e,\text{tr}})^2$. Taking the natural logarithm of both sides of Equation (2.42) gives us

$$\varepsilon_{k,\alpha} \approx -\frac{\Delta t}{\sqrt{2}}\frac{\dot{\gamma}_k}{\tau_k^v}\tau'_{k,\alpha} + \varepsilon_{k,\alpha}^{\text{tr}} \tag{2.46}$$

$$\text{with} \quad \varepsilon_{k,\alpha} := \ln \lambda_{k,\alpha}^e \quad \text{and}$$

$$\varepsilon_{k,\alpha}^{\text{tr}} := \ln \lambda_{k,\alpha}^{e,\text{tr}} \,.$$

The nonlinear Equation (2.46) can be solved using the Newton-Raphson iterative scheme. To solve the nonlinear equation, we first define the residual as a function of the vector of the elastic principal logarithmic stretches $\boldsymbol{\varepsilon}_k := [\varepsilon_{k,1}\ \varepsilon_{k,2}\ \varepsilon_{k,3}]^T$

$$\boldsymbol{r}_k(\boldsymbol{\varepsilon}_k) := \boldsymbol{\varepsilon}_k + \frac{\Delta t}{\sqrt{2}}\frac{\dot{\gamma}_k}{\tau_k^v}\boldsymbol{\tau}'_k - \boldsymbol{\varepsilon}_k^{\mathrm{tr}} = \mathbf{0}\,, \tag{2.47}$$

where $\boldsymbol{\tau}'_k := \left[\tau'_{k,1}\ \tau'_{k,2}\ \tau'_{k,3}\right]^T$ is a vector of the principal viscous isochoric stresses. We linearize the residual in the neighborhood of the value of $\boldsymbol{\varepsilon}_k$ at the m^{th} iteration

$$\mathrm{Lin}\,\boldsymbol{r}_k(\boldsymbol{\varepsilon}_k)\big|_{\boldsymbol{\varepsilon}_{k,m}} = \boldsymbol{r}_k(\boldsymbol{\varepsilon}_{k,m}) + \boldsymbol{\mathcal{K}}_k\big|_{\boldsymbol{\varepsilon}_{k,m}}\Delta\boldsymbol{\varepsilon}_k = \mathbf{0}\,, \tag{2.48}$$

where we define the local tangent of the Newton iteration as

$$\boldsymbol{\mathcal{K}}_k := \frac{\partial \boldsymbol{r}_k}{\partial \boldsymbol{\varepsilon}_k}\,. \tag{2.49}$$

The solution of the linearized residual (2.48) gives us the local Newton update step at the m^{th} iteration

$$\boldsymbol{\varepsilon}_{k,m+1} \leftarrow \boldsymbol{\varepsilon}_{k,m} - \boldsymbol{\mathcal{K}}_k^{-1}\big|_{\boldsymbol{\varepsilon}_{k,m}}\boldsymbol{r}_k(\boldsymbol{\varepsilon}_{k,m})\,. \tag{2.50}$$

We compute the local tangent $\boldsymbol{\mathcal{K}}_k$ as follows

$$(\mathcal{K}_k)_{\alpha\beta} = \delta_{\alpha\beta} + \frac{\Delta t}{\sqrt{2}}\frac{\tau'_{k,\alpha}}{\tau_k^v}\frac{\partial\dot{\gamma}_k}{\partial\varepsilon_{k,\beta}} + \frac{\Delta t}{\sqrt{2}}\frac{\dot{\gamma}_k}{\tau_k^v}\frac{\partial\tau'_{k,\alpha}}{\partial\varepsilon_{k,\beta}} + \frac{\Delta t}{\sqrt{2}}\dot{\gamma}_k\tau'_{k,\alpha}\frac{\partial\tau_k^{v^{-1}}}{\partial\varepsilon_{k,\beta}}\,. \tag{2.51}$$

After taking the required derivatives, we get

$$(\mathcal{K}_k)_{\alpha\beta} = \delta_{\alpha\beta} + \frac{\Delta t}{\sqrt{2}}\frac{\tau'_{k,\alpha}}{\tau_k^v}\frac{\partial\dot{\gamma}_k}{\partial\varepsilon_{k,\beta}} + \frac{\Delta t}{\sqrt{2}}\frac{\dot{\gamma}_k}{\tau_k^v}\left(\bar{\mathcal{T}}_k\right)_{\alpha\beta} - \frac{\Delta t}{2\sqrt{2}}\frac{\dot{\gamma}_k}{\tau_k^{v3}}\tau'_{k,\alpha}(\mathcal{D}_k)_\beta \tag{2.52}$$

$$\text{with}\quad \left(\bar{\mathcal{T}}_k\right)_{\alpha\beta} := (\mathcal{T}_k)_{\alpha\beta} - \frac{1}{3}\sum_{\eta=1}^{3}(\mathcal{T}_k)_{\eta\beta}\,,$$

$$(\mathcal{D}_k)_\beta := \sum_{\eta=1}^{3}\tau'_{k,\eta}(\mathcal{T}_k)_{\eta\beta} \quad \text{and}$$

$$(\mathcal{T}_k)_{\alpha\beta} := \frac{\partial\tau_{k,\alpha}}{\partial\varepsilon_{k,\beta}} \quad \text{with} \quad \bar{\boldsymbol{\tau}}_k^v = \sum_{a=1}^{3}\tau_{k,\alpha}\,\boldsymbol{n}_\alpha^k\otimes\boldsymbol{n}_\alpha^k\,.$$

The solution of the nonlinear Equation (2.46) using a Newton-Raphson iterative scheme is summarized in Algorithm 1. Once we get $\boldsymbol{b}_k^e$ and $\bar{\boldsymbol{\tau}}_k^v$, we perform a pull back operation on $\boldsymbol{b}_k^e$ to update and obtain the value of $(\boldsymbol{C}_k^v)^{-1}$ for the next time step

$$(\boldsymbol{C}_k^v)^{-1} = \bar{\boldsymbol{F}}^{-1}\boldsymbol{b}_k^e\bar{\boldsymbol{F}}^{-T}\,. \tag{2.53}$$

Algorithm 1: Newton-Raphson algorithm to solve for $\boldsymbol{b}^e_k$ and $\bar{\boldsymbol{\tau}}^v_k$

```
for $k = 1$ to $n_{\text{branch}}$ do
    Given:
        $\boldsymbol{b}^{e,\text{tr}}_k = \sum_{\alpha=1}^{3} \Lambda^{e,\text{tr}}_{k,\alpha}\, \boldsymbol{n}^k_\alpha \otimes \boldsymbol{n}^k_\alpha \quad , \quad \varepsilon^{\text{tr}}_{k,\alpha} = \ln(\Lambda^{e,\text{tr}}_{k,\alpha})/2$
    Initialize:
        $m = 0 \quad , \quad \boldsymbol{\varepsilon}_{k,0} = \boldsymbol{\varepsilon}^{\text{tr}}_k \quad , \quad \boldsymbol{r}_k = 1 \times 10^8$
    while $TOL \leq |\boldsymbol{r}^e_k|$ do
        Calculate required terms:
            $\varepsilon_{k,\alpha} = (\boldsymbol{\varepsilon}_{k,m})_\alpha$
            $\boldsymbol{b}^e_k = \sum_{\alpha=1}^{3} \Lambda^e_{k,\alpha}\, \boldsymbol{n}^k_\alpha \otimes \boldsymbol{n}^k_\alpha \quad , \quad \Lambda^e_{k,\alpha} = \exp(2\varepsilon_{k,\alpha})$
            $\bar{\boldsymbol{\tau}}^v_k = \bar{\Psi}^v_{k,1}\boldsymbol{b}^e_k + \bar{\Psi}^v_{k,2}\left(I^v_{1k}\boldsymbol{b}^e_k - \boldsymbol{b}^{e^2}_k\right)$
            $\boldsymbol{\tau}^{v,\text{iso}}_k = \mathbb{P} : \bar{\boldsymbol{\tau}}^v_k \quad , \quad \boldsymbol{\tau}^{v,\text{iso}}_k = \sum_{\alpha=1}^{3} \tau'_{k,\alpha}\, \boldsymbol{n}^k_\alpha \otimes \boldsymbol{n}^k_\alpha$
            $\tau^v_k = \|\boldsymbol{\tau}^{v,\text{iso}}_k\|/\sqrt{2}$
            $\dot{\gamma}_k$ can be a function of $\boldsymbol{b}^e_k$ and $\boldsymbol{\tau}^{v,\text{iso}}_k$; a specific form for $\dot{\gamma}_k$ is
              proposed in Equation (2.72).
        Calculate residual and local tangent:
            $\boldsymbol{r}_k(\boldsymbol{\varepsilon}_{k,m}) = \boldsymbol{\varepsilon}_{k,m} + \frac{\Delta t}{\sqrt{2}}\frac{\dot{\gamma}_k}{\tau^v_k}\boldsymbol{\tau}'_k - \boldsymbol{\varepsilon}^{\text{tr}}_k$
            $\boldsymbol{\mathcal{K}}_k = \left.\frac{\partial \boldsymbol{r}_k}{\partial \boldsymbol{\varepsilon}_k}\right|_{\boldsymbol{\varepsilon}_{k,m}}$ using Equation (2.52); or using Equation (2.83)
              for the specific model.
        Update:
            $\boldsymbol{\varepsilon}_{k,m+1} \leftarrow \boldsymbol{\varepsilon}_{k,m} - \boldsymbol{\mathcal{K}}^{-1}_k\big|_{\boldsymbol{\varepsilon}_{k,m}} \boldsymbol{r}_k(\boldsymbol{\varepsilon}_{k,m})$
            $m \leftarrow m + 1$
    end
    Output:
        $\boldsymbol{b}^e_k$ and $\bar{\boldsymbol{\tau}}^v_k$
end
```

2.4.3.2 Algorithmic moduli

We can use an alternative multiplicative decomposition of $\bar{\boldsymbol{F}}$ using $\boldsymbol{F}^{e,\text{tr}}_k = \bar{\boldsymbol{F}}\,(\boldsymbol{F}^v_k)^{-1}_n$

$$\bar{\boldsymbol{F}} = \boldsymbol{F}^{e,\text{tr}}_k\,(\boldsymbol{F}^v_k)_n \;, \tag{2.54}$$

where $(\boldsymbol{F}^v_k)_n$ is the viscous part of the deformation gradient of the k^{th} branch from the previous time step t_n. Because $(\boldsymbol{F}^v_k)_n$ is a constant, we can define the change in $\bar{\boldsymbol{F}}$ as

$$\Delta\bar{\boldsymbol{F}} = \Delta\boldsymbol{F}^{e,\text{tr}}_k\,(\boldsymbol{F}^v_k)_n\ . \tag{2.55}$$

This allows us to define the moduli $\tilde{\mathbb{C}}^v_{k,\text{algo}}$ in the trial intermediate configuration as follows

$$\Delta\tilde{\boldsymbol{S}}^v_k = \tilde{\mathbb{C}}^v_{k,\text{algo}} : \frac{1}{2}\Delta\boldsymbol{C}^{e,\text{tr}}_k \quad \text{and} \quad \tilde{\mathbb{C}}^v_{k,\text{algo}} := 2\partial_{\boldsymbol{C}^{e,\text{tr}}_k}\tilde{\boldsymbol{S}}^v_k\,, \tag{2.56}$$

where we have defined a stress measure $\tilde{\boldsymbol{S}}^v_k$ in the intermediate configuration coming from the alternative multiplicative decomposition (2.54)

$$\tilde{\boldsymbol{S}}^v_k := \boldsymbol{F}^{e,\text{tr}^{-1}}_k\bar{\boldsymbol{\tau}}^v_k\boldsymbol{F}^{e,\text{tr}^{-T}}_k\ . \tag{2.57}$$

We can write $\tilde{\boldsymbol{S}}^v_k$ in the principal stretch directions

$$\begin{aligned}\tilde{\boldsymbol{S}}^v_k &= \sum_{\alpha=1}^{3}\tilde{S}_{k,\alpha}\boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha \\ &= \sum_{\alpha=1}^{3}\frac{\tau_{k,\alpha}}{\Lambda^{e,\text{tr}}_{k,\alpha}}\boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha\,,\end{aligned} \tag{2.58}$$

and take its derivative with respect to $\boldsymbol{C}^{e,\text{tr}}_k$ to obtain

$$\begin{aligned}\tilde{\mathbb{C}}^v_{k,\text{algo}} = &\sum_{\alpha=1}^{3}\sum_{\beta=1}^{3}\left(\frac{1}{\Lambda^{e,\text{tr}}_{k,\alpha}\Lambda^{e,\text{tr}}_{k,\beta}}\frac{\partial\tau_{k,\alpha}}{\partial\varepsilon^{\text{tr}}_{k,\beta}} - \frac{2\tau_{k,\alpha}}{\Lambda^{e,\text{tr}}_{k,\alpha}\Lambda^{e,\text{tr}}_{k,\beta}}\delta_{\alpha\beta}\right)\boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha\otimes\partial_{\boldsymbol{C}^{e,\text{tr}}_k}\Lambda^{e,\text{tr}}_{k,\beta} \\ &+\sum_{\alpha=1}^{3}2\frac{\tau_{k,\alpha}}{\Lambda^{e,\text{tr}}_{k,\alpha}}\partial_{\boldsymbol{C}^{e,\text{tr}}_k}\left(\boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha\right)\ .\end{aligned} \tag{2.59}$$

We take the partial derivatives of the eigenvalues

$$\partial_{\boldsymbol{C}^{e,\text{tr}}_k}\Lambda^{e,\text{tr}}_{k,\alpha} = \boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha \tag{2.60}$$

and eigenvectors

$$\partial_{\boldsymbol{C}^{e,\text{tr}}_k}\left(\boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha\right) = \sum_{\beta\neq\alpha}^{3}\frac{1}{2}\frac{1}{\Lambda^{e,\text{tr}}_{k,\beta}-\Lambda^{e,\text{tr}}_{k,\alpha}}\left(\mathbb{G}^k_{\alpha\beta}+\mathbb{G}^k_{\beta\alpha}\right) \tag{2.61}$$

$$\text{with}\quad \left(\mathbb{G}^k_{\alpha\beta}\right)^{ABCD} := \left(M^k_\alpha\right)^{AC}\left(M^k_\beta\right)^{BD} + \left(M^k_\alpha\right)^{AD}\left(M^k_\beta\right)^{BC}$$

$$\boldsymbol{M}^k_\alpha := \boldsymbol{N}^k_\alpha\otimes\boldsymbol{N}^k_\alpha \quad , \quad \left(M^k_\alpha\right)^{AB} := \left(N^k_\alpha\right)^{A}\left(N^k_\alpha\right)^{B}\ .$$

We also need

$$\frac{\partial \tau_{k,\alpha}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} = \frac{\partial \tau_{k,\alpha}}{\partial \varepsilon_{k,\eta}} \frac{\partial \varepsilon_{k,\eta}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} = (\mathcal{T}_k)_{\alpha\eta} \frac{\partial \varepsilon_{k,\eta}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} \tag{2.62}$$

for which we use the consistency condition, which requires the total derivative of the residual expression (2.47) with respect to the trial principal logarithmic stretches to equal zero

$$\begin{aligned} \frac{dr_{k,\alpha}}{d\varepsilon^{\text{tr}}_{k,\beta}} &= \frac{\partial r_{k,\alpha}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} + \frac{\partial r_{k,\alpha}}{\partial \varepsilon_{k,\eta}} \frac{\partial \varepsilon_{k,\eta}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} = 0 \\ &= \delta_{\alpha\beta} + (\mathcal{K}_k)_{\alpha\eta} \frac{\partial \varepsilon_{k,\eta}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} = 0 \,. \end{aligned} \tag{2.63}$$

Using (2.63), we obtain

$$\frac{\partial \varepsilon_{k,\alpha}}{\partial \varepsilon^{\text{tr}}_{k,\beta}} = (\mathcal{K}_k)^{-1}_{\alpha\beta} \,. \tag{2.64}$$

We reach the following final form for $\tilde{\mathbb{C}}^v_{k,\text{algo}}$ in the intermediate configuration

$$\begin{aligned} \tilde{\mathbb{C}}^v_{k,\text{algo}} = & \sum_{\alpha=1}^{3} \sum_{\beta=1}^{3} \left(\frac{(\mathcal{T}_k)_{\alpha\eta} (\mathcal{K}_k)^{-1}_{\eta\beta} - 2\tau_{k,\alpha}\delta_{\alpha\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha} \Lambda^{e,\text{tr}}_{k,\beta}} \right) \boldsymbol{M}^k_\alpha \otimes \boldsymbol{M}^k_\beta \\ & + \frac{1}{2} \sum_{\alpha=1}^{3} \sum_{\beta \neq \alpha}^{3} \frac{\tilde{S}_{k,\alpha} - \tilde{S}_{k,\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha} - \Lambda^{e,\text{tr}}_{k,\beta}} \left(\mathbb{G}^k_{\alpha\beta} + \mathbb{G}^k_{\beta\alpha} \right) \,. \end{aligned} \tag{2.65}$$

One notes that there is a singularity in Equation (2.65) when two eigenvalues are equal

$$\lim_{\Lambda^{e,\text{tr}}_{k,\alpha} \to \Lambda^{e,\text{tr}}_{k,\beta}} \frac{\tilde{S}_{k,\alpha} - \tilde{S}_{k,\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha} - \Lambda^{e,\text{tr}}_{k,\beta}} = \frac{0}{0} \,. \tag{2.66}$$

We can use L'Hopital's rule and take the derivative of both the numerator and the denominator with respect to $\Lambda^{e,\text{tr}}_{k,\alpha}$ to get the following for equal eigenvalues

$$\lim_{\Lambda^{e,\text{tr}}_{k,\alpha} \to \Lambda^{e,\text{tr}}_{k,\beta}} \frac{\tilde{S}_{k,\alpha} - \tilde{S}_{k,\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha} - \Lambda^{e,\text{tr}}_{k,\beta}} = \frac{1}{2} \frac{(\mathcal{T}_k)_{\alpha\eta} (\mathcal{K}_k)^{-1}_{\eta\alpha} - 2\tau_{k,\alpha}}{\Lambda^{e,\text{tr}^2}_{k,\alpha}} \,. \tag{2.67}$$

Finally, we push forward $\bar{\mathbb{C}}^v_{k,\text{algo}}$ from the intermediate configuration to the spatial configuration using

$$\left(\bar{\mathbb{C}}^v_{k,\text{algo}}\right)^{abcd} = \left(F^{e,tr}_k\right)^a_A \left(F^{e,tr}_k\right)^b_B \left(F^{e,tr}_k\right)^c_C \left(F^{e,tr}_k\right)^d_D \left(\tilde{\mathbb{C}}^v_{k,\text{algo}}\right)^{ABCD} \,. \tag{2.68}$$

2.5 Specific constitutive model

In this section, we propose a specific constitutive model to be used in this work that is consistent with the general constitutive equations described for finite viscoelasticity above.

2.5.1 Helmholtz free energy function and effective creep rate

For the Helmholtz free energy functions, we use

$$U(J) = \frac{\kappa}{4}\left(J^2 - 1 - 2\ln J\right) , \tag{2.69}$$

$$\bar{\Psi}^e(\bar{I}_1, \bar{I}_2) = c_1(\bar{I}_1 - 3) + c_2(\bar{I}_2 - 3) \quad \text{and} \tag{2.70}$$

$$\bar{\Psi}^v_k(I^e_{1k}, I^e_{2k}) = c^v_{1k}(I^e_{1k} - 3) + c^v_{2k}(I^e_{2k} - 3) , \tag{2.71}$$

where U is taken as the default volumetric function from FEAP (Finite Element Analysis Program) [51] that we used for the finite element implementation in Section 4.3.4. The material parameter κ is taken large enough to enforce incompressibility. The functions $\bar{\Psi}^e$ and $\bar{\Psi}^v_k$ are inspired by the Mooney-Rivlin [52,53] hyperelastic material model. In the above equations, $c_1, c_2, c^v_{1k}, c^v_{2k}$ are the identifiable material parameters. We suggest the use of five Maxwell viscous branches ($n_{branch} = 5$) in the subsequent sections to allow for a model with a high enough capacity to fit the data. We propose a polynomial-type equation for the required effective creep rate of each branch

$$\dot{\gamma}_k = \sum_{q=1}^{n_{\text{pow}}} a_{k,q}\left(\frac{\tau^v_k}{\hat{\tau}_k}\right)^q , \tag{2.72}$$

where $a_{k,q}$ and $\hat{\tau}_k$ are the identifiable material parameters in the above equation. We use five power terms ($n_{\text{pow}} = 5$) for the rest of the paper. With c^v_{1k}, c^v_{2k}, $\hat{\tau}_k$ and the coefficients $a_{k,q}$ of the five power terms, we have eight learnable parameters for each branch. For the five viscous branches, in addition to the two learnable parameters of the elastic branch, the total number of learnable parameters equals 42. The aim is to represent this model using the recurrent neural network architecture described in the forthcoming chapter and identify a select number of important material parameters.

2.5.2 Algorithmic setting of the constitutive model

The volumetric response requires the values of the volumetric stress p and modulus s which we calculate as

$$p = \frac{\kappa}{2}(J^2 - 1) \quad \text{and} \quad s = \frac{\kappa}{2}(J^2 + 1) . \tag{2.73}$$

For the elastic part of the isochoric response, we only need to take the following first and second order partial derivatives of $\bar{\Psi}^e$ with respect to the invariants

$$\frac{\partial \bar{\Psi}^e}{\partial \bar{I}_1} = c_1 \, , \tag{2.74}$$

$$\frac{\partial \bar{\Psi}^e}{\partial \bar{I}_2} = c_2 \quad \text{and} \tag{2.75}$$

$$\frac{\partial^2 \bar{\Psi}^e}{\partial \bar{I}_1^2} = \frac{\partial^2 \bar{\Psi}^e}{\partial \bar{I}_2 \, \partial \bar{I}_1} = \frac{\partial^2 \bar{\Psi}^e}{\partial \bar{I}_1 \, \partial \bar{I}_2} = \frac{\partial^2 \bar{\Psi}^e}{\partial \bar{I}_2^2} = 0 \, . \tag{2.76}$$

The equations of the stress $\bar{\boldsymbol{\tau}}^e$ and moduli $\bar{\mathbb{C}}^e$ follows from the above results

$$\bar{\boldsymbol{\tau}}^e = 2c_1 \bar{\boldsymbol{b}} + 2c_2 \left(\bar{I}_1 \bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2 \right) \tag{2.77}$$

$$\bar{\mathbb{C}}^e = 4c_2 \left(\bar{\boldsymbol{b}} \otimes \bar{\boldsymbol{b}} - \mathbb{I}_{\bar{\boldsymbol{b}}} \right) \, . \tag{2.78}$$

Similary, for the viscous part of the isochoric response, we need the partial derivatives of $\bar{\Psi}_k^v$ with respect to the invariants

$$\frac{\partial \bar{\Psi}_k^v}{\partial I_{1k}^e} = c_{1k}^v \quad \text{and} \tag{2.79}$$

$$\frac{\partial \bar{\Psi}_k^v}{\partial I_{2k}^e} = c_{2k}^v \, , \tag{2.80}$$

which gives us the equation for the stress $\bar{\boldsymbol{\tau}}_k^v$

$$\bar{\boldsymbol{\tau}}_k^v = 2c_{1k}^v \boldsymbol{b}_k^e + 2c_{2k}^v \left(I_{1k}^e \boldsymbol{b}_k^e - \boldsymbol{b}_k^{e^2} \right) \, . \tag{2.81}$$

The evaluation of Equation (2.81) requires $\boldsymbol{b}_k^e$ which can be obtained through Algorithm 1. For the computation of local tangent (2.52), we need the following derivative of the effective creep rate

$$\frac{\partial \dot{\gamma}_k}{\partial \varepsilon_{k,\beta}} = \frac{1}{2\hat{\tau}_k^2} \sum_{q=1}^{n_{\text{pow}}} q a_{k,q} \left(\frac{\tau_k^v}{\hat{\tau}_k} \right)^{q-2} (\mathcal{D}_k)_\beta \, . \tag{2.82}$$

Using (2.82) in (2.52), we can write the local tangent $\boldsymbol{\mathcal{K}}_k$ in the following compact form

$$(\mathcal{K}_k)_{\alpha\beta} = \delta_{\alpha\beta} + \mu_{1k} \tau'_{k,\alpha} (\mathcal{D}_k)_\beta + \mu_{2k} \left(\bar{\mathcal{T}}_k \right)_{\alpha\beta} \tag{2.83}$$

$$\text{with} \quad \mu_{1k} := \frac{\Delta t}{2\sqrt{2}} \frac{1}{\hat{\tau}_k^3} \sum_{q=2}^{n_{\text{pow}}} a_{k,q} (q-1) \left(\frac{\tau_k^v}{\hat{\tau}_k} \right)^{q-3} \quad \text{and}$$

$$\mu_{2k} := \frac{\Delta t}{\sqrt{2}} \frac{\dot{\gamma}_k}{\tau_k^v} \, .$$

The final quantity required to compute $\bar{\mathbb{C}}^v_{k,\text{algo}}$ and $\boldsymbol{\mathcal{K}}_k$ is the derivative of the principal viscous isochoric stresses

$$(\mathcal{T}_k)_{\alpha\beta} = \frac{\partial \tau_{k,\alpha}}{\partial \varepsilon_{k,\beta}} = 4\Lambda^e_{k,\alpha}\left[c^v_{1k}\delta_{\alpha\beta} + c^v_{2k}\left(I^e_{1k}\delta_{\alpha\beta} + \Lambda^e_{k,\beta} + 2\Lambda^e_{k,\beta}\delta_{\alpha\beta}\right)\right] . \tag{2.84}$$

We now have all the required quantities to perform an algorithmic stress update for a finite element implementation. We summarize this stress update in Algorithm 2.

Algorithm 2: Stress and Moduli update algorithm for the specific constitutive model

```
Given: $\boldsymbol{F}$, $\Delta t$, $(\boldsymbol{C}^v_k)_n$, $\boldsymbol{\theta} = \{c_1, c_2, c^v_{1k}, c^v_{2k}, \hat{\tau}_k, a_{k,q}\}$
Elastic stress and moduli:
    $\bar{\boldsymbol{b}} = \bar{\boldsymbol{F}}\boldsymbol{g}\bar{\boldsymbol{F}}^T$ with $\bar{\boldsymbol{F}} = J^{-1/3}\boldsymbol{F}$
    $\bar{\boldsymbol{\tau}}^e = 2c_1\bar{\boldsymbol{b}} + 2c_2\left(\bar{I}_1\bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2\right)$
    $\bar{\mathbb{C}}^e = 4c_2\left(\bar{\boldsymbol{b}} \otimes \bar{\boldsymbol{b}} - \mathbb{I}_{\bar{\boldsymbol{b}}}\right)$
Viscous stress and moduli:
    Elastic Predictor: $\boldsymbol{b}^{e,\text{tr}}_k = \bar{\boldsymbol{F}}(\boldsymbol{C}^v_k)^{-1}_n\bar{\boldsymbol{F}}^T$
    Inelastic Corrector: Find values of $\boldsymbol{b}^e_k$ and $\bar{\boldsymbol{\tau}}^v_k$ using Algorithm 1 where
        $\bar{\boldsymbol{\tau}}^v_k = 2c^v_{1k}\boldsymbol{b}^e_k + 2c^v_{2k}\left(I^e_{1k}\boldsymbol{b}^e_k - \boldsymbol{b}^{e^2}_k\right)$ with $I^e_{1k} = \text{tr}(\boldsymbol{b}^e_k)$
    History variable update: $\boldsymbol{C}^v_k = \bar{\boldsymbol{F}}^T\boldsymbol{b}^e_k\bar{\boldsymbol{F}}$
    $\tilde{\mathbb{C}}^v_{k,\text{algo}} = \sum_{\alpha=1}^{3}\sum_{\beta=1}^{3}\left(\frac{(\mathcal{T}_k)_{\alpha\eta}(\mathcal{K}_k)^{-1}_{\eta\beta} - 2\tau_{k,\alpha}\delta_{\alpha\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha}\Lambda^{e,\text{tr}}_{k,\beta}}\right)\boldsymbol{M}^k_\alpha \otimes \boldsymbol{M}^k_\beta$
            $+\frac{1}{2}\sum_{\alpha=1}^{3}\sum_{\beta\neq\alpha}^{3}\frac{\tilde{S}_{k,\alpha} - \tilde{S}_{k,\beta}}{\Lambda^{e,\text{tr}}_{k,\alpha} - \Lambda^{e,\text{tr}}_{k,\beta}}\left(\mathbb{G}^k_{\alpha\beta} + \mathbb{G}^k_{\beta\alpha}\right)$
    $\left(\bar{\mathbb{C}}^v_{k,\text{algo}}\right)^{abcd} = \left(F^{e,tr}_k\right)^a_A\left(F^{e,tr}_k\right)^b_B\left(F^{e,tr}_k\right)^c_C\left(F^{e,tr}_k\right)^d_D\left(\tilde{\mathbb{C}}^v_{k,\text{algo}}\right)^{ABCD}$
    $\boldsymbol{\mathcal{T}}_k$ and $\boldsymbol{\mathcal{K}}_k$ are calculated using Equations (2.83) and (2.84), respectively.
Combine Isochoric and Volumetric parts:
    $\boldsymbol{\tau} = p\boldsymbol{g}^{-1} + \mathbb{P} : \left(\bar{\boldsymbol{\tau}}^e + \sum_{k=1}^{n_\text{branch}}\bar{\boldsymbol{\tau}}^v_k\right)$
    $\bar{\mathbb{C}} = \bar{\mathbb{C}}^e + \sum_{k=1}^{n_\text{branch}}\bar{\mathbb{C}}^v_{k,\text{algo}}$
    $\mathbb{C}^\text{iso} = \mathbb{P} : \left[\bar{\mathbb{C}} + \frac{2}{3}\left(\bar{\boldsymbol{\tau}} : \boldsymbol{g}\right)\mathbb{I} - \frac{2}{3}\left(\bar{\boldsymbol{\tau}} \otimes \boldsymbol{g}^{-1} + \boldsymbol{g}^{-1} \otimes \bar{\boldsymbol{\tau}}\right)\right] : \mathbb{P}$
    $\mathbb{C}^\text{vol} = (p + s)\boldsymbol{g}^{-1} \otimes \boldsymbol{g}^{-1} - 2p\mathbb{I}$
    $\mathbb{C} = \mathbb{C}^\text{vol} + \mathbb{C}^\text{iso}$
Return: $\boldsymbol{\tau}$, $\boldsymbol{C}^v_k$, $\mathbb{C}$
```

CHAPTER 3

MODEL DISCOVERY FRAMEWORK

In this chapter, we briefly overview recurrent neural networks (RNNs) and explain the details of the model discovery framework. We describe the recurrent unit we use in the RNN architecture and construct the corresponding loss function explaining the terms included. Additionally, we provide details of the algorithm employed for model training, covering critical aspects such as derivative calculation and hyperparameter tuning.

3.1 Recurrent neural networks

Here, we give a brief introduction of neural networks and activation functions. We also explain the importance of recurrent neural networks in handling sequential data, and how advanced recurrent units handle long-term dependencies effectively.

3.1.1 Neural Networks

Neural networks (NNs) are computational models inspired by the human brain. They are designed to recognize patterns and solve problems by learning from data. Neural networks consist of layers of interconnected nodes (neurons). Each connection has an associated weight that is adjusted during training. A typical neural network comprises an input layer that receives data, one or more hidden layers that perform computations, and an output layer that produces the final result. The neural network shown in Figure 3.1 has a single input layer, two hidden layers, and one output layer.

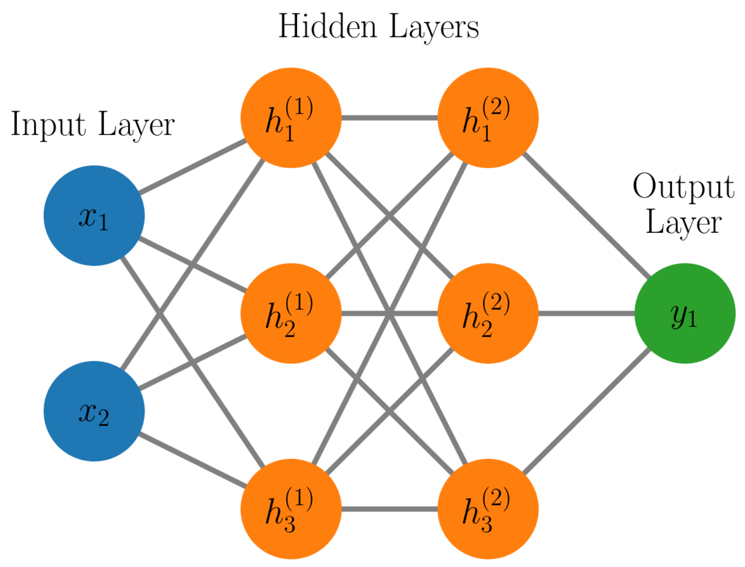


Figure 3.1: A simple neural network with an input layer with two nodes (blue), two hidden layers with three nodes each (orange), and an output layer with a single node (green). The output of this network can be calculated from the inputs using Equation (3.1).

The output can be calculated using

$$\begin{aligned} \boldsymbol{h}^{(1)} &= f_1(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})\,, \\ \boldsymbol{h}^{(2)} &= f_2(\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}) \quad \text{and} \\ \boldsymbol{y} &= f_3(\boldsymbol{W}^{(3)}\boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)})\,, \end{aligned} \tag{3.1}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{h}^{(i)}$ is the hidden vector of the i^{th} layer and $\boldsymbol{y}$ is the output vector. $\boldsymbol{W}^{(i)}$ are the weight matrices corresponding to the connections between the neurons of two adjacent layers, $\boldsymbol{b}^{(i)}$ is the bias vector of the i^{th} layer, and f_i is the activation function of the i^{th} layer.

3.1.2 Activation Functions

Activation functions play a critical role in neural networks by introducing nonlinearities into the model, enabling it to learn complex patterns and perform tasks beyond mere linear classification. These functions help determine the output of individual

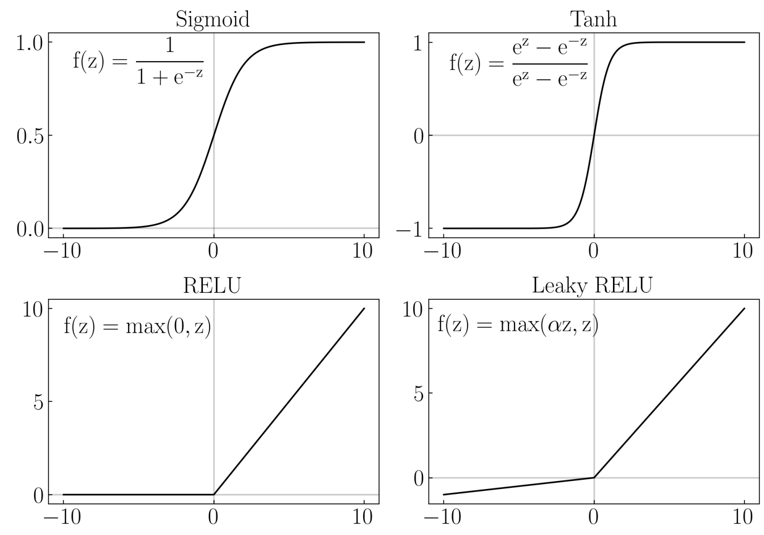


Figure 3.2: Plots of common activation functions for input values between -10 and 10.

neurons and, ultimately, the overall output of the network. If we remove the activation functions from Equation (3.1), the neural network would essentially function as a linear regression model, significantly limiting its ability. Activation functions like the ones shown in Figure 3.2 ensure that neural networks can generalize and adapt to varied datasets, making them indispensable for effective deep learning.

3.1.3 Recurrent units

Traditional NNs like the one in Figure 3.1 assume that inputs are independent. However, many applications, including viscoelastic modeling, involve sequential data where current inputs depend on previous ones. Recurrent neural networks (RNNs) address this by maintaining a memory of previous inputs. In an RNN, each hidden neuron receives inputs from the current and previous time steps, as shown in Figure 3.3, allowing the network to maintain a state over time. The hidden state $\boldsymbol{h}_{n+1}$ and the output $\boldsymbol{y}_{n+1}$ at time step t_{n+1} are given by

$$\begin{aligned} \boldsymbol{h}_{n+1} &= f_h(\boldsymbol{W}_{xh}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hh}\boldsymbol{h}_n + \boldsymbol{b}_h) \quad \text{and} \\ \boldsymbol{y}_{n+1} &= f_y(\boldsymbol{W}_{hy}\boldsymbol{h}_{n+1} + \boldsymbol{b}_y)\,, \end{aligned} \tag{3.2}$$

where $\boldsymbol{x}_{n+1}$ denotes the input vector at time step t_{n+1}, while weight matrices $\boldsymbol{W}$ and bias-vectors $\boldsymbol{b}$ are parameters shared across all time steps of the RNN. The symbols

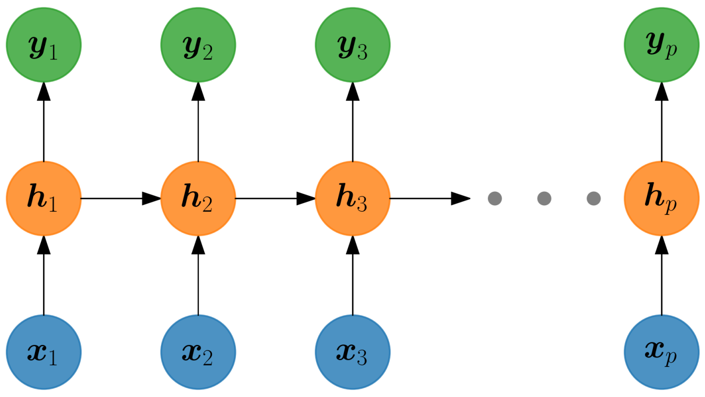


Figure 3.3: A recurrent neural network architecture with each node representing either an input $\boldsymbol{x}$ (blue), hidden $\boldsymbol{h}$ (orange), or output $\boldsymbol{y}$ (green) vector. This architecture's hidden state vectors and outputs can be calculated using Equation (3.2).

f_h and f_y denote activation functions of the hidden and output layers, respectively. This structure allows RNNs to capture dependencies in sequential data. However, standard RNNs struggle with long-term dependencies due to issues like vanishing and exploding gradients. The vanishing gradient problem occurs when gradients become exceedingly small, leading to insignificant weight updates, while the exploding gradient problem involves gradients growing exponentially, causing unstable weight updates. To address these problems, more advanced units such as Long Short-Term Memory (LSTM) [54] and Gated Recurrent Units (GRU) [55] were developed.

LSTM units include input, forget, and output gates to regulate information flow, enabling better long-term dependency management. The input gate controls new information addition, the forget gate discards unnecessary information, and the output gate determines the output. The following equations can describe an LSTM cell

$$\begin{aligned}
\boldsymbol{i}_{n+1} &= \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hi}\boldsymbol{h}_n + \boldsymbol{b}_i)\,,\\
\boldsymbol{f}_{n+1} &= \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hf}\boldsymbol{h}_n + \boldsymbol{b}_f)\,,\\
\boldsymbol{o}_{n+1} &= \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{ho}\boldsymbol{h}_n + \boldsymbol{b}_o)\,,\\
\tilde{\boldsymbol{c}}_{n+1} &= \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hc}\boldsymbol{h}_n + \boldsymbol{b}_c)\,,\\
\boldsymbol{c}_{n+1} &= \boldsymbol{f}_{n+1} \odot \boldsymbol{c}_n + \boldsymbol{i}_{n+1} \odot \tilde{\boldsymbol{c}}_{n+1} \quad \text{and}\\
\boldsymbol{h}_{n+1} &= \boldsymbol{o}_{n+1} \odot \tanh(\boldsymbol{c}_{n+1})\,,
\end{aligned} \tag{3.3}$$

where $\boldsymbol{i}_{n+1}$, $\boldsymbol{f}_{n+1}$, and $\boldsymbol{o}_{n+1}$ are the input, forget, and output gates, respectively. $\boldsymbol{c}_{n+1}$ is the cell state, and $\boldsymbol{h}_{n+1}$ is the hidden state. σ denotes the sigmoid function, and $\odot$ denotes element-wise multiplication. The input gate $\boldsymbol{i}_{n+1}$ decides which new information to add to the cell state by processing the current input $\boldsymbol{x}_{n+1}$ and previous hidden state $\boldsymbol{h}_n$. The forget gate $\boldsymbol{f}_{n+1}$ determines which information from the previous cell state $\boldsymbol{c}_n$ should be discarded, using the same inputs. The output gate $\boldsymbol{o}_{n+1}$ determines the next hidden state $\boldsymbol{h}_{n+1}$ by filtering the current input and previous hidden state. The candidate cell state $\tilde{\boldsymbol{c}}_{n+1}$ is generated from the current input and previous hidden state. The new cell state $\boldsymbol{c}_{n+1}$ is computed by combining the previous cell state, regulated by the forget gate, and the new candidate cell state, regulated by the input gate. Finally, the new hidden state $\boldsymbol{h}_{n+1}$ is derived by applying the output gate to the updated cell state.

GRU units streamline the gating mechanisms found in LSTM by merging the input and forget gates into a single update gate and by incorporating a reset gate that controls the influence of the past hidden state. The GRU cell operates based on the following equations

$$
\begin{aligned}
\boldsymbol{z}_{n+1} &= \sigma(\boldsymbol{W}_{xz}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hz}\boldsymbol{h}_n + \boldsymbol{b}_z)\,, \\
\boldsymbol{r}_{n+1} &= \sigma(\boldsymbol{W}_{xr}\boldsymbol{x}_{n+1} + \boldsymbol{W}_{hr}\boldsymbol{h}_n + \boldsymbol{b}_r)\,, \\
\tilde{\boldsymbol{h}}_{n+1} &= \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}_{n+1} + \boldsymbol{r}_{n+1} \odot (\boldsymbol{W}_{hh}\boldsymbol{h}_n) + \boldsymbol{b}_h) \quad \text{and} \\
\boldsymbol{h}_{n+1} &= (1 - \boldsymbol{z}_{n+1}) \odot \boldsymbol{h}_n + \boldsymbol{z}_{n+1} \odot \tilde{\boldsymbol{h}}_{n+1}\,,
\end{aligned}
\tag{3.4}
$$

where $\boldsymbol{z}_{n+1}$, the update gate, combines the functions of the input and forget gates from the LSTM, deciding both which past information to retain and which new information to add. $\boldsymbol{r}_{n+1}$, the reset gate, determines how much of the past hidden state $\boldsymbol{h}_n$ should be forgotten, influencing the creation of the candidate hidden state $\tilde{\boldsymbol{h}}_{n+1}$. The candidate hidden state $\tilde{\boldsymbol{h}}_{n+1}$ is generated based on the current input $\boldsymbol{x}_{n+1}$ and the conditioned past hidden state, affected by the reset gate. The new hidden state $\boldsymbol{h}_{n+1}$ is then calculated by a weighted update between the old hidden state and the new candidate hidden state, as controlled by the update gate. This structure allows GRUs to effectively manage long-term dependencies in a more streamlined manner than LSTMs.

3.2 Specialized recurrent unit

In this work, we build a customized recurrent unit to perform the algorithmic stress update. The stress update is consistent with the finite viscoelastic model formulations of the previous chapter. The recurrent unit that performs the stress update from time t_n to t_{n+1} is shown in Figure 3.4. Subscripts $n+1$ of the current time step are dropped for convenience.

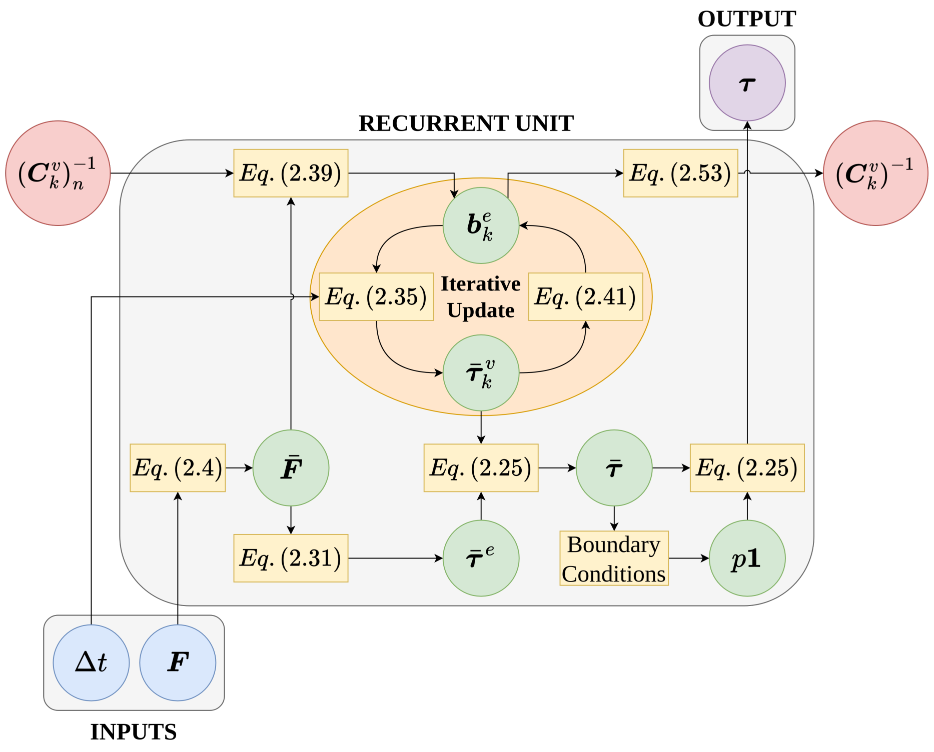


Figure 3.4: Recurrent unit representing the stress update. Subscripts $n+1$ of the current time step are dropped for convenience. Each node denotes a second-order tensor, and we use rectangular boxes to reference equations used for the calculations. Inputs to the recurrent unit are Δt and $\boldsymbol{F}$. $(\boldsymbol{C}_k^v)_n^{-1}$ also acts as an input coming from the previous time step's unit. The main output of the unit is $\boldsymbol{\tau}$, while $(\boldsymbol{C}_k^v)^{-1}$ is the secondary output fed to the next time step's unit. The iterative update (orange ellipse) relates to the implicit update Equation required to calculate $\boldsymbol{b}_k^e$ and $\bar{\boldsymbol{\tau}}_k^v$.

The inputs to the recurrent unit are the deformation gradient $\boldsymbol{F}$ and the change in time Δt. The internal variables $(\boldsymbol{C}_k^v)_n^{-1}$ of each Maxwell branch, an output from the previous time step's stress update, are also an input to the recurrent unit as a hidden state in the context of an RNN. The recurrent unit calculates the unimodular part of the deformation gradient $\bar{\boldsymbol{F}}$ using $\boldsymbol{F}$. For the elastic part of the isochoric stress, the recurrent unit uses $\bar{\boldsymbol{F}}$ to calculate the invariants I_1 and I_2 and, subsequently, to calculate $\bar{\boldsymbol{\tau}}^e$. For the viscous branches, the change in time Δt, the internal variables $(\boldsymbol{C}_k^v)_n^{-1}$, and $\bar{\boldsymbol{F}}$ act as inputs to the iterative update scheme that outputs $\boldsymbol{b}_k^e$ and $\bar{\boldsymbol{\tau}}_k^v$. Adding up the $\bar{\boldsymbol{\tau}}_k^v$ of each branch and $\bar{\boldsymbol{\tau}}^e$ gives us the total isochoric Kirchhoff stress $\bar{\boldsymbol{\tau}}$. The recurrent unit uses boundary conditions to calculate the pressure p. The addition of $p\mathbf{1}$ and $\bar{\boldsymbol{\tau}}$ results in the final output, the total Kirchhoff stress $\boldsymbol{\tau}$. Additionally, a pull-back operation on $\boldsymbol{b}_k^e$ outputs the internal variables $(\boldsymbol{C}_k^v)^{-1}$ that act as inputs to the next time step's recurrent unit.

3.3 Loss function

We assume that sufficient data, either generated synthetically or obtained from physical experiments, is available to characterize the macroscopic finite viscoelastic behavior of the material in question. The dataset comprises multiple temporal sequences, each representing a distinct loading path where an evolving deformation gradient $\boldsymbol{F}$ is applied over time, resulting in corresponding stress measurements $\boldsymbol{\tau}$ at each time step. We regard $z_n^{(s)} = \left(\boldsymbol{F}_n^{(s)}, \boldsymbol{\tau}_n^{(s)}, t_n^{(s)}\right)$ as a data point at time t_n of experiment number s. Let $D = \left\{\left\{z_n^{(s)}\right\}_{n=1}^{N_s}\right\}_{s=1}^{n_{\text{exp}}}$ denote the training dataset where N_s is the total number of time steps of experiment number s, and n_{exp} is the total number of experiments. We define the set of identifiable parameters as $\boldsymbol{\theta}$, and n_{params} denotes the number of identifiable parameters. We construct the loss function,

$$L(\boldsymbol{\theta}; D) = \frac{1}{\left(\sum_{s=1}^{n_{\text{exp}}} N_s\right)} \sum_{s=1}^{n_{\text{exp}}} \sum_{n=1}^{N_s} \left\| \hat{\boldsymbol{\tau}}\left(\boldsymbol{\theta}; \left\{\boldsymbol{F}_i^{(s)}, t_i^{(s)}\right\}_{i=1}^{n}\right) - \boldsymbol{\tau}_n^{(s)} \right\|^2 + \eta_1 \sum_{j=1}^{n_{\text{params}}} |\theta_j| \tag{3.5}$$

where $\hat{\boldsymbol{\tau}}$ is the predicted value of the stress as a function of the history of the inputs $\boldsymbol{F}$ and t up to the current time step. $\boldsymbol{\tau}$ is the true value of the stress and $\|\cdot\| := \sqrt{(\cdot) : (\cdot)}$ denotes the norm of a second-order tensor. The first term quantifies the accuracy of the RNN's predictions by calculating the mean squared error of the predicted stresses.

An additional L_1-regularization term, which calculates the sum of the absolute value of the parameters, is incorporated to promote sparsity. The strength of the sparsification is controlled by the regularization parameter η_1. As shown in [56], the L_1-regularization term pushes some of the learnable parameters to zero, resulting in interpretable models with a few meaningful non-zero parameters.

3.4 Model training

In the process of model training, values of material parameters are identified that minimize the loss function constructed in Equation (3.5)

$$\min_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}; D)\,. \tag{3.6}$$

We use the so-called Adam optimizer [57] for the gradient-based optimization of the defined problem. Calculating the output stresses at each time step requires an iterative update, as shown in Figure 3.4. The iterative update necessitates the calculation of the derivatives analytically as commonly used automatic differentiation (AD) is not possible due to the information flow in the RNN architecture not being unidirectional. Supplying derivatives and allowing the stress update to have an implicit update equation permits using very few data points with bigger time steps to train the model, as the implicit update scheme is unconditionally stable. One can see all the parameter identification algorithm steps in Algorithm 3.

The parameter identification algorithm operates by taking the following inputs: a training dataset D, a randomly initialized set of identifiable parameters $\boldsymbol{\theta}$, the number of update steps n_{steps}, and a specific set of hyperparameters. We initialize the identifiable parameters with small random values to break symmetry and to avoid large initial values that could potentially overshoot towards zero in the first few steps. It is important to note that a non-negativity constraint is included within the algorithm to ensure thermodynamic consistency; this condition sets any parameter that becomes negative after an update to zero. The process runs for the designated n_{steps}, during which $\boldsymbol{\theta}$ is updated at each iteration. The algorithm loops through each experiment within D, and within each experiment, it proceeds through the complete temporal sequence. At every time step, the stress and derivative update is performed. Once

Algorithm 3: Optimization algorithm

```
Input: Training dataset $D$, Randomly initialized parameters $\boldsymbol{\theta}$, Number of
       update steps $n_{\text{steps}}$, Set of hyperparameters $\alpha, \eta_1, \beta_1, \beta_2$
Output: Learned parameters $\boldsymbol{\theta}$
Initialize moment estimates for Adam optimizer
$m_j = 0, v_j = 0 \quad \text{for } j = 1 \text{ to } n_{\text{params}}$ ;
for $l = 1$ to $n_{\text{steps}}$ do
    Initialize derivatives of parameters with respect to loss
    $\frac{dL}{d\theta_j} = 0 \quad \text{for } j = 1 \text{ to } n_{\text{params}}$ ;
    for $s = 1$ to $n_{\text{exp}}$ do
        Initialize internal variable and it's derivative
        $\boldsymbol{\mathcal{I}}_0 = \mathbf{0}, \frac{d\boldsymbol{\mathcal{I}}_0}{d\theta_j} = \mathbf{0} \quad \text{for } j = 1 \text{ to } n_{\text{params}}$ ;
        for $n = 1$ to $N_s$ do
            $\hat{\boldsymbol{\tau}}_n, \boldsymbol{\mathcal{I}}_n = \text{stress\_update}\left(\boldsymbol{\theta}, \boldsymbol{F}_n, \Delta t_n, \boldsymbol{\mathcal{I}}_{n-1}\right)$ ;
            $\frac{d\hat{\boldsymbol{\tau}}_n}{d\boldsymbol{\theta}}, \frac{d\boldsymbol{\mathcal{I}}_n}{d\boldsymbol{\theta}} = \text{derivative\_update}\left(\boldsymbol{\theta}, \boldsymbol{F}_n, \Delta t_n, \boldsymbol{\mathcal{I}}_{n-1} \frac{d\boldsymbol{\mathcal{I}}_{n-1}}{d\boldsymbol{\theta}}, \boldsymbol{\mathcal{I}}_n\right)$ ;
            $\frac{dL}{d\theta_j} \leftarrow \frac{dL}{d\theta_j} + 2\left\|\hat{\boldsymbol{\tau}}_n - \boldsymbol{\tau}_n^{(s)}\right\| \frac{d\hat{\boldsymbol{\tau}}_n}{d\theta_j} \quad \text{for } j = 1 \text{ to } n_{\text{params}}$ ;
        end
    end
    $\frac{dL}{d\boldsymbol{\theta}} \leftarrow \left(\sum_{s=1}^{n_{\text{exp}}} N_s\right)^{-1} \frac{dL}{d\boldsymbol{\theta}}$ ;
    for $j = 1$ to $n_{\text{params}}$ do
        $m_j = \beta_1(m_j) + (1 - \beta_1)\left(\frac{dL}{d\theta_j}\right)$ ;
        $v_j = \beta_2(v_j) + (1 - \beta_2)\left(\frac{dL}{d\theta_j}\right)^2$ ;
        $\hat{m}_j = \frac{m_j}{1 - (\beta_1)^l}, \ \hat{v}_j = \frac{v_j}{1 - (\beta_2)^l}$ ;
        $\theta_j \leftarrow \theta_j - \alpha\left[\left(\frac{\hat{m}_j}{\sqrt{\hat{v}_j} + 1 \times 10^{-8}}\right) + \eta_1 \frac{\theta_j}{|\theta_j|}\right]$
    end
end
return $\boldsymbol{\theta}$ ;
```

all experiments have been processed for all time steps, the derivatives are averaged across the total number of data points. Subsequently, the algorithm updates the moment estimates used by the Adam optimizer and updates each one of the identifiable material parameters. Ultimately, the algorithm concludes by returning an optimized set of identified parameters $\boldsymbol{\theta}$.

3.4.1 Calculation of Derivatives

For the calculation of derivatives of the loss function, we require the derivative of the output stresses with respect to the material parameters. From Equation (3.5), we see that the predicted stress $\hat{\boldsymbol{\tau}}$ depends on the entire history of inputs up to that time step. Based on Equation (2.2) and our use of an RNN to represent the stress update, we can use the reduced form for the Kirchhoff stress $\boldsymbol{\tau}_{n+1}$ at time t_{n+1} in terms of the temporal inputs and internal variables

$$\boldsymbol{\tau}_{n+1} = \hat{\boldsymbol{\tau}}\left(\boldsymbol{\theta}, \boldsymbol{F}_{n+1}, \Delta t_{n+1}; \boldsymbol{\mathcal{I}}_{n+1}\right) , \tag{3.7}$$

where $\boldsymbol{F}_{n+1}$ and Δt_{n+1} are inputs at the current time step as shown in Figure 3.4, while $\boldsymbol{\mathcal{I}}_{n+1}$ represents the internal variables. The calculation of the internal variables requires another equation describing their evolution. We can write the general form of the nonlinear update equation governing the evolution of the internal variables as

$$\boldsymbol{f}\left(\boldsymbol{\theta}, \boldsymbol{\mathcal{I}}_n, \boldsymbol{F}_{n+1}, \Delta t_{n+1}; \boldsymbol{\mathcal{I}}_{n+1}\right) = \boldsymbol{0} \tag{3.8}$$

which requires an iterative update scheme to be solved. Equations (3.7) and (3.8) constitute the total stress update from time t_n to t_{n+1}. The values of $\boldsymbol{\tau}_{n+1}$, $\boldsymbol{\mathcal{I}}_{n+1}$, and $\boldsymbol{\mathcal{I}}_n$ depend on the set of learnable parameters $\boldsymbol{\theta}$ and only $\boldsymbol{F}_{n+1}$ and Δt_{n+1} are fully independent inputs to this system of equations. If we take the derivative of Equations (3.7) and (3.8) with respect to a single parameter θ_j, we obtain

$$\frac{d\boldsymbol{\tau}_{n+1}}{d\theta_j} = \frac{\partial \hat{\boldsymbol{\tau}}}{\partial \theta_j} + \frac{\partial \hat{\boldsymbol{\tau}}}{\partial \boldsymbol{\mathcal{I}}_{n+1}} : \frac{d\boldsymbol{\mathcal{I}}_{n+1}}{d\theta_j} \tag{3.9}$$

$$\frac{\partial \boldsymbol{f}}{\partial \theta_j} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mathcal{I}}_n} : \frac{d\boldsymbol{\mathcal{I}}_n}{d\theta_j} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\mathcal{I}}_{n+1}} : \frac{d\boldsymbol{\mathcal{I}}_{n+1}}{d\theta_j} = \boldsymbol{0} \tag{3.10}$$

Equations (3.9) and (3.10) constitute the analytical recurrent derivative update from time t_n to t_{n+1}. The solution of these equations gives us the derivative of the current time step's stress $d\boldsymbol{\tau}_{n+1}/d\theta_j$ and internal variables $d\boldsymbol{\mathcal{I}}_{n+1}/d\theta_j$ with respect to the parameter θ_j. The derivative of the previous internal variables $d\boldsymbol{\mathcal{I}}_n/d\theta_j$ comes from the previous time step's derivative update. During the identification procedure, the algorithm executes both the stress and derivative updates at each time step, as shown in Algorithm 3. We refer the interested reader to the work of Mahnken and Stein [58], which provides a detailed formulation for the calculation of the analytical derivatives for internal variables-based material models and the related stability investigations for the identification procedure. For the exact equations of the stress and derivative update of the homogenous uniaxial response for the model used in the framework, please see Appendix A. In this work, we used no deep-learning libraries for the optimization algorithm, the stress update procedure, or the derivative update procedure. All functions were implemented using Python and its NumPy [59] library.

3.4.2 Hyperparameter Tuning

The identification procedure in Algorithm 3 requires the user to provide the four hyperparameters. The characteristics of the four hyperparameters are as follows:

- The learning rate α controls the size of the step taken in the direction of the calculated gradient.
- The regularization parameter η_1 penalizes the magnitude of the coefficients to prevent overfitting and promote sparsity.
- The first-moment β_1 and second-moment β_2 hyperparameters, relating to the Adam optimizer, control the exponential decay rate for the moving averages of past gradients and past squared gradients, respectively.

A small learning rate ensures avoiding overshooting the minimum but slows down the overall convergence. On the other hand, a larger learning rate can accelerate convergence but might overshoot the minimum. For the regularization parameter, a larger η_1 can lead to underfitting if too large, oversimplifying the model. Conversely, a smaller value of η_1 may result in an uninterpretable complex model with more than necessary non-zero parameters. For all the representative examples in the subsequent chapter, we set β_1 to 0.9 and β_2 to 0.999, as the original paper that introduced the Adam optimizer recommended [57].

We find reasonable values of α and η_1 by multiple short runs of training the model with small changes to the values of both hyperparameters. We begin with an initial learning rate $\alpha = 1 \times 10^{-3}$, which we observed to produce a continuous and gradual decrease in loss without oscillations when training without regularization on our datasets. Starting with $\alpha = 1 \times 10^{-3}$, we chose an initial value of 1 for η_1 as this resulted in an increase in loss during training on the datasets, indicating over-regularization. With this combination, we expect the initial training run to produce an oversimplified model with low accuracy, reflecting underfitting of the dataset.

The next step is to decrease η_1 in a logarithmically spaced sequence: 1, 3×10^{-1}, 1×10^{-1}, 3×10^{-2}, and so on, until the model's loss plot shows a consistent decrease during the entire short training run instead of increasing or flattening at a high value. Once we finalize η_1, we proceed to increase the learning rate α in the sequence: 1×10^{-3}, 3×10^{-3}, 1×10^{-2}, and 3×10^{-2}, progressively. We continue this until the final run's lowest loss is greater than that of the previous run, indicating potential overshooting. We then select the penultimate α value.

We run each combination of α and η_1 for 1000 update steps to identify the optimal combination. Using the final selected hyperparameters, we conduct an extended run of 10,000 update steps to finalize the model. The value of 10,000 was chosen for the number of steps because the number of non-zero parameters stabilized before the end of the extended training run for the datasets used in this study. From these 10,000 update steps, we select the learnable parameters corresponding to the lowest loss as the final result.

CHAPTER 4

REPRESENTATIVE EXAMPLES

This chapter showcases the proposed framework's applicability to perform model discovery on different datasets. We generate the first dataset synthetically using the model proposed in Section 2.5. The other two datasets are experimental datasets obtained from the literature: one of the VHB 4910 polymer [45], and another of the HNBR50 polymer [11]. We train the model on each dataset's training split and evaluate the predictions on the test split, which is unseen during training. Moreover, we assess the performance of the model learned on the HNBR50 polymer's dataset through a finite element simulation and compare the results with shear loading experimental data.

4.1 Synthetic Data

We generate the synthetic data using a reduced version of the specific constitutive model proposed in Section 2.5. We use two Maxwell viscous branches ($n_{\text{branch}} = 2$) in addition to the elastic branch

$$\bar{\Psi}^e(\bar{I}_1, \bar{I}_2) = c_1(\bar{I}_1 - 3) + c_2(\bar{I}_2 - 3) \tag{4.1}$$

$$\bar{\Psi}^v_k(I^e_{1k}, I^e_{2k}) = c^v_{1k}(I^e_{1k} - 3) + c^v_{2k}(I^e_{2k} - 3) \quad \text{for} \quad k = 1, 2 \tag{4.2}$$

Furthermore, we use five power terms in the effective creep rate ($n_{\text{pow}} = 5$) with only one term active in each branch.

$$\dot{\gamma}_k = \sum_{q=1}^{5} a_{k,q} \left(\frac{\tau^v_k}{\hat{\tau}_k} \right)^q \tag{4.3}$$

4.1.1 Data

We have summarized the values of the material parameters used to generate the data under the column labeled "True" in Tables 4.1 and 4.2. The two active values of the coefficients $a_{k,q}$ are different in their magnitudes to simulate the activation of the viscoelastic behavior of each branch at different loading rates. Moreover, we use different orders of the active power terms - 5th for the first branch and 2nd for the second branch - to simulate different levels of nonlinearity in the viscous response.

We generate the temporal sequences of data by simulating a homogenous uniaxial tension-compression loading that starts at the undeformed state $\lambda_1 = 1$ and is loaded up to a maximum tensile stretch of $\lambda_1^{\text{max}} = 3$ and a minimum compressive stretch of $\lambda_1^{\text{min}} = 0.75$ as shown in Figure 4.1.

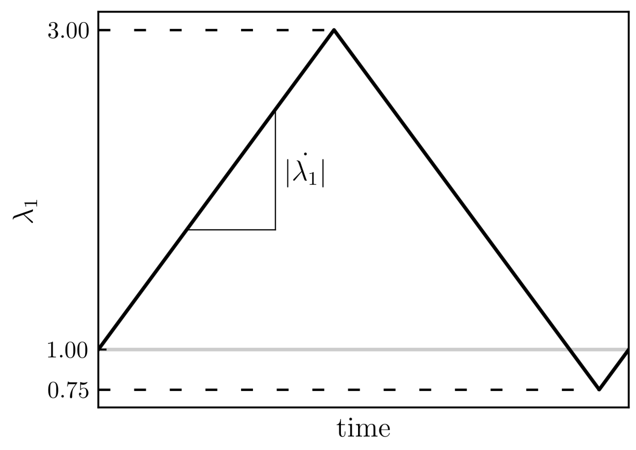


Figure 4.1: Uniaxial tension-compression loading plot for the absolute stretch rate $|\dot{\lambda}_1|$. The loading starts at the undeformed state $\lambda_1 = 1$. Then it is loaded to a maximum tensile stretch of $\lambda_1^{\text{max}} = 3$ followed by unloading and subsequent compressive loading to a minimum compressive stretch of $\lambda_1^{\text{min}} = 0.75$. Finally, the compressive load is removed to return to the undeformed state.

We simulate the loading using four different stretch rates, $|\dot{\lambda}_1| = 5 \times 10^{-3}$, 5×10^{-2}, 5×10^{-1}, $5\ \text{s}^{-1}$, and store data points at time steps $\Delta t = 0.05/|\dot{\lambda}|$ s. The resultant uniaxial nominal stress P_{11} against the uniaxial stretch λ_1 of the generated data can be seen in Figure 4.2

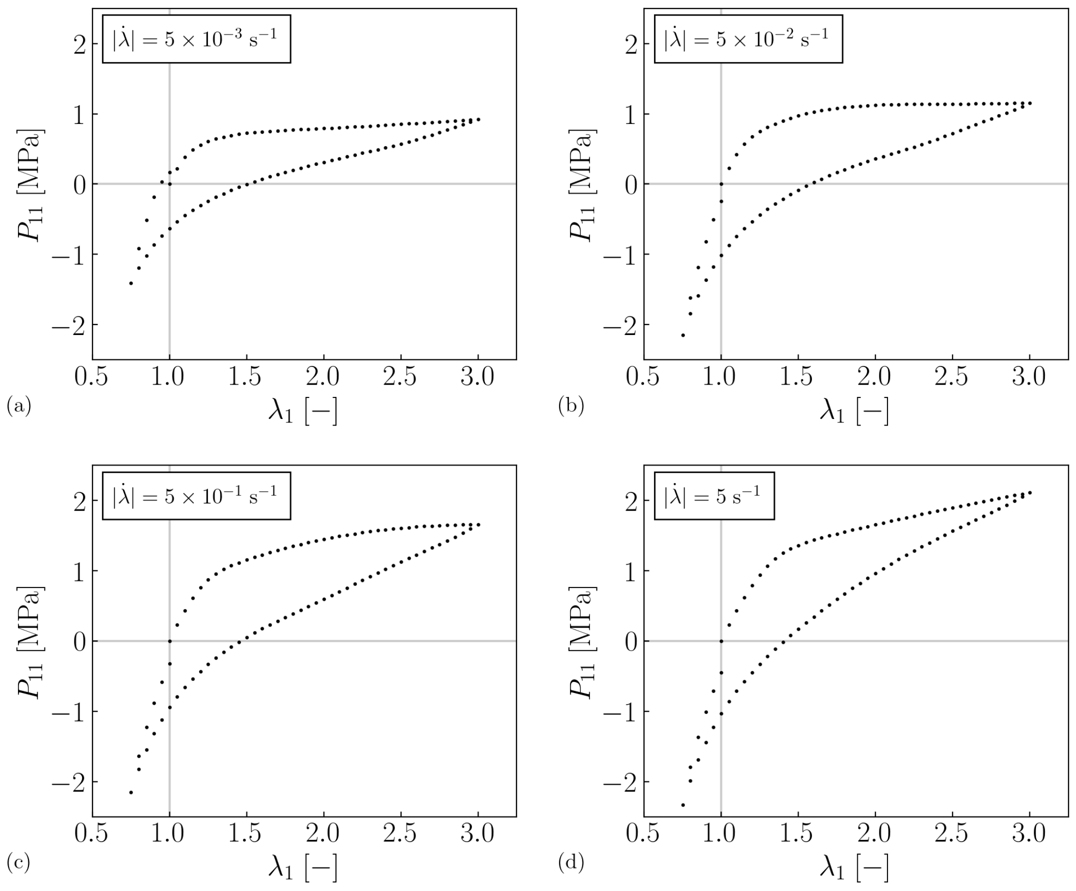


Figure 4.2: Uniaxial nominal stretch-stress response of the synthetically generated dataset. The response is stiffer for higher loading rates (c) and (d) as relaxation of the stress of the second viscous branch, with a smaller coefficient $a_{2,2} = 0.10$ in $\dot{\gamma}_2$, is relatively slower. The relaxation over time is smoother for the slowest rate (a) compared to the highest one (d) due to the difference in order of the selected power terms.

4.1.2 Parameter Identification

For the parameter identification, we train the model using only sequences of the fastest and slowest loading rates. Instead of using the entire sequence, we sample every fourth data point to make the training process faster and showcase the framework's ability to identify parameters with scarce data availability. The sampling is shown in Figure 4.3.

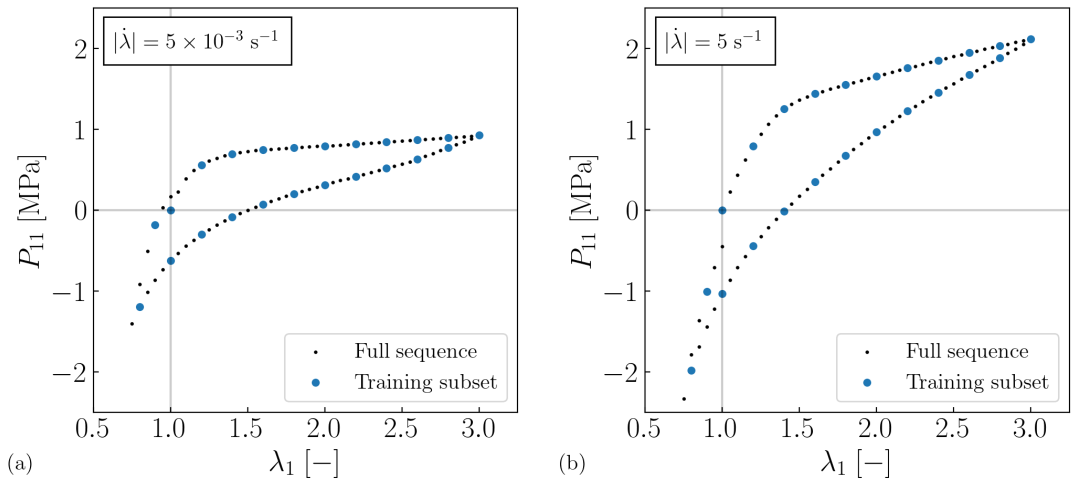


Figure 4.3: Training sampling of the synthetic dataset using the uniaxial nominal stretch-stress response corresponding to the slowest (a) and fastest (b) loading rates. Every fourth data point (blue) is selected from the original sequence (black).

Using the hyperparameter tuning instructions detailed in Section 3.4.2, we identify the suitable combination of hyperparameters as $\alpha = 1 \times 10^{-2}$ and $\eta_1 = 3 \times 10^{-1}$. We execute two separate training sessions with 10,000 update steps ($n_{\text{steps}} = 10,000$) following Algorithm 3. The first session included regularization ($\eta_1 = 3 \times 10^{-1}$), while the second had no regularization ($\eta_1 = 0$). We select the final set of parameters corresponding to the lowest loss of each training session.

For plotting the loss, we use only the first term in Equation (3.5), quantifying the accuracy of the predictions. The variation in the loss of both training sessions against the step number is shown in Figure 4.4. The training without regularization resulted in a lower loss compared to the regularized model. We anticipated this outcome, as regularization tends to drive the learnable parameters towards zero, which reduces accuracy but helps prevent overfitting. However, the minimal difference in loss suggests that the regularized model, despite having significantly fewer parameters as reported in Table 4.2, learned the training dataset almost as effectively.

Interestingly, we observe intermittent spikes in the loss curves of both training sessions. Upon investigation of the evolution of the learnable parameters around these spikes, we found these spikes occur when a previously inactive branch (with zero c^v_{1k} or c^v_{2k}) becomes active (non-zero c^v_{1k} or c^v_{2k}). This activation increases the overall elastic response, causing the derivatives of c^v_{1k} and c^v_{2k} of other branches to indicate decreasing these parameters to fit the dataset better. Such disturbances induce an oscillation in the derivatives, but the Adam optimizer, crafted to accelerate convergence, quickly suppresses these oscillations.

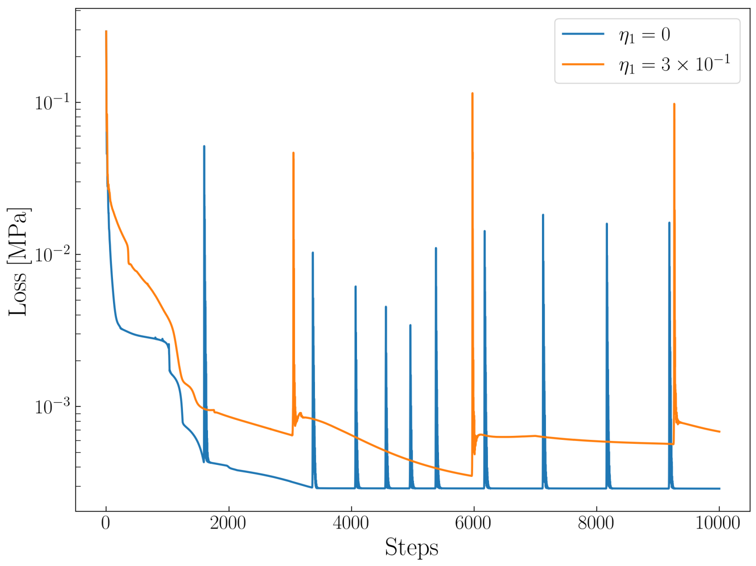


Figure 4.4: Semi-logarithmic plot of the training loss for the synthetically generated data versus the update-step number of the parameter identification procedure. The loss associated with the procedure run without regularization (blue) is lower than that with regularization (orange).

4.1.3 Results

For the training procedure, we used only a sparsely sampled subset of the loading sequences of the two extreme (fastest and slowest) loading rates, excluding the data corresponding to the intermediate loading rates. We plot the predictions of the learned models on both the training and testing splits of the dataset in Figure 4.5.

Firstly, we observe no visible difference between the predictions of the models learned with and without regularization. For the training split, the simulations and data are closely aligned, which was expected since the model used to generate the data is the same as the one that was trained. This means that the model's capacity to fit the data is sufficient. However, the fit on the test split is not as precise, though we still consider it highly satisfactory. We attribute this imprecision to the training data not being comprehensive enough to characterize the overall behavior of the material.

In addition to evaluating the model's accuracy in fitting the data, we also examine the framework's capability to produce interpretable models with few non-zero parameters. We present the parameters used to generate the data and the learned parameters in Tables 4.1 and 4.2. We already know from Figures 4.4 and 4.5 that predictions of the models trained with and without regularization show negligible differences. However, for the identified parameters of the elastic branch reported in Table 4.1, the values obtained using regularization are closer to the true values than those obtained without regularization.

Table 4.1: True and identified material parameters of the elastic branch for the synthetically generated data set

	Elastic Branch		
Parameter	True	Identified ($\eta_1 = 0$)	Identified ($\eta_1 = 3 \times 10^{-1}$)
c_1 (MPa)	0.10	0.03	0.10
c_2 (MPa)	0.10	0.09	0.09

For the identified parameters of the viscous branches reported in Table 4.2, we observe many non-zero identified parameters for the model learned with $\eta_1 = 0$. All the viscous branches are active, which is unnecessary since we generate the data using only two branches. With this set of identified parameters, we can make no assumptions about the underlying viscous behavior of the material. Additionally, the

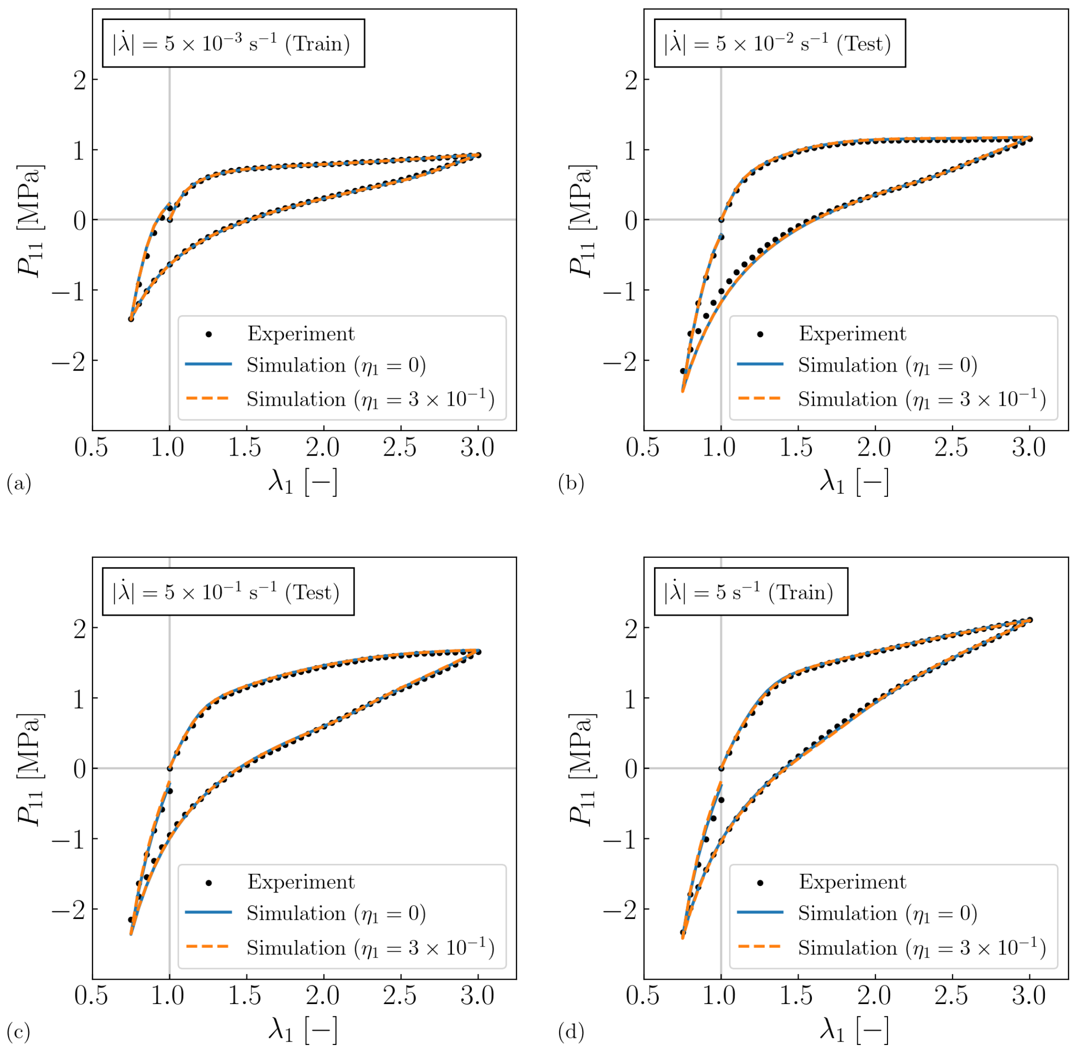


Figure 4.5: Predictions of the learned models on the synthetically generated uniaxial nominal stretch-stress response. Subplots (a) and (d) represent the fastest and slowest loading rates used for training, while subplots (b) and (c) show intermediate loading rates from the test split. Simulations of the regularized $\eta_1 = 3 \times 10^{-1}$ and unregularized $\eta_1 = 0$ models show no difference.

learned model is computationally inefficient for finite element implementation due to the many non-zero parameters and excessive viscous branches.

In contrast, the model learned with $\eta_1 = 3 \times 10^{-1}$ accurately identifies the need for only two active viscous branches. Moreover, branch number 1 has an active coefficient corresponding to the 5th power term, matching branch number 1 of the original model. Similarly, branch number 5 of the learned model has the highest coefficient for the 2nd power term, aligning with branch number 2 of the model used for generating the data. Even though the learned parameters of the regularized model are not exactly the same as the original model, we can infer that the material exhibits viscous phenomena at two distinct loading rates and understand the nonlinearity of the viscoelastic behavior. Moreover, the regularized model's computational efficiency for finite element implementation will be considerably higher than the implementation of the model learned without regularization.

Table 4.2: True and identified material parameters of the viscous branches for the synthetically generated data set

	Branch Number											
	True		Identified ($\eta_1 = 0$)					Identified ($\eta_1 = 3 \times 10^{-1}$)				
Parameter	1	2	1	2	3	4	5	1	2	3	4	5
c_1^v (MPa)	0.40	0.20	0.32	0.07	0.03	0.07	0.09	0.21	-	-	-	0.18
c_2^v (MPa)	-	-	0.14	-	-	0.02	0.03	0.25	-	-	-	0.04
$\hat{\tau}^{-1}$ (MPa^{-1})	1.00	1.00	2.05	-	0.86	0.48	0.45	1.90	-	-	-	0.20
a_1	-	-	-	-	0.20	0.02	-	-	-	-	-	0.02
a_2	-	0.10	-	0.31	3.27	2.40	1.65	-	-	-	-	1.79
a_3	-	-	-	0.87	2.51	3.86	0.57	-	-	-	-	1.32
a_4	-	-	-	0.88	2.91	4.21	0.05	-	-	-	-	-
a_5	100.00	-	3.19	0.34	1.85	2.01	0.40	4.18	-	-	-	-

4.1.4 Robustness of identification in the presence of noise

In this section, we analyze the robustness of the parameter identification procedure in the presence of noise in the experimental data. We again generate the synthetic data set visualized in Figure 4.2 but with added Gaussian noise. We plot the generated noisy data in Figure 4.7 where we add the noise by sampling random values drawn from a normal distribution $\mathcal{N}(\mu, \sigma)$ with a mean μ of zero and a standard deviation σ

of 0.1 MPa.

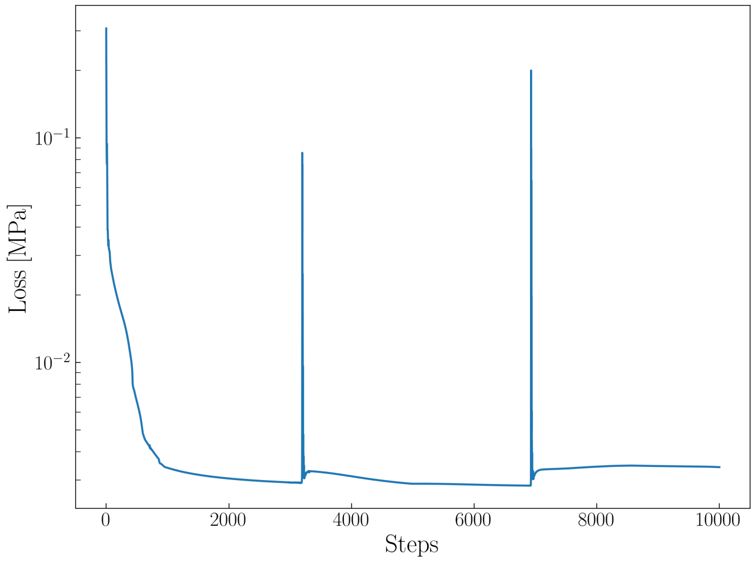


Figure 4.6: Semi-logarithmic plot of the training loss for the synthetically generated noisy data versus the update-step number of the parameter identification procedure.

Similar to the training sampling illustrated in Figure 4.3, we select the fastest and slowest experiments from the noisy dataset and sample every fourth data point. We execute the training procedure with regularization ($\eta_1 = 3 \times 10^{-1}$) and plot the training loss for the parameter identification of the noisy dataset in Figure 4.6. Due to the added noise, the loss values are higher in Figure 4.6 than those in Figure 4.4, that is obtained through training on the dataset without noise. We also observe similar spikes, indicating inactive branches' activation during training.

The predictions of the learned model shown in Figure 4.7 illustrate that the learned model accurately predicts the nonlinear viscoelastic behavior of the training and test data even in the presence of noise. We also report the parameters used to generate the data and the learned parameters in Tables 4.3 and 4.4. For the elastic branch, similar to the previous section, the values of the true and identified parameters are close.

For the identified parameters of the viscous branches reported in Table 4.4, we first notice that the model correctly predicts the need for two branches to characterize the behavior. Moreover, it predicts the nonlinearity of the two branches sufficiently. The power term with the highest value in the first branch of the identified model matches

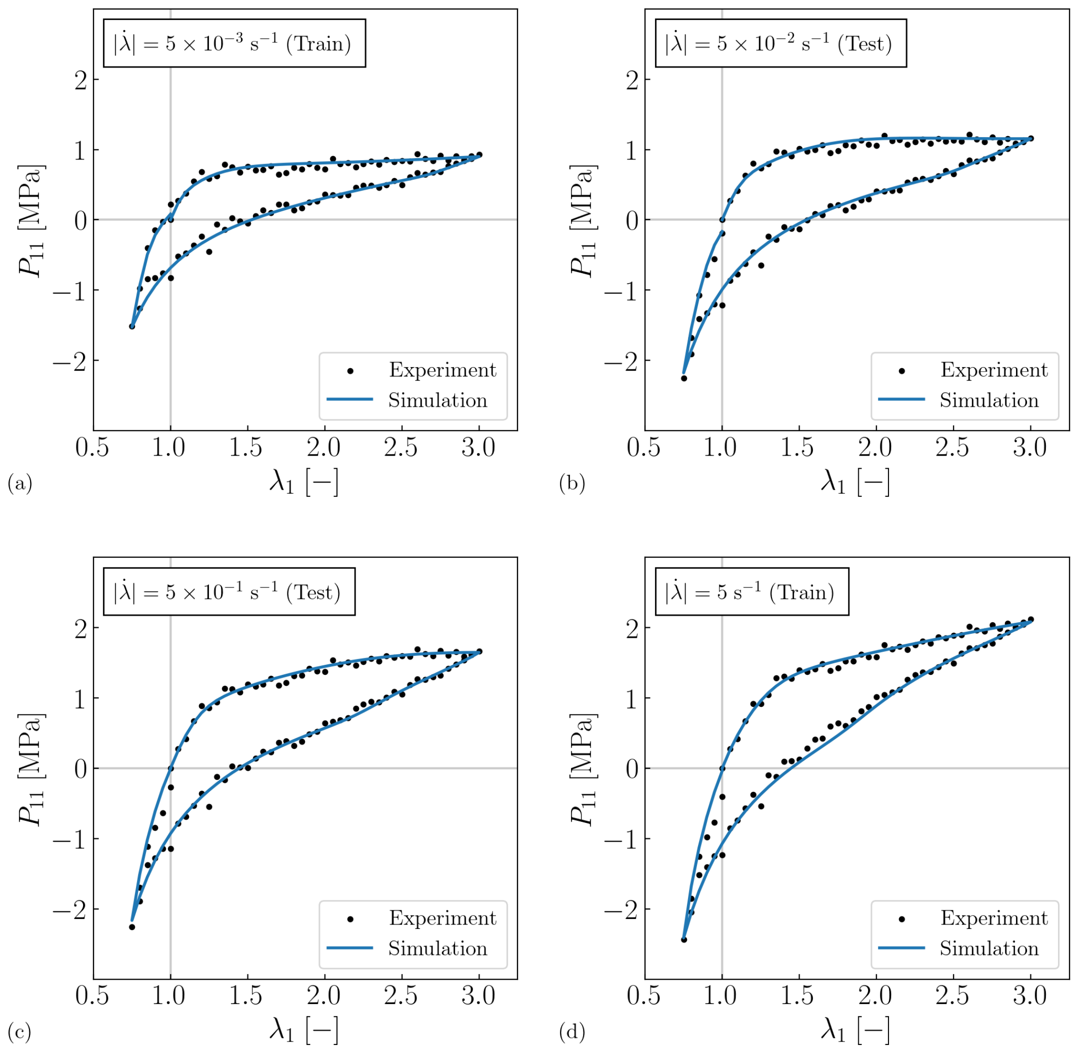


Figure 4.7: Predictions of the learned model on the synthetically generated noisy data.

Table 4.3: True and identified material parameters of the elastic branch for the synthetically generated noisy data set

	Elastic Branch	
Parameter	True	Identified
c_1 (MPa)	0.10	0.09
c_2 (MPa)	0.10	0.12

the first branch of the model used to generate the data. Moreover, the third power term has the maximum value for the fifth branch of the learned model that is close to the active second power term of the second branch of the true model.

Table 4.4: True and identified material parameters of the viscous branches for the synthetically generated noisy data set

	Branch Number						
	True		Identified				
Parameter	1	2	1	2	3	4	5
c_1^v (MPa)	0.40	0.20	0.02	-	-	-	0.20
c_2^v (MPa)	-	-	0.49	-	-	-	-
$\hat{\tau}^{-1}$ (MPa^{-1})	1.00	1.00	1.82	-	-	-	0.20
a_1	-	-	-	-	-	-	-
a_2	-	0.10	-	-	-	-	1.57
a_3	-	-	-	-	-	-	3.17
a_4	-	-	0.72	-	-	-	0.10
a_5	100.00	-	3.66	-	-	-	0.09

The results obtained by training the model on data with added noise show the framework's robustness in accurately predicting the viscoelastic behavior and identifying important characteristics of the material, even in the presence of noise in the experimental data.

4.2 VHB 4910 polymer Data

In this section, we train the model and evaluate the results using the experimental dataset of the VHB 4910 polymer. We selected this dataset because it has been previously used for validation by similar studies on data-driven approaches to finite viscoelasticity [43, 44].

4.2.1 Data

The VHB 4910 polymer is a commercially available acrylic polymer. The dataset we use was generated by Hossain et al. [45] through tensile loading and unloading experiments at different stretch rates. Figure 4.8 shows 11 uniaxial loading sequences from the dataset, where the specimen is loaded to four distinct maximum deformations of $\lambda_1 = 1.5, 2.0, 2.5,$ and 3.0 at three different loading rates: $|\dot{\lambda}_1| = 0.01, 0.03,$ and $0.05\ \mathrm{s}^{-1}$.

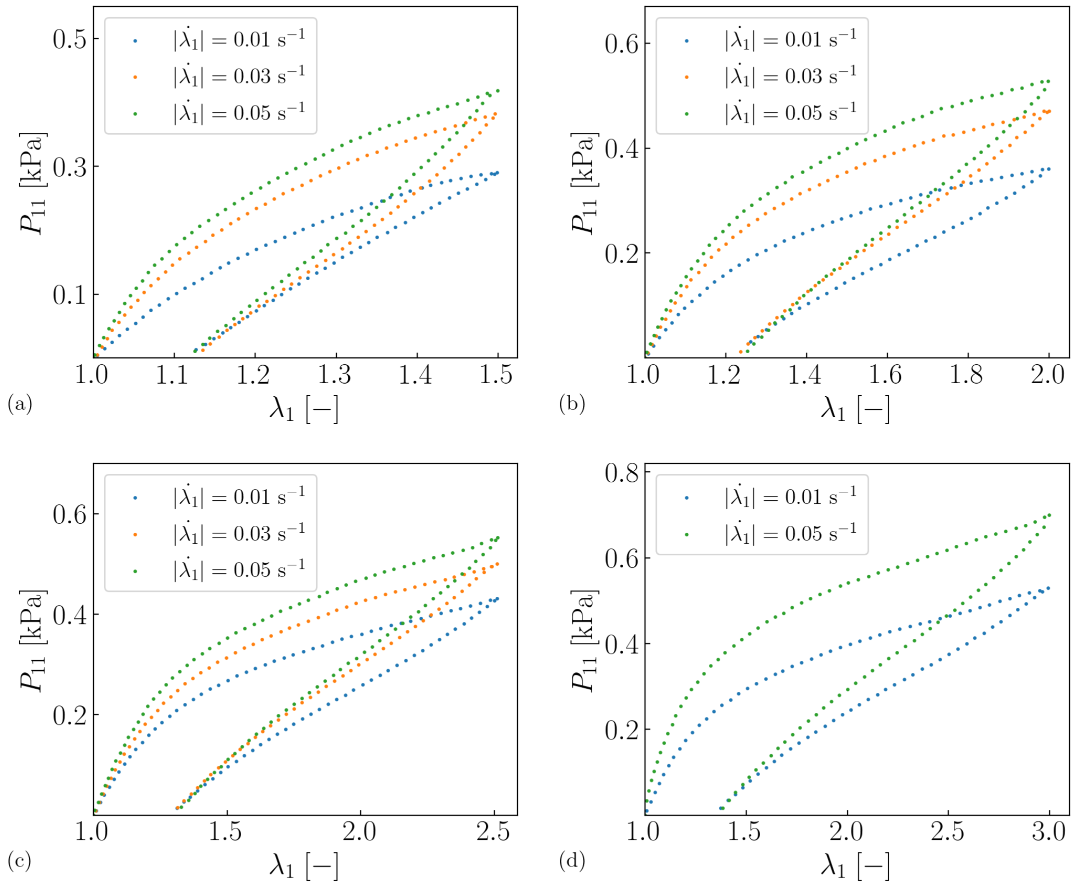


Figure 4.8: Uniaxial nominal stretch-stress response of the VHB 4910 polymer under tensile loading and unloading. Subplots (a), (b), (c), and (d) correspond to maximum deformations of $\lambda_1 = 1.5, 2.0, 2.5,$ and 3.0, respectively. Each maximum deformation is loaded at three distinct loading rates, except for $\lambda_1 = 3.0$, which is loaded at two.

4.2.2 Parameter identification

For model training, we split the dataset into training and test sets. The training set consists of the two loading sequences corresponding to the highest maximum deformation of $\lambda_1 = 3.0$ illustrated in subplot (d) of Figure 4.8. This set should comprehensively characterize the viscoelastic behavior represented in the other loading sequences. Similar to the approach used for the synthetically generated data, we sample every fourth data point from the training set. We ensure that we preserve the curve's shape characteristics when we sparsely sample the data for faster model training. Figure 4.9 illustrates the training sampling.

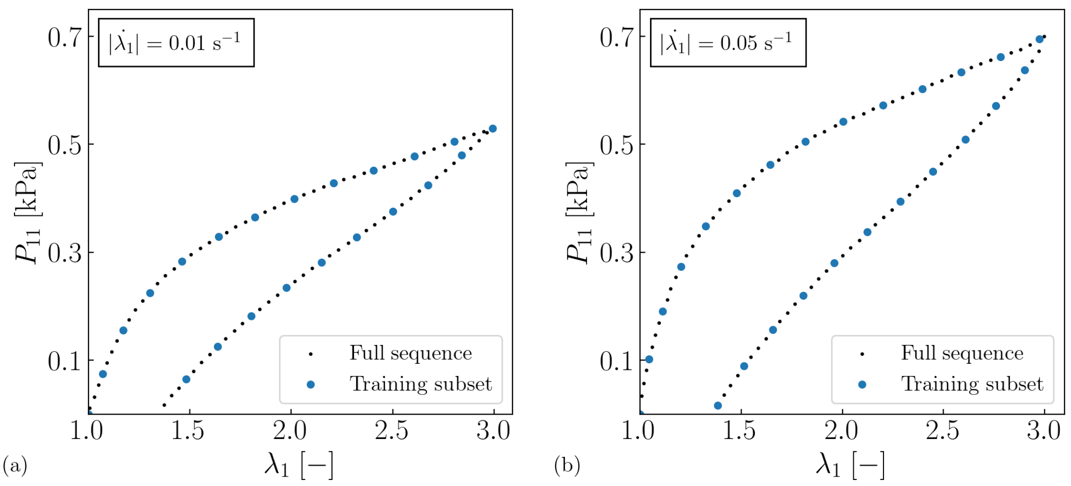


Figure 4.9: Training sampling of the VHB 4910 polymer data using the uniaxial nominal stretch-stress response for the two rates at the maximum deformation of $\lambda_1 = 3.0$. Every fourth data point (blue) is selected from the original sequence (black).

We tune and identify the optimal hyperparameters as $\alpha = 1 \times 10^{-3}$ and $\eta_1 = 1 \times 10^{-1}$. Using these hyperparameters, we perform an extended run of 10,000 learnable parameter update steps and select the final parameters corresponding to the lowest loss within these steps. The progression of the root mean squared error loss is illustrated in Figure 4.10.

The training reaches a loss value lower than 1×10^{-4} kPa within the 10,000 steps, which is sufficiently low, and the learned model should accurately fit the training data. We have explained the reasoning behind the spikes observed in the loss curves

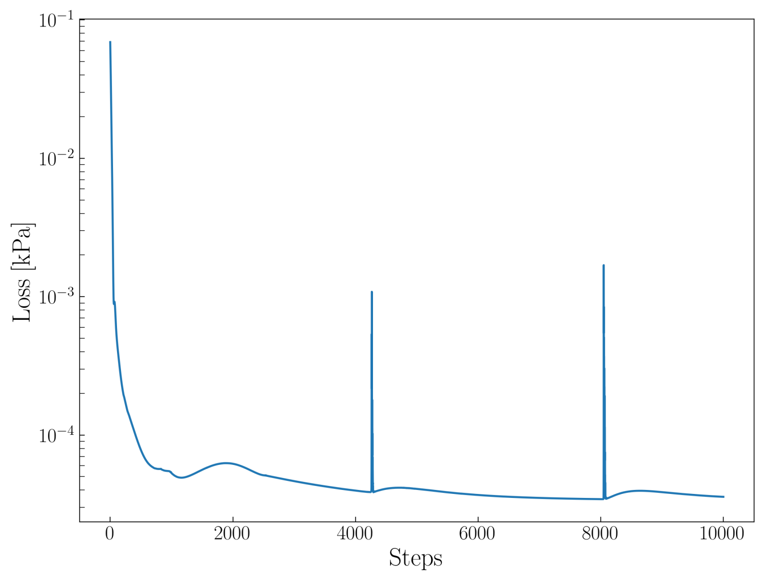


Figure 4.10: Semi-logarithmic plot of the training loss for the VHB 4910 polymer data versus the update-step number during the parameter identification procedure.

in Section 4.1.2.

4.2.3 Results

We selected the sparsely sampled temporal sequences loaded to the highest maximum deformation of $\lambda_1 = 3.0$ at the fastest and slowest loading rates to train the model. We plot the predictions of the learned model on both the training and testing splits of the dataset in Figure 4.11.

The model's predictions on the training data are accurate. For the predictions on the test data, we observe satisfactory predictions of the maximum nominal stresses and relatively adequate alignment of the loading and unloading curves for the maximum deformations $\lambda_1 = 1.5$ and 2.0. However, the model's predictions on the stretch-stress response for the maximum deformation of $\lambda_1 = 2.5$ are inadequate. The results of the model learned using this framework are on par with previous studies that used this dataset. The original study [45] that produced this dataset, as well as later studies employing data-driven approaches [43, 44], to characterize this dataset reported similar inadequacies in the prediction of this dataset.

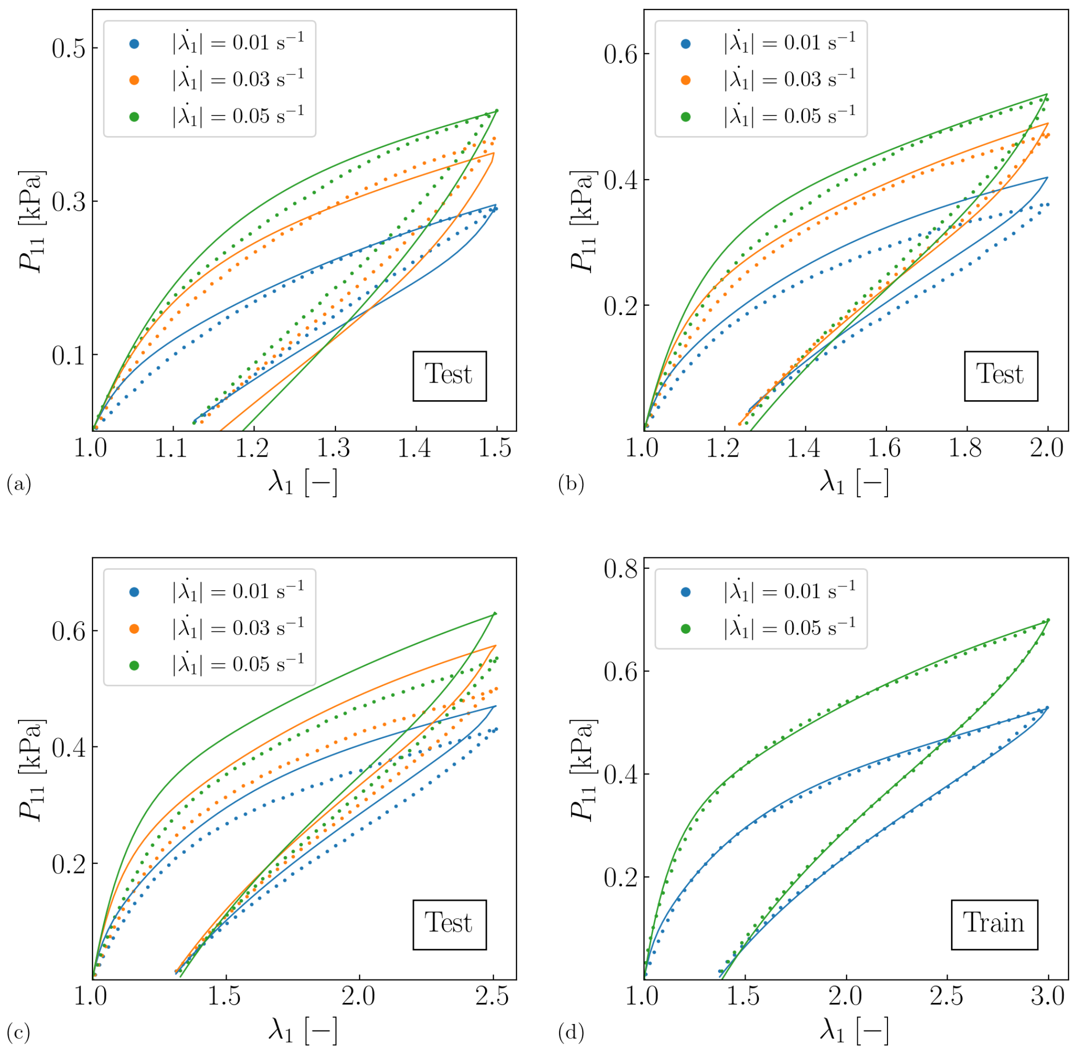


Figure 4.11: Predictions of the learned model on the VHB 4910 polymer's uniaxial nominal stretch-stress response. Solid lines represent the simulations, and the scatter plot represents the experimental data. Subplot (d) shows the training data, while subplots (a), (b), and (c) present the model's predictions on the test data.

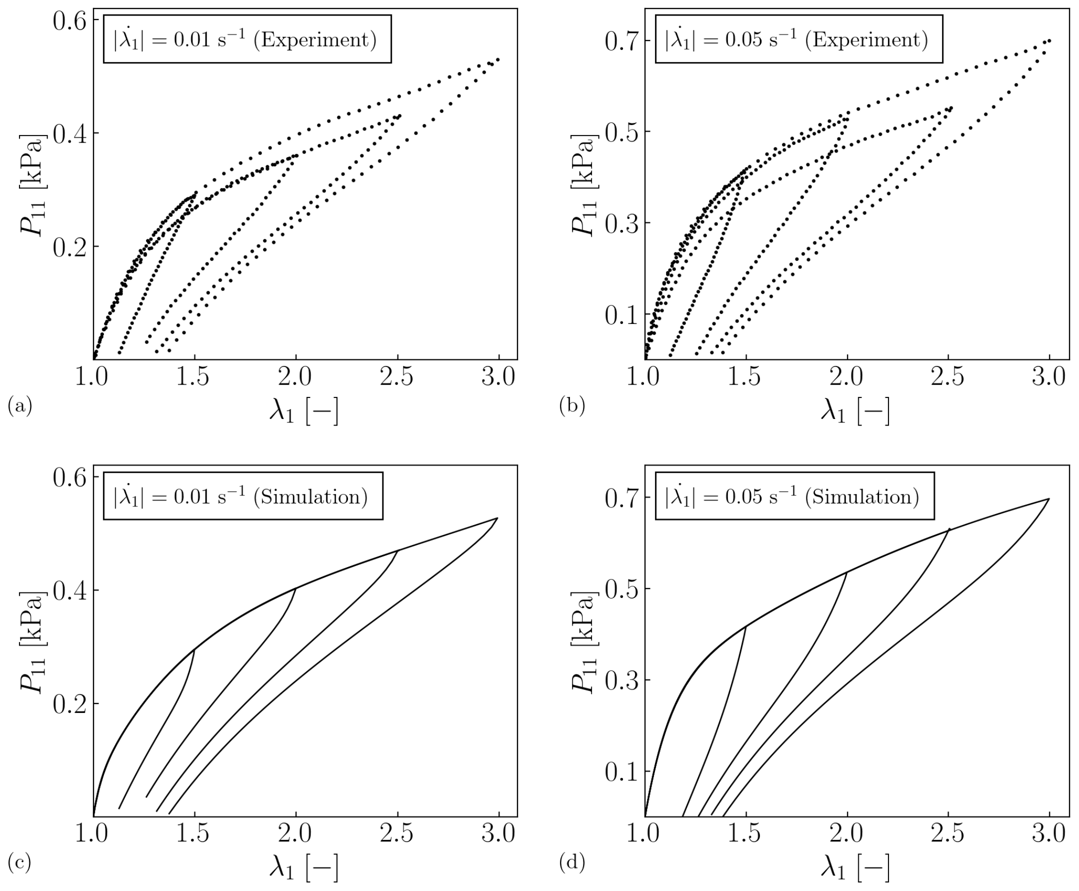


Figure 4.12: Inconsistency in the experimental data (a, b) and consistency in the learned model's simulations (c, d). Subplots (a) and (b) illustrate the misalignment of the stretch-stress response in the experimental data for a specific loading rate at different maximum deformations. In contrast, subplots (c) and (d) demonstrate that the learned model's simulations align.

We analyze the experimental dataset by plotting the loading and unloading curves of the same loading rate together in Figure 4.12. The figure reveals misalignment in the loading curves for the same loading rate, which should align, indicating a possible error in the conducted experiments. The curve showing the highest misalignment corresponds to the maximum deformation of $\lambda_1 = 2.5$, the subset for which our model's predictions were the most inaccurate. On the other hand, the simulations produced by the learned model for the same loading rates show perfect alignment of the loading curves. This alignment demonstrates that our model maintained physical consistency by not fitting to inconsistent data, a quality that might not be present if we used a black-box neural network.

Table 4.5: Identified material parameters of the elastic branch for the VHB 4910 polymer

Parameter	Elastic Branch
c_1 (kPa)	0.08
c_2 (kPa)	-

We report the identified elastic branch parameters in Table 4.5. Only the c_1 parameter, corresponding to the first invariant $\bar{I}_1$ term in the free energy function, is non-zero. The second parameter, c_2, associated with the second invariant $\bar{I}_2$, is zero. The second invariant $\bar{I}_2$ relates to the response under changes in cross-sectional area, which is highly active during compressive loading. Since the VHB 4910 polymer dataset does not include compressive loading data, this explains why c_2 is zero.

Similar to the elastic branch, only the parameters corresponding to the first invariant I^v_{1k} are active in the viscous branch, as summarized in Table 4.6. The identified parameters show that only two viscous branches are needed to fit the training data. The algorithm outputs an efficient constitutive model with eight non-zero identified parameters, a significant reduction from the available 42 parameters. Additionally, the active coefficients correspond only to the first power term, indicating that the dataset requires only a model of linear viscoelasticity to characterize the behavior. This indication contrasts with the suggestion made by Hossain et al. [45] to use finite nonlinear viscoelasticity to improve the accuracy of the predictions.

Table 4.6: Identified material parameters of the viscous branches for the VHB 4910 polymer

	Branch Number				
Parameter	1	2	3	4	5
c_1^v (kPa)	0.29	-	0.07	-	-
c_2^v (kPa)	-	-	-	-	-
$\hat{\tau}^{-1}$ (kPa^{-1})	0.77	-	0.11	-	-
a_1	0.55	-	0.31	-	-
a_2	-	-	-	-	-
a_3	-	-	-	-	-
a_4	-	-	-	-	-
a_5	-	-	-	-	-

4.3 HNBR 50 polymer Data

We selected the experimental dataset of the HNBR50 polymer due to the highly non-linear viscoelastic behavior exhibited by the material. Interestingly, previous data-driven approaches have yet to attempt to characterize the behavior of the HNBR50 polymer. Past efforts have exclusively relied on micromechanically-motivated models [11, 12, 46, 47].

In this section, we train the model and evaluate the results using the experimental data of the HNBR50 polymer. Additionally, we implement the discovered model in a finite element program to demonstrate the framework's compatibility with simulation software and compare the simulation results with actual experimental data.

4.3.1 Data

We obtained this dataset containing experimental data for a highly saturated nitrile rubber (HNBR50) from the study by Miehe and Göktepe [11], which develops the micro-sphere model of finite rubber viscoelasticity. The dataset includes data from homogeneous uniaxial experiments: tension-compression cyclic loadings (Figure 4.13), compressive cyclic loadings (Figure 4.17), and tension-compression cyclic loading with relaxation breaks (Figure 4.18). Additionally, the dataset contains data from a non-homogeneous three-dimensional experiment using a hyperboloid specimen (Figure 4.21).

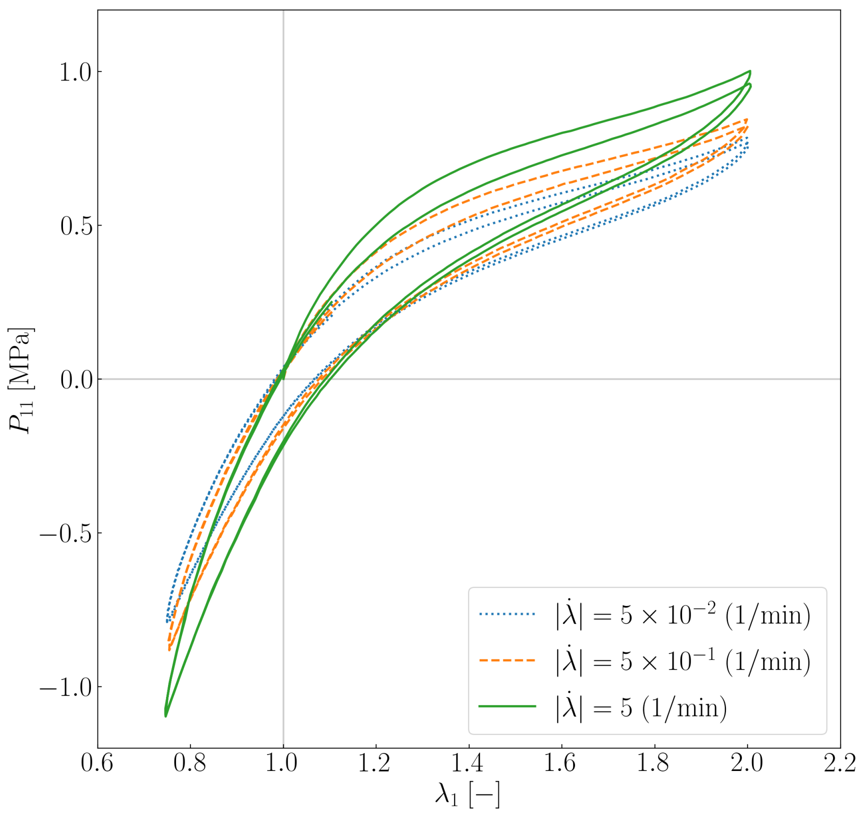


Figure 4.13: Uniaxial nominal stretch-stress response of the HNBR50 polymer under tension-compression cyclic loading. The specimen is loaded for two cycles at three distinct loading rates: $|\dot{\lambda}_1| = 5\times10^{-2}$, 5×10^{-1}, and 5 (1/min) between maximum and minimum stretch values of $\lambda_1^{\max} = 2.0$ and $\lambda_1^{\min} = 0.75$, respectively. The response is stiffer, and the hysteresis is larger at higher loading rates.

4.3.2 Parameter identification

For the parameter identification procedure, we select two temporal sequences corresponding to the fastest $|\dot{\lambda}_1| = 5$ (1/min) and slowest $|\dot{\lambda}_1| = 5 \times 10^{-2}$ (1/min) loading rates of the tension-compression cyclic loading experiments. We will test the model learned using these data sequences on the rest of the dataset. Similar to the approach used for the other two datasets, we sample every tenth and fifteenth data point from the slowest and fastest loadings, respectively. We select the sampling intervals, ensuring we do not lose the definition of the viscoelastic response. The training sampling is shown in Figure 4.14.

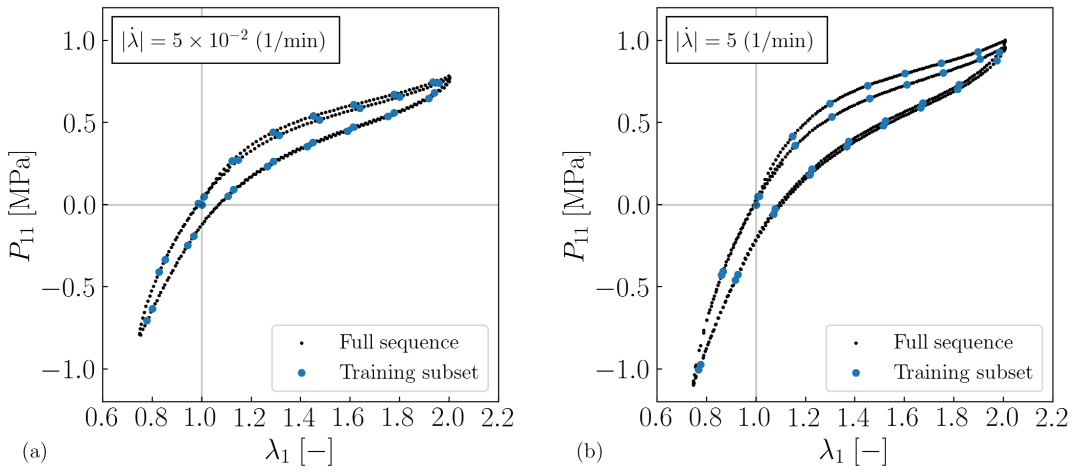


Figure 4.14: Training sampling of the HNBR50 polymer data using tension-compression cyclic loading for the two extreme loading rates (fastest and slowest). Every tenth data point (blue) is selected from the original slowest sequence (black), and every fifteenth data point (blue) is selected from the original fastest sequence (black).

We tune the hyperparameters through repeated training procedures with incremental adjustments, as detailed in Section 3.4.2. The final combination of hyperparameters is found to be $\alpha = 3 \times 10^{-3}$ and $\eta_1 = 3 \times 10^{-1}$. Using these values, we execute the parameter identification procedure with Algorithm 3 for an extended 10,000 update steps. The progression of the loss, specifically the first term in Equation (3.5), which quantifies accuracy, is shown in Figure 4.15.

In Figure 4.15, we observe that the loss has flattened on the logarithmic axis to a value below 1×10^{-3} MPa, indicating convergence of the training procedure. As reported with the other datasets in previous sections, we observe a spike, suggesting the activation of a previously inactive viscous branch. We select the final set of learnable parameters corresponding to the lowest loss among the 10,000 update steps.

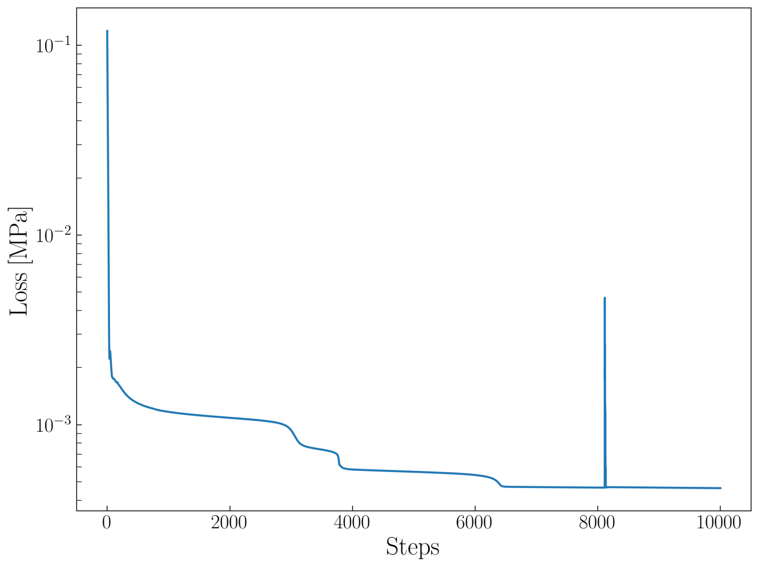


Figure 4.15: Semi-logarithmic plot of the training loss for the HNBR50 polymer data versus the update-step number during the parameter identification procedure.

4.3.3 Results

We trained the model using data from the fastest and slowest tension-compression loading experiments, excluding the data corresponding to the intermediate loading rate. Figure 4.16 shows the experimental data and the learned model's predictions on the nominal stretch-stress response for each tension-compression cyclic loading experiment.

The learned model effectively captures the rate-dependent response in the training data, as shown in Figure 4.16 subplots (b) and (d). Additionally, the simulation for the intermediate loading rate in subplot (c) adequately matches the experimental data, demonstrating the model's excellent interpolation capabilities for unseen data. Overall, the results highlight the relatively simple model's ability to characterize the nonlinear viscoelastic behavior of the material as accurately as previous attempts [11, 12, 46, 47].

We further investigate the learned model's capability to fit unseen data from cyclic compressive loading experiments. We did not train the model using any of the loading sequences shown in Figure 4.17. The model's simulation curves align closely with the experimental data corresponding to the slowest loading rate. While the model's

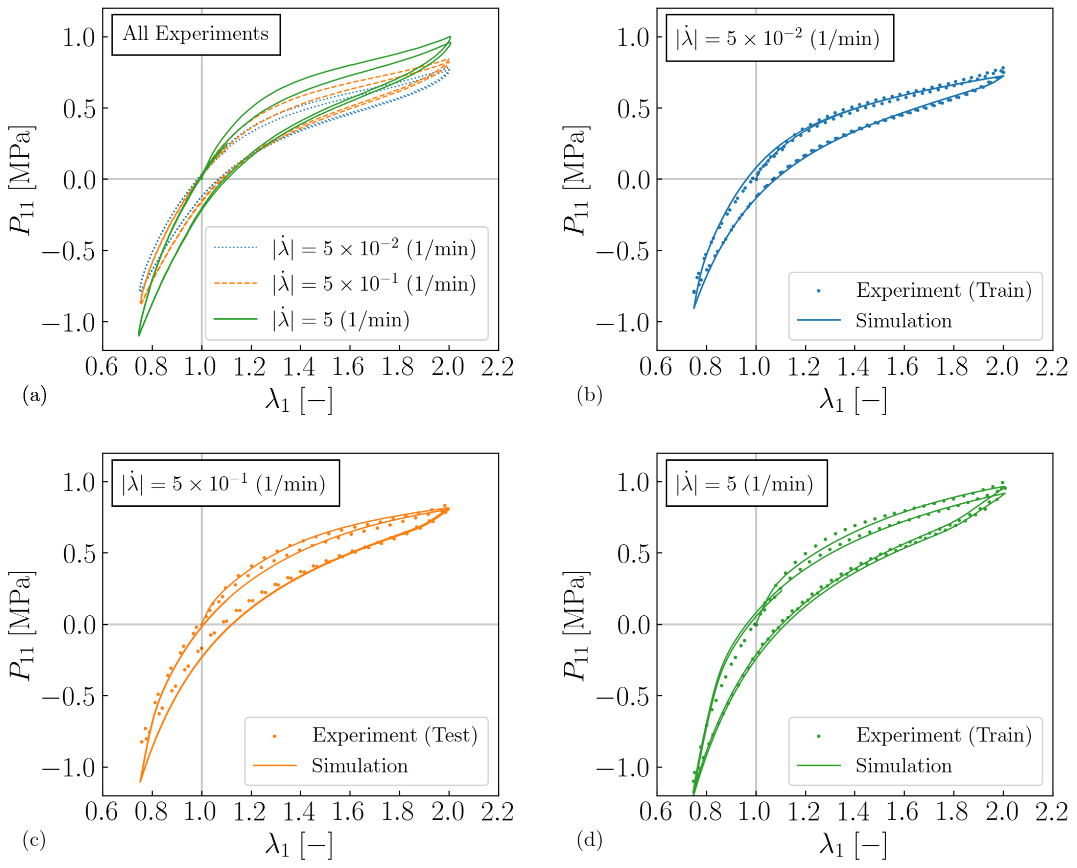


Figure 4.16: Experimental data and corresponding learned model simulations of the HNBR50 polymer under tension-compression cyclic loading. Subplots (b) and (d) show predictions on the training data for the slowest and fastest loading rates, respectively. Subplot (c) presents the model's prediction on the unseen intermediate loading rate data.

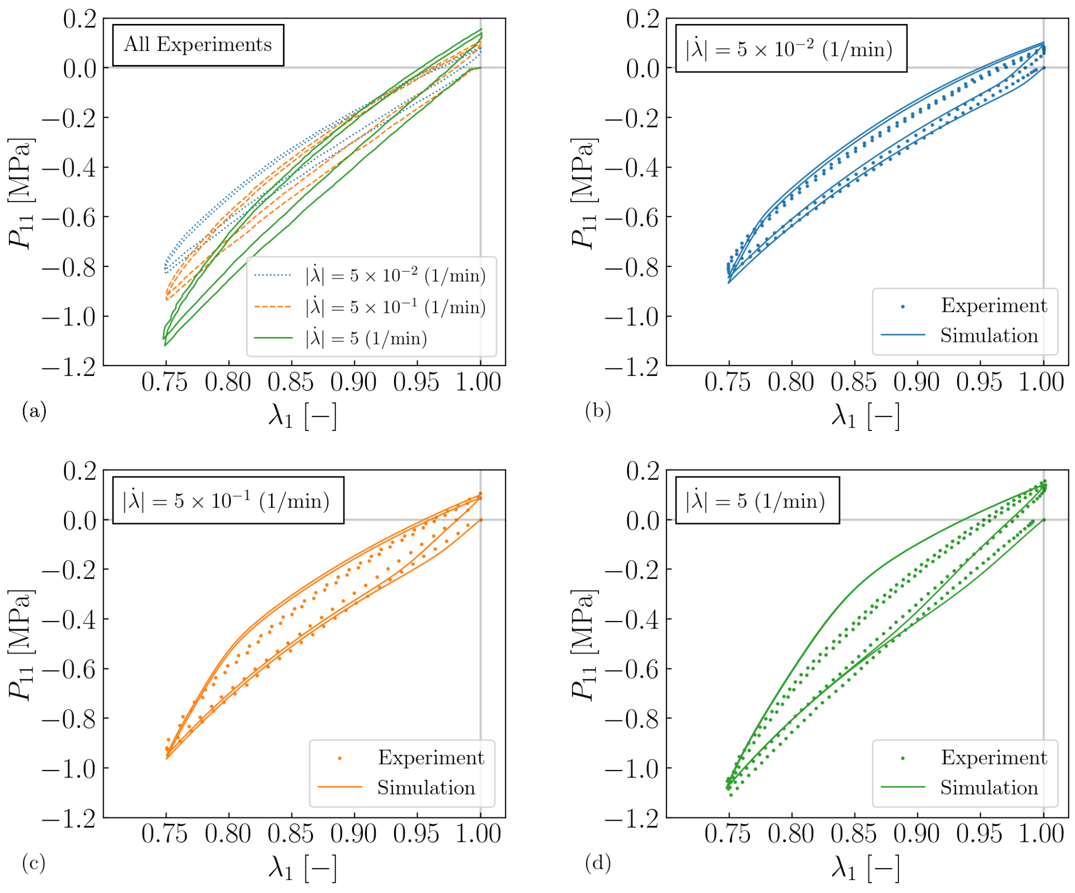


Figure 4.17: Experimental data and corresponding learned model simulations of the HNBR50 polymer under cyclic compressive loading. Subplot (a) presents the experimental data for three loading rates: $|\dot{\lambda}_1| = 5 \times 10^{-2}$, 5×10^{-1}, and 5 (1/min). Subplots (b), (c), and (d) show the model's predictions on the unseen cyclic compressive loading experimental data.

predictions for the two faster loading rates could be more precise, the results are still adequate and comparable to previous efforts fitting this data.

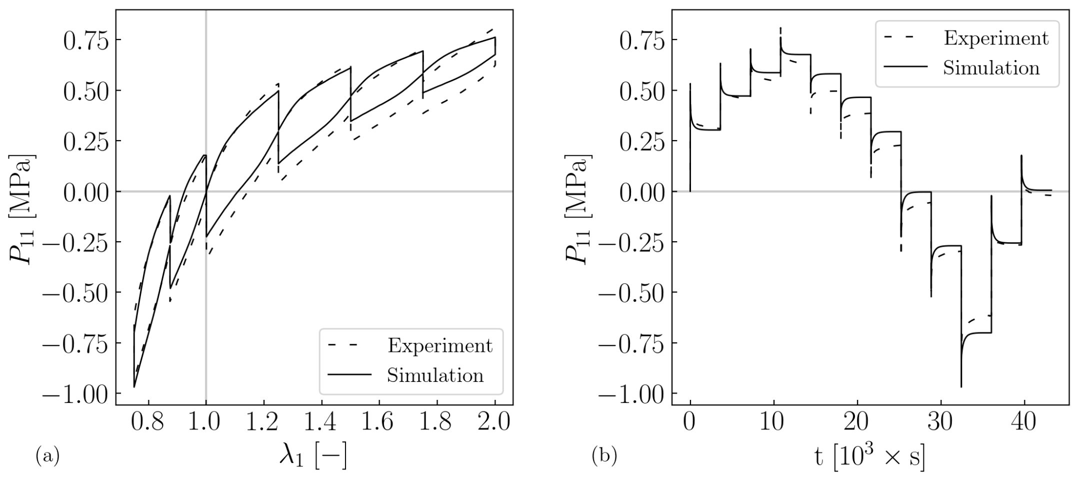


Figure 4.18: Experimental data and learned model simulations of the HNBR50 polymer under tension-compression loading at a rate of $|\dot{\lambda}_1| = 3$ (1/min) with 12 one-hour relaxation breaks. Subplot (a) shows the nominal stress versus stretch, while subplot (b) shows the nominal stress versus time for the same experiment.

Furthermore, we analyze the model's ability to characterize the viscous behavior in relaxation curves obtained from cyclic tension-compression loading with 12 one-hour relaxation breaks. The experimental data and the model's simulations are shown in Figure 4.18. We observe that the predictions of the learned model are fairly adequate compared to past attempts. Overall, we demonstrate that the proposed framework sufficiently captures the finite nonlinear viscoelastic behavior of the HNBR50 polymer using a much simpler model.

Table 4.7: Identified material parameters of the elastic branch for the HNBR50 polymer

Parameter	Elastic Branch
c_1 (MPa)	0.11
c_2 (MPa)	0.17

We examine the two identified material parameters of the elastic branch, as reported in Table 4.7. Both parameters corresponding to the two invariants are active. The training process identifies a larger value for the c_2 parameter, indicating that changes in the specimen's cross-sectional area more influence the underlying elastic response, which is especially significant under compression.

Table 4.8: Identified material parameters of the viscous branches for the HNBR50 polymer

	Branch Number				
Parameter	1	2	3	4	5
c_1^v (MPa)	0.36	-	-	-	0.08
c_2^v (MPa)	-	-	-	-	-
$\hat{\tau}^{-1}$ (MPa^{-1})	2.61	-	-	-	0.44
a_1	-	-	-	-	0.02
a_2	-	-	-	-	0.86
a_3	5.92	-	-	-	6.04
a_4	13.30	-	-	-	8.13
a_5	13.45	-	-	-	4.12

The identified parameters are reported in Table 4.8 for the viscous part. The training procedure reveals that only two viscous branches are necessary, and the active power terms indicate the need for a nonlinear viscoelastic model to characterize the behavior of the HNBR50 polymer. Additionally, we observe that the c_1^v parameter of one branch is approximately four times higher than that of the other, indicating that one branch exhibits significantly more stress relaxation. Finally, the $\hat{\tau}$ parameters suggest that the two branches' viscous behavior activates at different loading rates.

4.3.4 Finite Element Implementation

In the previous section, we analyzed the learned model's ability to capture the nonlinear viscoelastic behavior of the HNBR50 polymer for homogeneous uniaxial loading experiments. In this section, we implement the same learned model in the finite element analysis program (FEAP) [51] and simulate three-dimensional non-homogeneous experiments to compare with experimental data. The geometry and spatial discretization of the specimen into 1,152 eight-node Q1P0 mixed brick finite elements are shown in Figures 4.19 and 4.20.

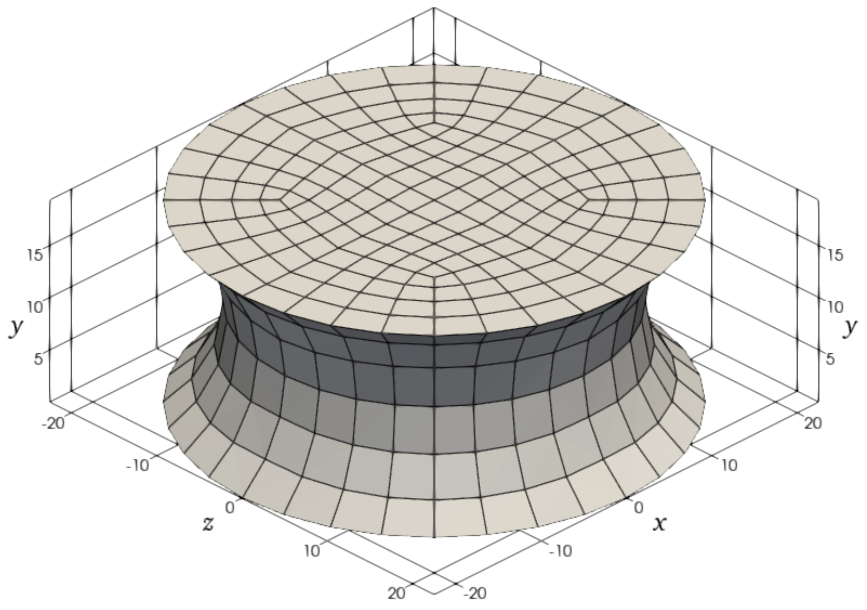


Figure 4.19: Geometry and spatial discretization of the specimen for the finite element analysis. All three axes are in millimeters.

The bottom face of the specimen is fixed, while the top face is deformed only in the x-direction, with movement in the y and z-directions constrained, as shown in Figure 4.20. The actual experiment involved loading the specimen up to a displacement of 10 mm in the positive and negative x-directions for two cycles at two different absolute loading rates, $|\dot{u}(t)| = 4$ and 40 mm/min. For the simulations, we divided each cycle into 240 time steps. Further refinement of the time step has not led to any significant changes in the results. The load versus deflection curves of the experiments and the corresponding simulations are illustrated in Figure 4.21.

The load-deflection curves in Figure 4.21 show that the learned model sufficiently predicts the maximum loads and reasonably approximates the hysteresis curves. The simulations do not indicate non-physical behavior, as the model is physics-informed from the outset and computationally efficient for finite element implementation. Although the curves do not precisely align, these results demonstrate the model's capability to be integrated into finite element software and adequately predict the nonlin-

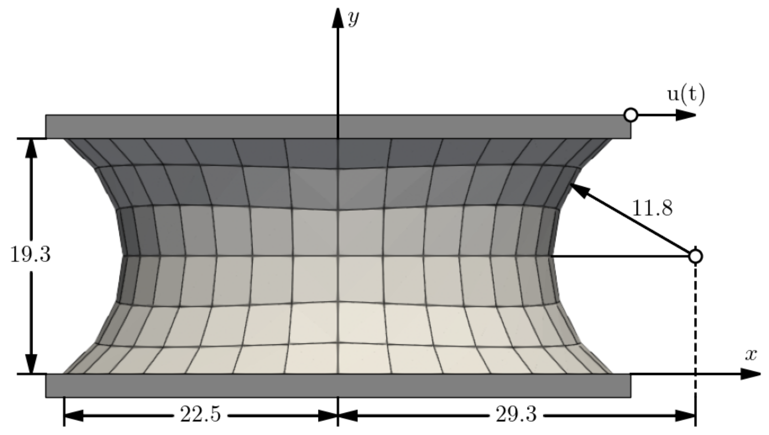


Figure 4.20: Geometry and spatial discretization of the specimen for the finite element analysis. The bottom face, aligned with the x-axis, is fixed in all directions, while the top face is constrained in the y and z-directions. The prescribed displacement in the x-direction on the top face is labeled with $u(t)$. All dimensions are in millimeters.

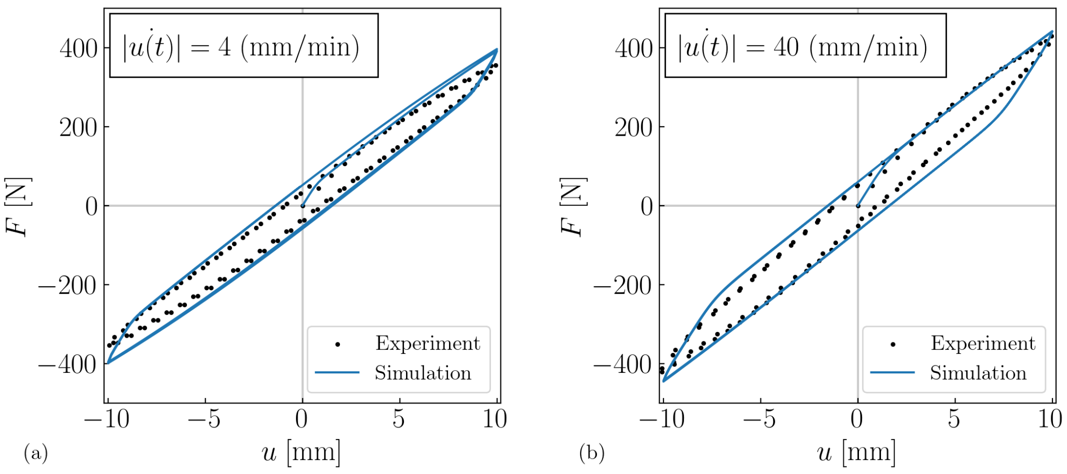


Figure 4.21: Load versus deflection curves of the experimental data and learned model simulations for three-dimensional non-homogeneous loadings at rates of $|\dot{u}(t)| = 4$ and 40 mm/min. The specimen is loaded up to a displacement of 10 mm in the negative and positive x-direction.

ear response of the HNBR50 rubber, even for three-dimensional non-homogeneous loadings.

CHAPTER 5

CONCLUDING REMARKS

This study presents the development of a framework based on a recurrent neural network designed to discover sparse, interpretable models of finite viscoelasticity. A generalized model of finite nonlinear viscoelasticity is constructed, and a specialized recurrent unit is designed to handle the stress update equations. An algorithm with an analytical derivative update function is detailed for efficient parameter identification. The framework's ability to effectively discover models and characterize finite nonlinear viscoelastic behavior is demonstrated through model training and evaluation on three unique datasets.

First, we outlined the theory of macroscopic finite viscoelasticity used in this study. We employed strain-based internal variables based on the multiplicative decomposition of the deformation gradient into elastic and inelastic parts to model nonlinear viscous phenomena. Assuming near incompressibility, we decoupled the response into volumetric and isochoric parts, further decomposing the isochoric part into elastic and viscous components following the generalized Maxwell model. Assuming isotropy, we developed invariant-based Helmholtz free energy functions for the response. To satisfy the second law of thermodynamics, we derived the thermodynamically consistent evolution law of the internal variables. Additionally, we derived equations for the general algorithmic setting of the model, including the implicit update equations required for the integration of the evolution law. We proposed simple constitutive equations that possess a high capacity for modeling finite nonlinear viscoelasticity within the developed framework.

Following the development of the constitutive model, we detailed the architecture of the recurrent neural network with a specialized recurrent unit to handle the stress update equations. We constructed the loss function and incorporated a sparsity term to aid in the model discovery process, ensuring the output of sparse, interpretable models. For the parameter identification procedure, we explained the concept of recurrent derivative calculations and provided analytical derivative update functions for a homogeneous uniaxial response. Additionally, we outlined the process followed to tune the hyperparameters for use in the optimization algorithm finalizing the framework.

The developed framework, which fits the parameters of a generalized model of finite viscoelasticity using a recurrent neural network, can be generalized for varying levels of incompressibility and modified to account for anisotropic material responses. For model discovery, we incorporate a sparsity-inducing term based on the L_1 norm of the learnable parameters, though future research could explore the potential benefits of using other L_p norms where $p < 1$. Additionally, while this framework requires the supply of a recurrent derivative update function, it allows for building a generalized model with an implicit update equation, enabling faster model training with smaller datasets. Future work could develop these implicit update equations to be compatible with automatic differentiation, eliminating the need to supply derivative update functions and allowing the use of advanced deep-learning libraries like PyTorch [60] or TensorFlow [61]. These libraries also facilitate automated hyperparameter tuning with built-in tools that systematically explore different combinations of hyperparameters to optimize model performance. Furthermore, this study only constructed functions for a homogeneous uniaxial response, but future research could extend this framework to fit datasets involving different homogeneous deformation modes (e.g., biaxial, pure shear) or non-homogeneous three-dimensional loadings.

Finally, we trained and tested the developed framework on three datasets with varying characteristics. We generated the first dataset synthetically and showcased the framework's ability to discover the underlying model used to create the data. We also compared the effects of the sparsity-inducing regularization term by training the model with and without regularization. For the second dataset, consisting of VHB 4910 polymer data, we demonstrated that the neural network, based on the viscoelastic model, was physically informed and failed to fit inconsistent data. Moreover, the interpretable model revealed that a linear viscoelastic model was sufficient to characterize the behavior of the VHB 4910 polymer. For the final dataset involving the HNBR50 polymer, we showcased the model's ability to effectively capture the material's highly nonlinear viscoelastic behavior with a simpler model compared to previous attempts from the literature. The results showed that the model trained on tension-compression cyclic experiments could adequately predict the behavior under compression and relaxation loadings. Finally, we demonstrated the model's computational efficiency by implementing it into finite element software, illustrating its capability to predict a non-homogeneous three-dimensional response.

Overall, this thesis has demonstrated the developed framework's ability to discover sparse and interpretable models of finite viscoelasticity from experimental data. The framework represents a significant advancement, potentially reducing the time and specialized effort required to develop functional and reliable constitutive models of finite viscoelasticity for novel materials.

APPENDIX A

EXACT EQUATIONS OF THE STRESS AND DERIVATIVE UPDATE FOR A HOMOGENOUS UNIAXIAL RESPONSE

This appendix specifies equations for the stress and derivative update functions used in Algorithm 3 for a homogenous uniaxial stretch-stress response.

A.1 Stress update

For a homogenous loading, we can use a diagonal representation for the deformation gradient

$$\boldsymbol{F} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}, \tag{A.1}$$

where only the principal stretch component $\lambda := \lambda_1$ is prescribed for a uniaxial loading. Due to the assumed incompressibility ($J = \det(\boldsymbol{F}) = 1$) and isotropy ($\lambda_2 = \lambda_3$), the deformation gradient takes the following form

$$\boldsymbol{F} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda^{-\frac{1}{2}} & 0 \\ 0 & 0 & \lambda^{-\frac{1}{2}} \end{bmatrix}. \tag{A.2}$$

For the incompressible case, the unimodular part of the deformation gradient $\bar{\boldsymbol{F}} = \boldsymbol{F}$. Subsequently, the unimodular part of the left Cauchy-Green deformation tensor $\bar{\boldsymbol{b}} = \bar{\boldsymbol{F}}\bar{\boldsymbol{F}}^T$ is equal to the following

$$\bar{\boldsymbol{b}} = \begin{bmatrix} \lambda^2 & 0 & 0 \\ 0 & \lambda^{-1} & 0 \\ 0 & 0 & \lambda^{-1} \end{bmatrix}, \tag{A.3}$$

where $\det(\bar{\boldsymbol{b}}) = 1$. For the tensors relating to the viscous branches, we first specify the initial value for the stored history variable $(\boldsymbol{b}^e_k)_0 = \mathbf{1}$ as the identity tensor. For the elastic predictor step, we obtain the trial value of the elastic left Cauchy-Green deformation tensor $\boldsymbol{b}^{e,\mathrm{tr}}_k$ using Equation (2.39) for the first time step as

$$\boldsymbol{b}^{e,\mathrm{tr}}_k = \bar{\boldsymbol{F}} \left(\bar{\boldsymbol{F}}\right)^{-1}_0 (\boldsymbol{b}^e_k)_0 \left(\bar{\boldsymbol{F}}\right)^{-T}_0 \bar{\boldsymbol{F}}^T . \tag{A.4}$$

The determinants of all tensors on the right-hand side of Equation ($A.4$) are one, which results in

$$\det(\boldsymbol{b}^{e,\mathrm{tr}}_k) = 1 . \tag{A.5}$$

From Equation (2.41), which is obtained after exponential mapping and subsequent discretization of the evolution law, we get the following for the determinant of the elastic left Cauchy-Green deformation tensor

$$\det(\boldsymbol{b}^e_k) = \det(\exp\left[-2\Delta t \dot{\gamma}_k \mathbf{N}_k\right]) , \tag{A.6}$$

where we have used $\det(\boldsymbol{b}^{e,\mathrm{tr}}_k) = 1$. For the determinant of the exponential tensor, we use the principal values of the tensor and represent the determinant as

$$\begin{aligned} \det(\exp\left[-2\Delta t \dot{\gamma}_k \mathbf{N}_k\right]) &= \prod_{\alpha=1}^{3} \exp\left[-\frac{\sqrt{2}\Delta t}{\tau^v_k} \dot{\gamma}_k \tau'_{k,\alpha}\right] \\ &= \exp\left[-\frac{\sqrt{2}\Delta t}{\tau^v_k} \dot{\gamma}_k \operatorname{tr}\left(\boldsymbol{\tau}^{v,\mathrm{iso}}_k\right)\right] \\ &= \exp[0] = 1 , \end{aligned} \tag{A.7}$$

where we have utilized the fact that the tensor $\boldsymbol{\tau}^{v,\mathrm{iso}}_k$ is deviatoric. From the above result, we can conclude that the determinant of $\det(\boldsymbol{b}^e_k) = 1$ at each time step. Using this result and our assumed isotropy, we can define

$$\boldsymbol{b}^e_k := \begin{bmatrix} \Lambda^e_k & 0 & 0 \\ 0 & (\Lambda^e_k)^{-\frac{1}{2}} & 0 \\ 0 & 0 & (\Lambda^e_k)^{-\frac{1}{2}} \end{bmatrix} , \tag{A.8}$$

where only $\Lambda^e_k := \Lambda^e_{k,1}$ will be the stored history variable. The trial value $\boldsymbol{b}^{e,\mathrm{tr}}_k$ takes a similar form

$$\boldsymbol{b}^{e,tr}_k = \begin{bmatrix} \Lambda^{e,\mathrm{tr}}_k & 0 & 0 \\ 0 & \left(\Lambda^{e,\mathrm{tr}}_k\right)^{-\frac{1}{2}} & 0 \\ 0 & 0 & \left(\Lambda^{e,\mathrm{tr}}_k\right)^{-\frac{1}{2}} \end{bmatrix} \quad \text{with} \quad \Lambda^{e,\mathrm{tr}}_k := \Lambda^{e,\mathrm{tr}}_{k,1} = (\Lambda^e_k)_n \frac{\lambda^2}{\lambda^2_n} . \tag{A.9}$$

We employ the boundary condition $\tau_{22} = \tau_{33} = 0$ for the homogenous uniaxial response to get stress $\boldsymbol{\tau}$ as

$$\boldsymbol{\tau} = \begin{bmatrix} \tau & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} , \tag{A.10}$$

where $\tau := \tau_{11}$. We then calculate the value of the volumetric stress p using

$$\begin{aligned} \tau_{22} &= p + (\boldsymbol{\tau}^{\text{iso}})_{22} \quad \text{and} \\ p &= -(\boldsymbol{\tau}^{\text{iso}})_{22} \end{aligned} \tag{A.11}$$

and get the final form for the output uniaxial stress

$$\tau = (\boldsymbol{\tau}^{\text{iso}})_{11} - (\boldsymbol{\tau}^{\text{iso}})_{22} \, . \tag{A.12}$$

Using the obtained results, we summarize the stress update algorithm for a homogenous uniaxial case in Algorithm 4.

```
Algorithm 4: Stress update algorithm for the homogenous uniaxial case
Given: $\lambda$, $(\Lambda_k^e)_n$, $\lambda_n$, $\Delta t$, $\boldsymbol{\theta} = \{c_1, c_2, c_{1k}^v, c_{2k}^v, \hat{\tau}_k, a_{k,q}\}$
    Get $\bar{\boldsymbol{b}}$ using Equation $(A.3)$ from $\lambda$.
    $\bar{\boldsymbol{\tau}}^e = 2c_1\bar{\boldsymbol{b}} + 2c_2\left(\bar{I}_1\bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2\right)$ with $\bar{I}_1 = \text{tr}(\bar{\boldsymbol{b}})$
for $k = 1$ to $n_{\text{branch}}$ do
    Get $\boldsymbol{b}_k^{e,\text{tr}}$ using Equation $(A.9)$ from $\lambda$, $(\Lambda_k^e)_n$ and $\lambda_n$.
    Calculate values of $\boldsymbol{b}_k^e$ and $\bar{\boldsymbol{\tau}}_k^v$ using Algorithm 1 where
        $\bar{\boldsymbol{\tau}}_k^v = 2c_{1k}^v\boldsymbol{b}_k^e + 2c_{2k}^v\left(I_{1k}^e\boldsymbol{b}_k^e - \boldsymbol{b}_k^{e^2}\right)$ with $I_{1k}^e = \text{tr}(\boldsymbol{b}_k^e)$
    History variable:
        $\Lambda_k^e = (\boldsymbol{b}_k^e)_{11}$
end
    $\bar{\boldsymbol{\tau}} = \bar{\boldsymbol{\tau}}^e + \sum_{k=1}^{n_{\text{branch}}} \bar{\boldsymbol{\tau}}_k^v$
    $\boldsymbol{\tau}^{\text{iso}} = \mathbb{P} : \bar{\boldsymbol{\tau}}$
    $\tau = (\boldsymbol{\tau}^{\text{iso}})_{11} - (\boldsymbol{\tau}^{\text{iso}})_{22}$
Return: $\tau$, $\Lambda_k^e$
```

A.2 Derivative update

In the recurrent derivative update, we calculate the derivatives of the output stress τ and the internal variable Λ_k^e of the current time step with respect to each learnable parameter. For the parameters c_1 and c_2 of the elastic branch, the calculation is straightforward

$$\frac{d\Lambda_k^e}{d\theta_j} = 0 \quad \text{and} \tag{A.13}$$

$$\frac{d\tau}{d\theta_j} = \left(\frac{\partial \boldsymbol{\tau}^{e,\text{iso}}}{\partial\theta_j}\right)_{11} - \left(\frac{\partial \boldsymbol{\tau}^{e,\text{iso}}}{\partial\theta_j}\right)_{22}, \tag{A.14}$$

where $\boldsymbol{\tau}^{e,\text{iso}} := \mathbb{P} : \bar{\boldsymbol{\tau}}^e$. The following partial derivatives are required to complete the evaluation

$$\frac{\partial \boldsymbol{\tau}^{e,\text{iso}}}{\partial c_1} = \mathbb{P} : 2\bar{\boldsymbol{b}} \quad \text{and} \tag{A.15}$$

$$\frac{\partial \boldsymbol{\tau}^{e,\text{iso}}}{\partial c_2} = \mathbb{P} : 2(\bar{I}_1\bar{\boldsymbol{b}} - \bar{\boldsymbol{b}}^2)\,. \tag{A.16}$$

On the other hand, for the viscous parameters of the k^{th} branch, we consider the equation of the output stress

$$\tau = \hat{\tau}(\theta_j; \Lambda_k^e) \tag{A.17}$$

and the nonlinear equation governing the internal variable update of the k^{th} branch

$$f_k(\theta_j, \Lambda_k^e, (\Lambda_k^e)_n) = 0\,. \tag{A.18}$$

Taking the derivatives of Equations $(A.17)$ and $(A.18)$ with respect to a parameter θ_j, we get

$$\frac{d\tau}{d\theta_j} = \frac{\partial\hat{\tau}}{\partial\theta_j} + \frac{\partial\hat{\tau}}{\partial\Lambda_k^e}\frac{d\Lambda_k^e}{d\theta_j} \quad \text{and} \tag{A.19}$$

$$\frac{\partial f_k}{\partial\theta_j} + \frac{\partial f_k}{\partial\Lambda_k^e}\frac{d\Lambda_k^e}{d\theta_j} + \frac{\partial f_k}{\partial(\Lambda_k^e)_n}\frac{d\,(\Lambda_k^e)_n}{d\theta_j} = 0\,, \tag{A.20}$$

which can be recast into the following form

$$\underbrace{\begin{bmatrix} 1 & -\dfrac{\partial\hat{\tau}}{\partial\Lambda_k^e} \\ 0 & \dfrac{\partial f_k}{\partial\Lambda_k^e} \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \dfrac{d\tau}{d\theta_j} \\ \dfrac{d\Lambda_k^e}{d\theta_j} \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \dfrac{\partial\hat{\tau}}{\partial\theta_j} \\ -\dfrac{\partial f_k}{\partial\theta_j} - \dfrac{\partial f_k}{\partial(\Lambda_k^e)_n}\dfrac{d\,(\Lambda_k^e)_n}{d\theta_j} \end{bmatrix}}_{\boldsymbol{y}}\,. \tag{A.21}$$

The matrix $\boldsymbol{A}$ and the partial derivative $\dfrac{\partial f_k}{\partial (\Lambda_k^e)_n}$ is independent of the parameters, and the value $\dfrac{d\,(\Lambda_k^e)_n}{d\theta_j}$ comes from the previous time step's derivative update. The function $\hat{\tau}$ is the same as Equation $(A.12)$ and the function f_k can be obtained using Equation (2.42)

$$f_k = \Lambda_k^e - \exp\left[\nu_k \dot{\gamma}_k\right] \Lambda_k^{e,\mathrm{tr}} \quad \text{with} \quad \nu_k := -\frac{\sqrt{2}\Delta t}{\tau_k^v} \tau'_{k,1}\,. \tag{A.22}$$

For the first term $\dfrac{\partial \hat{\tau}}{\partial \Lambda_k^e}$ in matrix A, we get

$$\frac{\partial \hat{\tau}}{\partial \Lambda_k^e} = \left(\frac{\partial \boldsymbol{\tau}_k^{v,\mathrm{iso}}}{\partial \Lambda_k^e}\right)_{11} - \left(\frac{\partial \boldsymbol{\tau}_k^{v,\mathrm{iso}}}{\partial \Lambda_k^e}\right)_{22}, \tag{A.23}$$

where

$$\frac{\partial \boldsymbol{\tau}_k^{v,\mathrm{iso}}}{\partial \Lambda_k^e} = \mathbb{P} : 2\left(\left(c_{1k}^v + I_{1k}^e c_{2k}^v\right) \frac{\partial \boldsymbol{b}_k^e}{\partial \Lambda_k^e} + c_{2k}^v \boldsymbol{b}_k^e \frac{\partial I_{1k}^e}{\partial \Lambda_k^e} - c_{2k}^v \frac{\partial \boldsymbol{b}_k^{e^2}}{\partial \Lambda_k^e} \right) \tag{A.24}$$

$$\text{with} \quad \frac{\partial \boldsymbol{b}_k^e}{\partial \Lambda_k^e} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{2}\left(\Lambda_k^e\right)^{-\frac{3}{2}} & 0 \\ 0 & 0 & -\frac{1}{2}\left(\Lambda_k^e\right)^{-\frac{3}{2}} \end{bmatrix},$$

$$\frac{\partial I_{1k}^e}{\partial \Lambda_k^e} = 1 - \left(\Lambda_k^e\right)^{-\frac{3}{2}} \quad \text{and}$$

$$\frac{\partial \boldsymbol{b}_k^{e^2}}{\partial \Lambda_k^e} = \begin{bmatrix} 2\Lambda_k^e & 0 & 0 \\ 0 & -\left(\Lambda_k^e\right)^{-2} & 0 \\ 0 & 0 & -\left(\Lambda_k^e\right)^{-2} \end{bmatrix}.$$

Furthermore, for the second term $\dfrac{\partial f_k}{\partial \Lambda_k^e}$, we obtain

$$\frac{\partial f_k}{\partial \Lambda_k^e} = 1 - \boldsymbol{\Gamma}_k : \frac{\partial \boldsymbol{\tau}_k^{v,\mathrm{iso}}}{\partial \Lambda_k^e}, \tag{A.25}$$

where $\dfrac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial \Lambda_k^e}$ is already calculated in Equation $(A.24)$ and we have defined the second-order tensor $\boldsymbol{\Gamma}_k$ as

$$\boldsymbol{\Gamma}_k := \frac{\partial\big(\exp\left[\nu_k \dot{\gamma}_k\right] \Lambda_k^{e,\text{tr}}\big)}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} = \exp\left[\nu_k \dot{\gamma}_k\right] \Lambda_k^{e,\text{tr}} \left(\nu_k \frac{\partial \dot{\gamma}_k}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} + \dot{\gamma}_k \frac{\partial \nu_k}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} \right) \tag{A.26}$$

$$\text{with} \quad \frac{\partial \dot{\gamma}_k}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} = \left(\frac{1}{2\hat{\tau}_k^2} \sum_{q=1}^{n_\text{pow}} a_{k,q} (q-1) \left(\frac{\tau_k^v}{\hat{\tau}_k} \right)^{q-2} \right) \boldsymbol{\tau}_k^{v,\text{iso}} \quad \text{and}$$

$$\frac{\partial \nu_k}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} = \nu_k \left(\frac{\partial \tau'_{k,1} / \partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\tau'_{k,1}} - \frac{\mathbf{N}_k}{\sqrt{2}\tau_k^v} \right) \quad \text{with} \quad \frac{\partial \tau'_{k,1}}{\partial \boldsymbol{\tau}_k^{v,\text{iso}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{A.27}$$

Lastly, we get the partial derivative of Equation $(A.22)$ with respect to the internal variable of the previous time step $(\Lambda_k^e)_n$

$$\frac{\partial f_k}{\partial (\Lambda_k^e)_n} = -\exp[\nu_k \dot{\gamma}_k] \frac{\lambda^2}{\lambda_n^2}. \tag{A.28}$$

For the term $\dfrac{\partial \hat{\tau}}{\partial \theta_j}$ in vector $\boldsymbol{y}$ of Equation $(A.21)$, we obtain

$$\frac{\partial \hat{\tau}}{\partial \theta_j} = \left(\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial \theta_j} \right)_{11} - \left(\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial \theta_j} \right)_{22}. \tag{A.29}$$

We only need the partial derivatives $\dfrac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial \theta_j}$ and $\dfrac{\partial f_k}{\partial \theta_j}$ for each individual parameter of the viscous branches. The values are given as

$$\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial c_{1k}^v} = \mathbb{P} : 2\boldsymbol{b}_k^e \quad \text{and} \quad \frac{\partial f_k}{\partial c_{1k}^v} = -\boldsymbol{\Gamma}_k : \frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial c_{1k}^v}, \tag{A.30}$$

$$\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial c_{2k}^v} = \mathbb{P} : 2\left(I_{1k}^e \boldsymbol{b}_k^e - \boldsymbol{b}_k^{e^2} \right) \quad \text{and} \quad \frac{\partial f_k}{\partial c_{2k}^v} = -\boldsymbol{\Gamma}_k : \frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial c_{2k}^v}, \tag{A.31}$$

$$\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial a_{k,q}} = \mathbf{0} \quad \text{and} \quad \frac{\partial f_k}{\partial a_{k,q}} = \nu_k \exp[\nu_k \dot{\gamma}_k] \left(\frac{\tau_k^v}{\hat{\tau}_k} \right)^q, \tag{A.32}$$

$$\frac{\partial \boldsymbol{\tau}_k^{v,\text{iso}}}{\partial \hat{\tau}_k} = \mathbf{0} \quad \text{and} \quad \frac{\partial f_k}{\partial \hat{\tau}_k} = -\nu_k \exp[\nu_k \dot{\gamma}_k] \frac{1}{\hat{\tau}_k} \sum_{q=1}^{n_\text{pow}} q a_{k,q} \left(\frac{\tau_k^v}{\hat{\tau}_k} \right)^q. \tag{A.33}$$

With the equations above, we complete all equations needed to construct the matrix $\boldsymbol{A}$ and vector $\boldsymbol{y}$ of Equation $(A.21)$ for each individual parameter to solve for the vector $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$ containing the updated derivatives at each time step.

REFERENCES

[1] F. Sidoroff, “Un modèle viscoélastique non linéaire avec configuration intermédiaire,” *Journal de Mécanique*, vol. 13, pp. 679–713, 1974.

[2] J. Lubliner, “A model of rubber viscoelasticity,” *Mechanics Research Communications*, vol. 12, no. 2, pp. 93–99, 1985.

[3] A. Lion, “A constitutive model for carbon black filled rubber: experimental investigations and mathematical representation,” *Continuum Mechanics and Thermodynamics*, vol. 8, pp. 153–169, 1996.

[4] G. A. Holzapfel and J. C. Simo, “A new viscoelastic constitutive model for continuous media at finite thermomechanical changes,” *International Journal of Solids and Structures*, vol. 33, no. 20-22, pp. 3019–3034, 1996.

[5] M. Kaliske and H. Rothert, “Formulation and implementation of three-dimensional viscoelasticity at small and finite strains,” *Computational Mechanics*, vol. 19, no. 3, pp. 228–239, 1997.

[6] S. Reese and S. Govindjee, “A theory of finite viscoelasticity and numerical aspects,” *International Journal of Solids and Structures*, vol. 35, no. 26-27, pp. 3455–3482, 1998.

[7] P. Haupt and K. Sedlan, “Viscoplasticity of elastomeric materials: experimental facts and constitutive modelling,” *Archive of Applied Mechanics*, vol. 71, pp. 89–109, 2001.

[8] D. Besdo and J. Ihlemann, “A phenomenological constitutive model for rubber-like materials and its numerical applications,” *International Journal of Plasticity*, vol. 19, no. 7, pp. 1019–1036, 2003.

[9] J. S. Bergström and M. C. Boyce, “Constitutive modeling of the large strain time-dependent behavior of elastomers,” *Journal of the Mechanics and Physics of Solids*, vol. 46, no. 5, pp. 931–954, 1998.

[10] S. Reese, “A micromechanically motivated material model for the thermo-viscoelastic material behaviour of rubber-like polymers,” *International Journal of Plasticity*, vol. 19, no. 7, pp. 909–940, 2003.

[11] C. Miehe and S. Göktepe, “A micro–macro approach to rubber-like materials. part II: The micro-sphere model of finite rubber viscoelasticity,” *Journal of the Mechanics and Physics of Solids*, vol. 53, no. 10, pp. 2231–2258, 2005.

[12] C. Linder, M. Tkachuk, and C. Miehe, “A micromechanically motivated diffusion-based transient network model and its incorporation into finite rubber viscoelasticity,” *Journal of the Mechanics and Physics of Solids*, vol. 59, no. 10, pp. 2134–2156, 2011.

[13] Y. Xiang, D. Zhong, S. Rudykh, H. Zhou, S. Qu, and W. Yang, “A review of physically based and thermodynamically based constitutive models for soft materials,” *Journal of Applied Mechanics*, vol. 87, no. 11, p. 110801, 2020.

[14] A. Kumar and O. Lopez-Pamies, “On the two-potential constitutive modeling of rubber viscoelastic materials,” *Comptes Rendus Mecanique*, vol. 344, no. 2, pp. 102–112, 2016.

[15] T. Kirchdoerfer and M. Ortiz, “Data-driven computational mechanics,” *Computer Methods in Applied Mechanics and Engineering*, vol. 304, pp. 81–101, 2016.

[16] R. Ibañez, D. Borzacchiello, J. V. Aguado, E. Abisset-Chavanne, E. Cueto, P. Ladeveze, and F. Chinesta, “Data-driven non-linear elasticity: constitutive manifold construction and problem discretization,” *Computational Mechanics*, vol. 60, pp. 813–826, 2017.

[17] R. Ibanez, E. Abisset-Chavanne, J. V. Aguado, D. Gonzalez, E. Cueto, and F. Chinesta, “A manifold learning approach to data-driven computational elasticity and inelasticity,” *Archives of Computational Methods in Engineering*, vol. 25, pp. 47–57, 2018.

[18] R. Eggersmann, T. Kirchdoerfer, S. Reese, L. Stainier, and M. Ortiz, “Model-free data-driven inelasticity,” *Computer Methods in Applied Mechanics and Engineering*, vol. 350, pp. 81–99, 2019.

[19] H. Salahshoor and M. Ortiz, “Model-free data-driven viscoelasticity in the frequency domain,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115657, 2023.

[20] M. Flaschel, S. Kumar, and L. De Lorenzis, “Unsupervised discovery of interpretable hyperelastic constitutive laws,” *Computer Methods in Applied Mechanics and Engineering*, vol. 381, p. 113852, 2021.

[21] M. Flaschel, S. Kumar, and L. De Lorenzis, “Discovering plasticity models without stress data,” *npj Computational Materials*, vol. 8, no. 1, p. 91, 2022.

[22] E. Marino, M. Flaschel, S. Kumar, and L. De Lorenzis, “Automated identification of linear viscoelastic constitutive laws with EUCLID,” *Mechanics of Materials*, vol. 181, p. 104643, 2023.

[23] M. Flaschel, S. Kumar, and L. De Lorenzis, “Automated discovery of generalized standard material models with EUCLID,” *Computer Methods in Applied Mechanics and Engineering*, vol. 405, p. 115867, 2023.

[24] K. Hornik, M. Stinchcombe, and H. White, “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks,” *Neural Networks*, vol. 3, no. 5, pp. 551–560, 1990.

[25] J. Ghaboussi, J. Garrett Jr, and X. Wu, “Knowledge-based modeling of material behavior with neural networks,” *Journal of Engineering Mechanics*, vol. 117, no. 1, pp. 132–153, 1991.

[26] 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*, vol. 357, p. 112594, 2019.

[27] M. B. Gorji, M. Mozaffar, J. N. Heidenreich, J. Cao, and D. Mohr, “On the potential of recurrent neural networks for modeling path dependent plasticity,” *Journal of the Mechanics and Physics of Solids*, vol. 143, p. 103972, 2020.

[28] G. Chen, “Recurrent neural networks (RNNs) learn the constitutive law of viscoelasticity,” *Computational Mechanics*, vol. 67, no. 3, pp. 1009–1019, 2021.

[29] J. Dornheim, L. Morand, H. J. Nallani, and D. Helm, “Neural networks for constitutive modeling: From universal function approximators to advanced models and the integration of physics,” *Archives of Computational Methods in Engineering*, vol. 31, no. 2, pp. 1097–1127, 2024.

[30] M. Rosenkranz, K. A. Kalina, J. Brummund, and M. Kästner, “A comparative study on different neural network architectures to model inelasticity,” *International Journal for Numerical Methods in Engineering*, vol. 124, no. 21, pp. 4802–4840, 2023.

[31] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[32] X. He and J.-S. Chen, “Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials,” *Computer Methods in Applied Mechanics and Engineering*, vol. 402, p. 115348, 2022.

[33] A. Danoun, E. Prulière, and Y. Chemisky, “Thermodynamically consistent recurrent neural networks to predict non linear behaviors of dissipative materials subjected to non-proportional loading paths,” *Mechanics of Materials*, vol. 173, p. 104436, 2022.

[34] E. Haghighat, S. Abouali, and R. Vaziri, “Constitutive model characterization and discovery using physics-informed deep learning,” *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105828, 2023.

[35] S. Rezaei, A. Moeineddin, and A. Harandi, “Learning solutions of thermodynamics-based nonlinear constitutive material models using physics-informed neural networks,” *Computational Mechanics*, pp. 1–34, 2024.

[36] P. Thakolkaran, A. Joshi, Y. Zheng, M. Flaschel, L. De Lorenzis, and S. Kumar, “NN-EUCLID: Deep-learning hyperelasticity without stress data,” *Journal of the Mechanics and Physics of Solids*, vol. 169, p. 105076, 2022.

[37] M. Rosenkranz, K. A. Kalina, J. Brummund, W. Sun, and M. Kästner, "Viscoelasticty with physics-augmented neural networks: Model formulation and training methods without prescribed internal variables," *Computational Mechanics*, pp. 1–23, 2024.

[38] F. As'ad and C. Farhat, "A mechanics-informed deep learning framework for data-driven nonlinear viscoelasticity," *Computer Methods in Applied Mechanics and Engineering*, vol. 417, p. 116463, 2023.

[39] V. Taç, M. K. Rausch, F. S. Costabal, and A. B. Tepole, "Data-driven anisotropic finite viscoelasticity using neural ordinary differential equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 411, p. 116046, 2023.

[40] K. Linka, M. Hillgärtner, K. P. Abdolazizi, R. C. Aydin, M. Itskov, and C. J. Cyron, "Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning," *Journal of Computational Physics*, vol. 429, p. 110010, 2021.

[41] K. Linka and E. Kuhl, "A new family of constitutive artificial neural networks towards automated model discovery," *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115731, 2023.

[42] L. M. Wang, K. Linka, and E. Kuhl, "Automated model discovery for muscle using constitutive recurrent neural networks," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 145, p. 106021, 2023.

[43] K. P. Abdolazizi, K. Linka, and C. J. Cyron, "Viscoelastic constitutive artificial neural networks (vCANNs)–a framework for data-driven anisotropic nonlinear finite viscoelasticity," *Journal of Computational Physics*, vol. 499, p. 112704, 2024.

[44] H. Holthusen, L. Lamm, T. Brepols, S. Reese, and E. Kuhl, "Theory and implementation of inelastic constitutive artificial neural networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 428, p. 117063, 2024.

[45] M. Hossain, D. K. Vu, and P. Steinmann, "Experimental study and numerical modelling of VHB 4910 polymer," *Computational Materials Science*, vol. 59, pp. 65–74, 2012.

[46] K. Srikanth, P. Sreejith, K. Arvind, K. Kannan, and M. Pandey, “An efficient mode-of-deformation dependent rate-type constitutive relation for multi-modal cyclic loading of elastomers,” *International Journal of Plasticity*, vol. 163, p. 103517, 2023.

[47] J. Zhou, L. Jiang, and R. E. Khayat, “A micro–macro constitutive model for finite-deformation viscoelasticity of elastomers with nonlinear viscosity,” *Journal of the Mechanics and Physics of Solids*, vol. 110, pp. 137–154, 2018.

[48] B. D. Coleman and M. E. Gurtin, “Thermodynamics with internal state variables,” *The Journal of Chemical Physics*, vol. 47, no. 2, pp. 597–613, 1967.

[49] J. C. Simo, “Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory,” *Computer Methods in Applied Mechanics and Engineering*, vol. 99, no. 1, pp. 61–112, 1992.

[50] H. Dal and M. Kaliske, “Bergström–Boyce model for nonlinear finite rubber viscoelasticity: theoretical aspects and algorithmic treatment for the FE method,” *Computational mechanics*, vol. 44, pp. 809–823, 2009.

[51] R. L. Taylor, “FEAP-A Finite Element Analysis Program,” 2014.

[52] M. Mooney, “A theory of large elastic deformation,” *Journal of Applied Physics*, vol. 11, no. 9, pp. 582–592, 1940.

[53] R. S. Rivlin, “Large elastic deformations of isotropic materials iv. further developments of the general theory,” *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and physical sciences*, vol. 241, no. 835, pp. 379–397, 1948.

[54] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[55] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*

(A. Moschitti, B. Pang, and W. Daelemans, eds.), (Doha, Qatar), pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[56] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, 1996.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] R. Mahnken and E. Stein, "Parameter identification for viscoplastic models based on analytical derivatives of a least-squares functional and stability investigations," *International Journal of Plasticity*, vol. 12, no. 4, pp. 451–479, 1996.

[59] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[60] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[61] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, (USA), p. 265–283, USENIX Association, 2016.