UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Martin Zlatić

Constitutive modelling of hyperelastic material behaviour with physics-augmented neural networks

DOCTORAL THESIS

Rijeka, 2025.

UNIVERSITY OF RIJEKA
FACULTY OF ENGINEERING

Martin Zlatić

Constitutive modelling of hyperelastic material behaviour with physics-augmented neural networks

DOCTORAL THESIS

Supervisor: Prof. dr. sc. Marko Čanađija

Rijeka, 2025.

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET

Martin Zlatić

Modeliranje procesa deformiranja hiperelastičnih materijala pomoću fizikalno proširenih neuronskih mreža

DOKTORSKI RAD

Mentor: Prof. dr.sc. Marko Čanađija

Rijeka, 2025.

Supervisor: Prof. dr. sc Marko Čanađija

This doctoral thesis was defended on _____ at the Faculty of Engineering, University of Rijeka, in front of the Committee consisting of:

1. Prof. dr. sc. Marino Brčić (Committee Chair, Faculty of Engineering, University of Rijeka)
2. Prof. Dr. Jörn Mosler (Institut für Mechanik, Fakultät Machinenbau, Technische Universität Dortmund, Germany)
3. Prof. dr. sc. Zdenko Tonković (Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb)

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Prof. Marko Čanađija D.Sc., for his continued support and countless discussions during this journey. Thank you for all the advice and time you have given me during these last few years.

I would also like to gratefully acknowledge the support of the Croatian Science Foundation under the project IP-2019-04-4703 and the University of Rijeka under the project uniri-iskusni-tehnic-23-37.

Many thanks to my colleagues Sanjin, Damjan and Sandra for making this journey more fun and for the numerous conversations over the years. Without you it would not have been as enjoyable.

I would like to thank my parents for their patience, support and encouragement during these years.

Lastly, I would like to thank my beloved Ana Maria for her many kind words. I look forward to sharing many more unforgettable moments by your side.

Abstract

Machine learning has gained an important place both in daily interactions and in scientific development. It has a wide range of applications and many different approaches which can be used to solve various problems. Neural networks are one of these approaches and are a potent regression tool which is applied in a large number of scientific fields. In recent years it has gained popularity in the field of computational mechanics as an alternative material model replacing (or complementing) existing analytical models. These neural network models are useful for describing complex relationships that existing models can not fully describe. They are also used as a general model that can be applied to a wide range of materials. This thesis places a focus on the latter, with the aim of creating a general material model for hyperelastic materials such as rubber. With analytical models there is a need to fit different models (Neo-Hookean, Mooney-Rivlin, Ogden, Yeoh, Gent, etc.) to experimental data and use the one that approximates the data best. Each analytical model has some advantages compared to the others as well as drawbacks, and the choice of the model also depends on what the application of the model will be. By using a neural network as a model the goal is to obtain a single model that has no drawbacks and can function for any material. This also speeds up the material modelling process as only one single model is generated instead of fitting multiple models and choosing between them. In this thesis a detailed investigation is made between different modelling strategies. Firstly, a neural network that predicts stress from strain is developed and is a representative of the most common approach in neural network modelling. Afterwards, a more general model using the same neural network architecture is created that is similar to conventional invariant models so that it predicts the energy while taking the invariants of the right Cauchy-Green deformation tensor as inputs. It was demonstrated that it can be used as a direct replacement for the invariant models by reusing the existing framework for the numerical implementation of invariant based models. Secondly, a significant improvement to the neural network architecture is done through the implementation of a custom activation function suited for modelling of hyperelastic behaviour. It is accompanied by the implementation of certain conditions from solid mechanics such as objectivity, thermodynamic consistency, normalisation of energy, non-negativity of energy, normalisation of stress and polyconvexity. All of these improvements have created a neural network model, referred to as LINEXP-PANN, that can capture a wider range of material behaviours with a significantly reduced dataset size compared to the previous simpler neural network models. The LINEXP-PANN model was used for modelling damage in rubber-like materials which is known as the Mullins effect. A new modelling strategy was developed for modelling the Mullins effect using neural networks where certain weights are reused and shared within the neural network. The majority of the work is based on the assumption of incompressibility, a common assumption when modelling rubber-like materials. The LINEXP-PANN model was also extended to compressible behaviour confirming the general modelling capabilities of the model. Finally, a comparison is made between the LINEXP-PANN model and another data-driven method called Data-Driven Computational Mechanics (DDCM) where the advantages of the LINEXP-PANN model are demonstrated.

Keywords: hyperelasticity, physics-augmented neural networks, material modelling

Prošireni sažetak

Strojno učenje je postalo dio naše svakodnevice i dio znanstvenoga razvoja. Ima široko područje primjene i postoje mnogi različiti pristupi koje se može primijeniti za rješavanje problema. Neuronske mreže su jedan od tih pristupa i moćni su regresijski alat koji se primjenjuje u mnoštvu znanstvenih polja. Sa porastom primjene strojnog učenja u inženjerstvu, neuronske mreže postaju sve popularniji alternativni materijalni model koji zamjenjuje (ili nadopunja) postojeće analitičke modele. Materijalni modeli temeljeni na neuronskim mrežama su korisni za opisivanje složenih ponašanja koje postojeći analitički modeli ne mogu u potpunosti opisati. Također se koriste i kao opći modeli koji mogu opisati širi raspon materijala. U ovom doktorskome radu se pozornost stavlja na drugo svojstvo primjene gdje se želi stvoriti opći materijalni model za hiperelastične materijale sa naglaskom na materijale slične gume. Kada se koriste analitički modeli potrebno je prilagoditi više modela (Neo-Hookean, Mooney-Rivlin, Ogden, Yeoh, Gent, itd.) na eksperimentalne podatke i odabrati onaj model koji najbolje opisuje podatke. Svaki analitički model ima neke prednosti u odnosu a druge modele, no i nedostatke u odnosu na njih, te se mora odabrati model ovisno o tome koji najviše odgovara određenoj primjeni. Koristeći neuronsku mrežu kao model cilj je dobiti jedinstveni model koji nema nedostataka i može se koristiti za bilo koji materijal. Ovo također ubrzava proces materijalnoga modeliranja jer je potrebno prilagoditi samo jedan model umjesto više njih između kojih se mora birati. Prvi korak u radu je bio razvoj neuronske mreže koja predviđa naprezanje iz deformacije i predstavnik je uobičajenoga načina modeliranja neuronskim mrežama. Potom je stvoren općenitiji model temeljen na istoj arhitekturi neuronske mreže koji je sličan invarijantnim modelima jer predviđa energiju i koristi invarijante desnog Cauchy-Greenova tenzora deformiranja kao ulazne podatke. Pokazalo se da ovaj model može biti korišten i kao jednostavna zamjena za klasične invarijantne modele jer koristi već postojeće rutine za računarsku implementaciju invarijantnih modela. Drugi korak čine unaprijeđenja samoj arhitekturi neuronske mreže od kojih je najznačajnija implementacija vlastite aktivacijske funkcije primjerene modeliranju hiperelastičnoga ponašanja. Korištenje vlastite aktivacijske funkcije omogućava ispunjenje određenih uvjeta iz mehanike čvrstoga tijela poput objektivnosti, termodinamičke konzistencije, normalizacije energije, ne-negativnost energije, normalizacija naprezanja i polikonveksnost. Promjene u arhitekturi i ispunjenje ovih uvjeta su vodile stvaranju modela temeljenog na neuronskoj mreži kojem se nadjenulo ime LINEXP-PANN (Linear EXPonential Physics-Augmented Neural Network, hrv. linearno eksponencijalna fizikalno proširena neuronska mreža). Ovaj model može opisati široki raspon materijalnih ponašanja koristeći znatno manju količinu podataka za treniranje neuronske mreže u usporedbi sa jednostavnijim modelima predstavljenim u prvome koraku. Prethodno spomenuti LINEXP-PANN model je korišten i za u modeliranju oštećenja kod gumenih materijala koji je poznat kao Mullinsov efekt. Razvijen je novi pristup modeliranju Mullinsova efekta neuronskim mrežama koji je specifičan jer su određene težine dijeljene u neuronskoj mreži među njenim dijelovima. Većina ovoga rada je temeljena na pretpostavci nestlačivosti koja je česta u kontekstu modeliranja gumenih materijala. Novorazvijeni LINEXP-PANN model je proširen i za modeliranje stlačivog hiperelastičnog ponašanja čime se dodatno potvrđuju prilagodljivost i općenitost modela. Naposljetku je napravljena usporedba između LINEXP-PANN modela i jedne alternative metode temeljene na podacima Data-Driven Computational Mechanics (DDCM, hrv. po-

dacima pogonjena računalna mehanika) te su prikazane prednosti LINEXP-PANN modela.
Ključne riječi: hiperelastičnost, fizikalno proširene neuronske mreže, materijalno modeliranje

Contents

1 Introduction

In the past several decades, and more intensely in the past 10 to 15 years, machine learning has gained a lot of attention and is being applied to a large variety of topics. Although the application of machine learning started to gain traction following the work of [28] where the possibility of using neural networks as general approximator for any function was proposed, there was earlier work done on applying machine learning techniques to solid and structural mechanics [71], although it was curtailed by the relatively modest hardware capabilities (compared to modern day CPUs and GPUs). As hardware became more potent the applications of neural networks to material modelling expanded, with work like that in [62] dealing with the application of neural networks to more complex non-linear behaviour. Now, neural networks (and other machine learning techniques) are applied in every aspect of engineering, including mechanics.

As the application of neural networks (NNs) became more widespread it has also been adopted in the field of constitutive modelling of materials. Neural networks could be simply explained as sophisticated regression algorithms that describe a relation between an independent and dependent quantity that would otherwise be difficult to describe. Conventional models have variables in them that have physical significance, e.g. Young's modulus in linear elasticity, while the alternative neural network models likely have a large amount of variables (usually in the hundreds or thousands in relation to solid mechanics) that most often do not have any physical significance but simply statistically relate the independent variables (inputs) to the dependent variables (outputs). Given the general approximative power of neural networks they are a potential candidate model in many areas of constitutive modelling when the relations are not simple and conventional models have difficulties capturing the material behaviour. A very early example of the application of NNs to model complex relations can be found in [20] where NNs were used to model the behaviour of concrete under cyclic loading. When describing metal plasticity in [29], proper orthogonal decomposition was used to prepare the data and then trained an NN model on this reduced dataset and the final model successfully represented the material behaviour. Another application of NNs to metal plasticity is shown in [17] where recurrent neural networks are used to describe the behaviour of steel under tension-compression loading cycles. Neural networks can be used as surrogate models to replace complex simulations such as liver deformation under point load during surgery [48] or the thoracic aorta [43] at a fraction of the computational cost. This was achieved by training the NNs on finite element simulations. On the other end of the size scale, NNs have been used to predict properties of carbon nanotubes which include non-local effects that arise from the small scale of the problem [8, 41]. Neural networks are also attractive due to their ability to filter out noises as shown in [19] where recurrent NNs were used for describing material behaviour with noisy data. In general other machine learning techniques are also good candidates for application on noisy data, another example is given in [30] where a Bayesian framework is used to recover a conventional model from noisy data. Another more general application of neural network can be to directly generate stiffness

matrices of finite elements rather than just describing material behaviour. This was done in [32] where finite element stiffness matrices that were obtained from NNs outperformed conventional finite elements in convergence. Similar work in a multi-scale approach was done in [10] using support vector regression in place of NNs.

This thesis deals with the application of NNs to hyperelastic behaviour (hyperelasticity) which are materials that undergo large strains and are defined by a strain energy density function, the most common application is modelling of rubber and rubber-like materials. The first work where a conventional NN was applied to model the strain energy density function in place of conventional models is [62]. Neural network were used to model the strain energy density function for a one dimensional case in the micro-sphere model in [75], with an extension to inelastic behaviour using recurrent NNs. In [69] the objective function of the NN was augmented in such a manner to consider material frame indifference while predicting stress directly from strain without modelling a strain energy density function. In the work of [73] conventional NNs were applied to modelling adiabatic rubber thermoelasticity both by directly predicting the stress from strain and by approximating the strain energy density function demonstrating the difference in the two approaches and that it is more advantageous to approximate the strain energy density function with an NN. A very useful concept introduced in [14] is the concept of Sobolev training for neural networks. Sobolev spaces are the spaces of derivatives of a function, thus Sobolev training implies that the NN is trained on its derivatives. In the original paper it is tested on a regression benchmark. In the work [68] training on the Sobolev space is introduced for elastoplasticity. This is a useful training strategy that is adopted by many other works since it enables training on stress data while the strain energy is predicted by the NN. One of the first works which used Sobolev training for an NN to capture hyperelastic behaviour is [45] which utilised a conventional NN design. This work was later followed by [46] where a specially constructed NN was designed inspired by existing hyperelastic material models. This approach is referred to as the *Constitutive Artificial Neural Network* (CANN). The CANN approach offers good performance but it is not as flexible as a NN so further work was done by exploring way to augment conventional NN architectures to incorporate restrictions normally required from conventional hyperelastic models. In the work of [58] physically meaningful loss measures were introduced into the NN so that the NN could better describe the physical problem at hand, this approach is widely referred to as *Physics Informed Neural Networks* (PINNs). In the work of [3] the authors introduced additional restrictions from solid mechanics, such as convexity, to conventional NNs that further improved the approximation qualities of NNs, they refer to their approach as *mechanics informed neural networks*. Further refined models with more restrictions were proposed in [44] summarizing the possible restrictions and their implementation strategies with the proposed term *Physics-Augmented Nueral Networks* (PANNs) where conventional NNs were adapted to effectively fulfil requirements stemming from solid mechanics. This approach and variations thereof was successfully adopted for modelling of isothermal compressible hyperelasticity [44], anisotropic hyperelasticity [38], in a multi-physics setting with magneto active

polymers [33], isotropic Mullins type damage [74], parametrized hyperelasticity [39], and viscoelasticity [59]. All the previously mentioned works deal with the implementation of neural networks to distinct material behaviours but none deal with the application to thermoelastic behaviour or materials with damage. These behaviours are specially interesting for rubber-like materials since during loading they exhibit an interesting property where their temperature firstly decreases but after a certain point begins to increase [23]. Another common behaviour of rubber materials with carbon fillers is the occurrence of damage so that the unloading path is not the same as the loading path, this is known as the Mullins effect [49]. Additionally, all of the previously mentioned works use standard activation functions that, although certainly viable, are not commonly used in describing hyperelastic behaviour and a better candidate function might exist. Therefore, two hypotheses are brought forward in this thesis:

(i)

General neural network based hyperelastic material models can be successfully applied in specific phenomena such as thermoelasticity or damage modelling.

(ii)

The introduction of specialised activation functions renders a more compact neural network material model.

In order to fulfil these goals an investigation is made starting from conventional methods of NN modelling used in existing work to discern the importance of different NN modelling strategies. New concepts and solutions are introduced and compared in order to obtain a NN model with superior accuracy and general performance compared to existing models.

A final note should be made that although neural networks are often referred to as *data driven* models or approaches, there is another approach to simulations that completely relies on data (which is obtained artificially or experimentally) and does not even have a model *per se* making it a ”true” data-driven model-free approach. This was first proposed in [36] with an example on linearly elastic trusses, the name for this family of solvers is taken from this work as *Data-Driven Computational Mechanics* (DDCM). It was later adopted to different applications such as fracture [11], hyperelasticity [57], inelastic behaviour [18], molecular dynamics [6], multiscale modelling [34], noisy data [37], frequency domain data [60] and other areas. A comparison is made in the thesis between the performance of NNs and DDCM.

A brief overview of the contents of this thesis is given as follows:

- In Section 2 a brief overview of hyperelasticity is given with a basic overview of the kinematics, conservation laws and basic constitutive relations. This section serves as the basis where all the background from mechanics that is needed to follow this work is given. Similarly, Section 3.1 contains the necessary background for following the work done related to the construction of neural networks, their training and implementation.

- In Section 3, a brief introduction is given about feedforward neural networks and their training procedure. Various types of activation functions are described which are usually used in neural networks followed by the application of conventional neural networks architectures on hyperelasticity. Also, in the same section the application of neural networks to adiabatic thermoelasticity is investigated since the difference between it and isothermal hyperelasticity are very small, giving an insight into the capabilities of conventional neural network architectures for capturing small differences in the same material type behaviour.

- In Section 4, drawing on insights from Subsection 3.3 and the literature, a *physics-augmented neural network* is presented. This is a neural network which fulfils certain condition from solid mechanics by its construction, rendering a neural network that can correctly capture general material behaviour. This section is the central piece of the thesis where the construction of a general neural network for modelling hyperelasticity is presented in detail and a new activation function is introduced as well as the concept of modelling incompressible hyperelasticity with neural networks.

- In Section 5 a framework for modelling Mullins-type simple isotropic damage is presented. The general neural network model from Section 4 is used and compared against several variations of the model as well as against another neural network model from the literature. Also, a novel strategy for modelling materials with damage is introduced where components of the network are reused in order to fulfil a modelling requirement from solid mechanics.

- Finally, in Section 6 the general neural network model is compared with a different novel approach called data driven computational mechanics. It is a model-free approach that directly uses data gathered from simulations/experiments to perform an analysis. In this chapter the viability of neural networks compared to the other most popular data based approach is investigated.

2 Brief Overview of Hyperelasticity

Within this chapter a brief overview is provided of the necessary fundamentals of hyperelastic material behaviour. Hyperelastic materials are those that are described by a strain energy (or stored energy) function, which is in turn a function of the deformation gradient **F**, both these terms are discussed later in this section. If there are no losses during the deformation process (i.e. if the deformation process is perfectly reversible) such a material may be referred to as a perfectly elastic Cauchy material. Rubber-like materials are an example of a hyperelastic material as their behaviour is often characterised by a strain energy function. This chapter encompasses the background needed to understand the presented work and introduces much of the used notation. For a more detailed overview of the subject the reader is referred to the works of [9, 27, 54, 66] and further in relation to its application in the finite element method in the works of [5, 15, 70].

2.1 Kinematics

Consider a body in a three-dimensional Euclidean space $\mathbb{R}^3$. A point on the body can be defined in its initial configuration $\mathbf{X} = X_a\mathbf{E}_a$ at a time t_0, also called the material or Lagrangian configuration, or in the current configuration, also called the spatial configuration, $\mathbf{x} = x_a\mathbf{e}_a$ at a time t, with $a = 1, 2, 3$, $\mathbf{E}$ and $\mathbf{e}$ the basis vectors such that $\mathbf{E}_i \times \mathbf{E}_j = \mathbf{E}_k$ and $\mathbf{e}_i \times \mathbf{e}_j = \mathbf{e}_k$. Note that the uppercase letters usually denote a quantity in the initial (material or Lagrangian) configuration and lowercase letters usually denote the quantity in the current (spatial) configuration. This also applies to mathematical operators such as the gradient and divergence with Grad or Div denoting the gradient or divergence with respect to the initial configuration and grad or div with respect to the current configuration.

The deformation gradient **F** is introduced as the operator which maps the line element from the initial to the current configuration as

$$\mathrm{d}\mathbf{x} = \mathbf{F}\mathrm{d}\mathbf{X}, \tag{2.1}$$

from which the deformation gradient can further be defined as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial x_i}{\partial X_j} = F_{ij}\mathbf{e}_i \otimes \mathbf{E}_j. \tag{2.2}$$

The deformation gradient serves as a basis for all further measures of deformation introduced in this work. Note that the deformation gradient is also depicted as the transformation from the initial to the current configuration as $\boldsymbol{\varphi}(\mathbf{X}, t)$. A general motion described by **F** can be separated into pure stretch, which can be defined by the right (**U**) and left (**V**) stretch tensors, and pure rotation which is defined by the rotation tensor **R**:

$$\mathbf{F} = \mathbf{V}\mathbf{R} = \mathbf{R}\mathbf{U}, \tag{2.3}$$

with $\mathbf{R}$ being a proper orthogonal tensor, i.e. $\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1}$. Also, $\mathbf{U}$ and $\mathbf{V}$ are both symmetric tensors. Furthermore, the stretch tensors can be decomposed using spectral decomposition to their eigenvalues λ_a such that

$$\mathbf{U} = \sum_{a=1}^{3} \lambda_a \mathbf{N}_a \otimes \mathbf{N}_a, \tag{2.4a}$$

$$\mathbf{V} = \sum_{a=1}^{3} \lambda_a \mathbf{n}_a \otimes \mathbf{n}_a, \tag{2.4b}$$

where $\mathbf{N}_a$ and $\mathbf{n}_a$ are the eigenvectors of the right and left stretch tensors. The deformation gradient can also be decomposed in the following form

$$\mathbf{F} = \sum_{a=1}^{3} \lambda_a \mathbf{n}_a \otimes \mathbf{N}_a, \tag{2.5}$$

with the previously mentioned principal stretches λ_a and the eigenvectors $\mathbf{N}_a$ and $\mathbf{n}_a$. Having defined the deformation gradient, a summary of the transformations from the initial configuration to the current one can be given:

(i) Mapping of line elements

As already presented in Eq. (2.1) the mapping of line elements is done by $\mathrm{d}\mathbf{x} = \mathbf{F}\mathrm{d}\mathbf{X}$.

(ii) Mapping of surface elements

The mapping between the initial and current configuration is given by Nanson's formula:

$$\mathrm{d}\boldsymbol{a} = J\mathbf{F}^{-T}\mathrm{d}\boldsymbol{A} = \mathrm{cof}(\mathbf{F})\mathrm{d}\boldsymbol{A}, \tag{2.6}$$

with $\mathrm{d}\boldsymbol{a} = \mathbf{n}\mathrm{d}a$ and $\mathrm{d}\boldsymbol{A} = \mathbf{N}\mathrm{d}A$, and where J is the volume ratio of the current configuration with respect to the initial one, i.e. the relative change of volume during deformation defined as the determinant of the deformation gradient

$$\det(\mathbf{F}) = J = \lambda_1\lambda_2\lambda_3. \tag{2.7}$$

(iii) Mapping of volume elements

The mapping of volume elements is simply related by the Jacobian between the volumes in the initial and current configuration:

$$\mathrm{d}v = J\mathrm{d}V, \tag{2.8}$$

with $\mathrm{d}v$ being the infinitesimal volume element in the current configuration defined by the triple scalar product $\mathrm{d}\mathbf{x}_1 \cdot (\mathrm{d}\mathbf{x}_2 \times \mathrm{d}\mathbf{x}_3)$, and $\mathrm{d}V$ the infinitesimal volume element in the initial configuration defined by the triple scalar product $\mathrm{d}\mathbf{X}_1 \cdot (\mathrm{d}\mathbf{X}_2 \times \mathrm{d}\mathbf{X}_3)$.

Another important deformation measure in the material configuration is the right Cauchy-Green deformation tensor **C** defined as

$$\mathbf{C} = \mathbf{F}^T\mathbf{F} = \mathbf{U}^T\mathbf{R}^T\mathbf{R}\mathbf{U} = \mathbf{U}^2 = \sum_{a=1}^{3} \lambda_a^2 \mathbf{N}_a \otimes \mathbf{N}_a, \tag{2.9}$$

with its counterpart in the current configuration being the left Cauchy-Green deformation tensor **B** defined as

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T = \mathbf{V}\mathbf{R}\mathbf{R}^T\mathbf{V}^T = \mathbf{V}^2 = \sum_{a=1}^{3} \lambda_a^2 \mathbf{n}_a \otimes \mathbf{n}_a. \tag{2.10}$$

Additionally, the the Green-Lagrange strain tensor can be expressed as

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right). \tag{2.11}$$

A convenient manner to express any state of deformation regardless of the change of basis would be to express it in terms of principal invariants. Later in this work invariants are going to be the base upon which general neural network material models will be formed. Using the Cayley-Hamilton theorem a matrix polynomial can be constructed such that it is equal to the zero matrix, i.e. $p_{\mathbf{A}}(\mathbf{A}) = \mathbf{0}$ where $p_{\mathbf{A}}(\mathbf{A})$ is the matrix polynomial and **A** is an invertible square matrix. The matrix polynomial can be written as

$$p_{\mathbf{A}}(\mathbf{A}) = \mathbf{A}^n + c_{n-1}\mathbf{A}^{n-1} + ... + c_1\mathbf{A} + c_0 I_n, \tag{2.12}$$

where c_i $(i = 0, ..., n)$ are the coefficients of the polynomial, I_n is the identity matrix of rank n and n is the rank of the matrix **A**. Having in mind that $\mathbf{C} \in \mathbb{R}^{3\times3}$ it follows from the Cayley-Hamilton theorem that

$$\mathbf{C}^3 + I_1\mathbf{C}^2 + I_2\mathbf{C} + I_3\mathbf{I} = 0, \tag{2.13}$$

where the coefficients c_2, c_1, c_0 are now replaced with the notation I_1, I_2, I_3 and called the principal invariants. The coefficients of the matrix polynomial (i.e. the principal invariants) are obtained in terms of exponential Bell polynomials as:

$$I_1 = \text{tr}(\mathbf{C}), \quad I_2 = \frac{1}{2}\big(\text{tr}(\mathbf{C})^2 - \text{tr}(\mathbf{C}^2)\big), \quad I_3 = \det(\mathbf{C}). \tag{2.14}$$

The invariants can further be interpreted in terms of the eigenvalues of **C** as

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2, \quad I_2 = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_1^2\lambda_3^2, \quad I_3 = \lambda_1^2\lambda_2^2\lambda_3^2 = J^2. \tag{2.15}$$

The displacement $\boldsymbol{u}$ of a point on a body in the current configuration can be defined as

$$\boldsymbol{u} = \mathbf{x} - \mathbf{X}, \tag{2.16}$$

and it does not change in the initial configuration as the displacement is defined in the same manner $\mathbf{U} = \mathbf{x} - \mathbf{X}$. The velocity and acceleration are further defined as material time derivatives $\frac{\mathrm{D}}{\mathrm{D}t}(\bullet)$ (i.e. a time derivative holding the reference $\mathbf{X}$ fixed) of the displacement

$$\mathbf{v} = \mathbf{V} = \frac{\mathrm{D}}{\mathrm{D}t}(\mathbf{x} - \mathbf{X}) = \frac{\partial \mathbf{x}}{\partial t} = \dot{\mathbf{x}}, \quad \boldsymbol{a} = \frac{\partial \mathbf{v}}{\partial t} = \frac{\partial^2 \mathbf{x}}{\partial t^2} = \ddot{\mathbf{x}}, \tag{2.17}$$

note that the time derivative of $\mathbf{X}$ vanishes since the position in the initial configuration is held fixed, i.e. it is constant. The material time derivative is also denoted with a dot above the derived quantity as $\dot{(\bullet)}$. The gradient of the velocity $\boldsymbol{\ell}$ is defined as

$$\boldsymbol{\ell} = \mathrm{grad}\mathbf{v}, \tag{2.18}$$

and by deriving the deformation gradient $\mathbf{F}$ with respect to time as

$$\dot{\mathbf{F}} = \frac{\partial}{\partial t}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{X}}\right) = \frac{\partial}{\partial \mathbf{X}}\left(\frac{\partial \mathbf{x}}{\partial t}\right) = \frac{\partial \mathbf{v}}{\partial \mathbf{X}} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \mathrm{grad}\mathbf{v}\mathbf{F} = \boldsymbol{\ell}\mathbf{F}, \tag{2.19}$$

the alternate definition of the velocity gradient can be formulated as

$$\boldsymbol{\ell} = \dot{\mathbf{F}}\mathbf{F}^{-1}. \tag{2.20}$$

The velocity gradient can be further decomposed to its symmetric and antisymmetric parts as

$$\boldsymbol{\ell} = \mathbf{d} + \mathbf{w}, \quad \mathbf{d} = \frac{1}{2}\left(\boldsymbol{\ell} + \boldsymbol{\ell}^T\right), \quad \mathbf{w} = \frac{1}{2}\left(\boldsymbol{\ell} - \boldsymbol{\ell}^T\right), \tag{2.21}$$

where $\mathbf{d}$ is the symmetric part and is called the rate of deformation tensor, and $\mathbf{w}$ is the rate of rotation tensor.

2.2 Stresses

Stress is defined as a force per unit area. In a more general sense, it can be expressed as a linear function of the normal unit vector acting on the boundary of a body or a surface area. The Cauchy stress tensor $\boldsymbol{\sigma}$ relates the traction vector $\mathbf{t}$ to the outward normal unit vector $\mathbf{n}$ in the current configuration, whereas the first Piola-Kirchhoff stress tensor $\mathbf{P}$ relates the traction vector $\mathbf{T}$ to the outward normal unit vector $\mathbf{N}$ in the initial configuration. Note that both $\mathbf{t}$ and $\mathbf{T}$ point in the same direction as they describe the same external load. This can be defined as

$$\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}, \quad \mathbf{T} = \mathbf{P}\mathbf{N}. \tag{2.22}$$

The traction vectors describe the same load so the following holds:

$$\mathbf{t}da = \mathbf{T}dA, \tag{2.23}$$

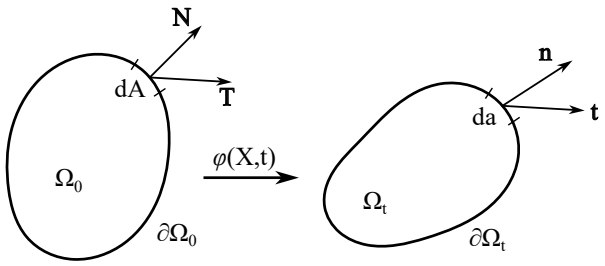


Figure 2.1: Initial and current configurations, depiction of the normal vectors **n** and **N**, traction vectors **t** and **T**, and surface areas dA and da.

so using the previous equations the relation between Cauchy and first Piola-Kirchhoff stress can be given as

$$\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T}. \tag{2.24}$$

Another useful stress tensor that is the Kirchhoff stress tensor $\boldsymbol{\tau}$ which is tied to Cauchy's stress tensor by the volume ratio J as

$$\boldsymbol{\tau} = J\boldsymbol{\sigma}, \tag{2.25}$$

and additionally the second Piola-Kirchhoff stress tensor is introduced and can be obtained by the pull-back operation on $\boldsymbol{\tau}$ as

$$\mathbf{S} = \mathbf{F}^{-1}\boldsymbol{\tau}\mathbf{F}^{-T} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} = \mathbf{F}^{-1}\mathbf{P}. \tag{2.26}$$

This is a fictitious tensor with no physical interpretation, however it is often used in the definition of material behaviour as it leads to simpler operations and will prove useful when defining a neural network model.

Incompressible behaviour If a material behaves in such a manner that during deformation there is no volume change, it can be classified as an incompressible material. Following from [66], section 30, in the case of material incompressibility an internal constraint in the form of $\det(\mathbf{C}) = 1$ arises and the stress can thus be computed up to an extra stress component p. In this case the Cauchy stress can be expressed as

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_E - p\mathbf{I}, \tag{2.27}$$

where $\boldsymbol{\sigma}_E$ is defined from a constitutive relation from the strain energy function which is discussed later in the next subchapter. See and compare with Eq. (2.84) and the accompanying text.

2.3 Balance Principles

Conservations and balance principles serve as the building blocks upon which behaviour of continua is described. They do not depend on any specific material behaviour, but are valid in general for any behaviour reduced to a continuum. In this section several principles are

covered, namely the conservation of mass, balance of linear and angular momentum, balance of mechanical energy, the entropy inequality, and the Clausius-Duhem inequality. Some balance principles include the material time derivative which will in general be defined with a dot above the derived quantity, i.e. $(\dot{\bullet})$.

Conservation of mass As a body deforms it changes its shape and potentially changes its volume. However, its mass remains unchanged. Considering an infinitesimal mass $\mathrm{d}m$ it can be expressed in the initial or current configuration

$$\mathrm{d}m = \rho_0 \mathrm{d}V = \rho \mathrm{d}v, \tag{2.28}$$

with ρ_0 and ρ the densities in the initial and current configuration. Given that mass doesn't change during deformation, the conservation of mass can be expressed using Eq. (2.8) as

$$\int_{\Omega_0} \big(\rho_0 - J\rho\big) \mathrm{d}V = 0, \tag{2.29}$$

in the global form and as

$$\rho_0 - J\rho = 0 \rightarrow \rho_0 = J\rho \tag{2.30}$$

in the local form giving the relation between the densities in the current and reference configuration.

Balance of Linear Momentum The balance of linear momentum states that the change of momentum $\mathbf{L}$ is equal to the externally applied force $\boldsymbol{f}$. Linear momentum is defined as

$$\mathbf{L} = \int_{\Omega} \rho \mathbf{v} \mathrm{d}v, \tag{2.31}$$

with Ω being the domain in the current configuration, ρ the density and $\mathbf{v}$ the velocity. The external force $\boldsymbol{f}$ is defined as

$$\boldsymbol{f} = \int_{\Omega} \mathbf{b} \mathrm{d}v + \int_{\partial\Omega} \mathbf{t} \mathrm{d}a, \tag{2.32}$$

with $\mathbf{b}$ the body (volume) forces that act in the domain Ω in the current configuration, and $\mathbf{t}$ the traction vector defined earlier in Eq. $(2.22)_1$ that is acting on the boundary $\partial\Omega$.

The balance of linear momentum $\dot{L} = \boldsymbol{f}$ can be written as

$$\int_{\Omega} \rho \dot{\mathbf{v}} \mathrm{d}v = \int_{\Omega} \mathbf{b} \mathrm{d}v + \int_{\partial\Omega} \mathbf{t} \mathrm{d}a. \tag{2.33}$$

Recalling Eq. $(2.22)_1$ and the divergence theorem the second term on the right hand side of Eq. (2.33) can be expressed as

$$\int_{\partial\Omega} \mathbf{t} \mathrm{d}a = \int_{\partial\Omega} \boldsymbol{\sigma} \mathbf{n} \mathrm{d}a = \int_{\Omega} \mathrm{div}\boldsymbol{\sigma} \mathrm{d}v, \tag{2.34}$$

leading to the balance of linear momentum in its global form in the current configuration as

$$\int_{\Omega} \big(\mathrm{div}\boldsymbol{\sigma} + \mathbf{b} - \rho\dot{\mathbf{v}}\big)\mathrm{d}v = 0, \tag{2.35}$$

which is also known as Cauchy's first equation of motion, and also to the equation of motion in its local form

$$\mathrm{div}\boldsymbol{\sigma} + \mathbf{b} - \rho\dot{\mathbf{v}} = 0. \tag{2.36}$$

The balance of linear momentum can be expressed in the material configuration by recalling Eqs. (2.8), (2.23), (2.24) and $(2.30)_2$. Each term in Eq. (2.33) can be transformed to the material configuration:

$$\int_{\Omega} \mathrm{div}\boldsymbol{\sigma}\mathrm{d}v = \int_{\partial\Omega} \boldsymbol{\sigma}\mathbf{n}\mathrm{d}a = \int_{\partial\Omega_0} \mathbf{P}\mathbf{N}\mathrm{d}A = \int_{\Omega_0} \mathrm{Div}\mathbf{P}\mathrm{d}V \tag{2.37a}$$

$$\int_{\Omega} \mathbf{b}\mathrm{d}v = \int_{\Omega_0} \mathbf{b}J\mathrm{d}V = \int_{\Omega_0} \mathbf{B}\mathrm{d}V, \tag{2.37b}$$

$$\int_{\Omega} \rho\dot{\mathbf{v}}\mathrm{d}v = \int_{\Omega_0} \rho_0\dot{\mathbf{v}}\mathrm{d}V, \tag{2.37c}$$

with the final global form of the balance of linear momentum in the material configuration

$$\int_{\Omega_0} \big(\mathrm{Div}\mathbf{P} + \mathbf{B} - \rho_0\dot{\mathbf{v}}\big)\mathrm{d}V = 0, \tag{2.38}$$

and in this equation $\mathbf{B}$ denotes the vector of body forces in the initial configuration, not to be confused with the left Cauchy-Green stretch tensor from Eq. (2.10).

Balance of angular momentum The balance of angular momentum states that the change of angular momentum is equal to the moment $\mathbf{M}$ of force $\boldsymbol{f}$ about $\mathbf{x}_0$. The angular momentum $\mathbf{J}$ relative to a fixed point $\mathbf{x}_0$ can be expressed using the vector (cross) product $\times$ as

$$\mathbf{J} = \int_{\Omega} \mathbf{r} \times \rho\mathbf{v}\mathrm{d}v, \tag{2.39}$$

with $\mathbf{r} = \mathbf{x} - \mathbf{x}_0$, and the moment $\mathbf{M}$ as

$$\mathbf{M} = \int_{\Omega} \mathbf{r} \times \mathbf{b}\mathrm{d}v + \int_{\partial\Omega} \mathbf{r} \times \mathbf{t}\mathrm{d}a. \tag{2.40}$$

The balance of angular momentum $\dot{\mathbf{J}} = \mathbf{M}$ follows as

$$\int_{\Omega} \mathbf{r} \times \rho\dot{\mathbf{v}}\mathrm{d}v = \int_{\Omega} \mathbf{r} \times \mathbf{b}\mathrm{d}v + \int_{\partial\Omega} \mathbf{r} \times \mathbf{t}\mathrm{d}a, \tag{2.41}$$

note that when deriving $\mathbf{r} \times \mathbf{v}$ from Eq. (2.39) with respect to time, the term $\dot{\mathbf{r}} \times \mathbf{v}$ vanishes since $\dot{\mathbf{r}} = \dot{\mathbf{x}} = \mathbf{v}$, so $\dot{\mathbf{r}} \times \mathbf{v} = \mathbf{v} \times \mathbf{v} = \mathbf{0}$. From this point, the expression for the balance of angular

momentum in the global form can be derived in a similar way as the global form of the balance of linear momentum. However, given that a cross product is present the divergence theorem does change somewhat and instead as in Eq. (2.34) the traction term on the boundary changes to

$$\int_{\partial\Omega} \mathbf{r} \times \mathbf{t} \mathrm{d}a = \int_{\Omega} \big(\mathbf{r} \times \mathrm{div}\boldsymbol{\sigma} + \boldsymbol{\epsilon} : \boldsymbol{\sigma}^T\big) \mathrm{d}v, \tag{2.42}$$

with $\boldsymbol{\epsilon}$ denoting the Levi-Civita symbol. Reordering the terms gives the expression

$$\int_{\Omega} \mathbf{r} \times \big(\rho\dot{\mathbf{v}} - \mathbf{b} - \mathrm{div}\boldsymbol{\sigma}\big) \mathrm{d}v = \int_{\Omega} \boldsymbol{\epsilon} : \boldsymbol{\sigma}^T \mathrm{d}v. \tag{2.43}$$

Knowing Cauchy's first equation of motion given in Eq. (2.36), it can be concluded that

$$\boldsymbol{\epsilon} : \boldsymbol{\sigma}^T = \mathbf{0}, \quad \epsilon_{abc}\sigma_{cb} = 0, \tag{2.44}$$

which gives the expressions

$$\begin{bmatrix} \sigma_{12} - \sigma_{21} \\ \sigma_{23} - \sigma_{32} \\ \sigma_{31} - \sigma_{13} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \sigma_{12} - \sigma_{21} = 0, \quad \sigma_{23} - \sigma_{32} = 0, \quad \sigma_{31} - \sigma_{13} = 0, \tag{2.45}$$

$$\sigma_{12} - \sigma_{21} = 0, \quad \sigma_{23} - \sigma_{32} = 0, \quad \sigma_{31} - \sigma_{13} = 0, \tag{2.46}$$

implying that the Cauchy stress tensor is symmetric so the relation

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^T \tag{2.47}$$

holds. This relation is also called Cauchy's second equation of motion. It also implies the symmetry of the Kirchhoff stress tensor and the 2^{nd} Piola-Kirchhoff stress tensor since they are related to Cauchy's stress tensor by Eqs. (2.25) and (2.26).

Balance of mechanical energy The balance of mechanical energy requires that the work done by external forces acting on a body $\mathcal{P}_{\text{ext}}$ is equal to the mechanical work done by the deformation of the body itself $\mathcal{P}_{\text{int}}$ and the change of kinematic energy of the body $\mathcal{K}$. This can be written as

$$\frac{\mathrm{D}}{\mathrm{D}t}\mathcal{K} + \mathcal{P}_{\text{int}} = \mathcal{P}_{\text{ext}}, \tag{2.48}$$

with the expression for the kinematic energy taken as

$$\mathcal{K} = \int_{\Omega} \frac{1}{2}\rho\mathbf{v} \cdot \mathbf{v} dv. \tag{2.49}$$

The external mechanical work is that of the external force in Eq. (2.32), $\boldsymbol{f} \cdot \mathbf{u}$, from which

the rate of exterior mechanical work follows as

$$\mathcal{P}_{\text{ext}} = \frac{\mathrm{D}}{\mathrm{D}t}(\boldsymbol{f} \cdot \mathbf{u}) = \int_{\Omega} \mathbf{b} \cdot \mathbf{v} \mathrm{d}v + \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{v} \mathrm{d}a. \tag{2.50}$$

In order to obtain the expression for $\mathcal{P}_{\text{int}}$, Eqs. (2.49) & (2.50) are put into Eq. (2.48). Before that the term for the work done on the boundary in Eq. (2.50) is expanded using the divergence theorem as

$$\int_{\partial\Omega} \mathbf{t} \cdot \mathbf{v} \mathrm{d}a = \int_{\partial\Omega} (\boldsymbol{\sigma}\mathbf{n}) \cdot \mathbf{v} \mathrm{d}a = \int_{\Omega} \mathrm{div}\left(\boldsymbol{\sigma}^T \mathbf{v}\right) \mathrm{d}v = \int_{\Omega} (\mathrm{div}\boldsymbol{\sigma} \cdot \mathbf{v} + \boldsymbol{\sigma} : \mathrm{grad}\mathbf{v}) \, \mathrm{d}v, \tag{2.51}$$

so that the expression from Eq. (2.48) can be rewritten using Eqs. (2.18), (2.21) & (2.36) as

$$\begin{aligned} \mathcal{P}_{\text{int}} &= \mathcal{P}_{\text{ext}} - \frac{\mathrm{D}}{\mathrm{D}t}\mathcal{K} = \int_{\Omega} \mathbf{b} \cdot \mathbf{v} \mathrm{d}v + \int_{\Omega} (\mathrm{div}\boldsymbol{\sigma} \cdot \mathbf{v} + \boldsymbol{\sigma} : \mathrm{grad}\mathbf{v}) \, \mathrm{d}v - \int_{\Omega} \rho\dot{\mathbf{v}} \cdot \mathbf{v} \mathrm{d}v = \\ &= \int_{\Omega} \boldsymbol{\sigma} : \underbrace{\mathrm{grad}\mathbf{v}}_{=\boldsymbol{\ell}} \mathrm{d}v + \int_{\Omega} \underbrace{(\mathrm{div}\boldsymbol{\sigma} + \mathbf{b} - \rho\dot{\mathbf{v}})}_{=0} \cdot \mathbf{v} \mathrm{d}v = \int_{\Omega} \boldsymbol{\sigma} : (\mathbf{d} + \mathbf{w}) \, \mathrm{d}v = \int_{\Omega} \boldsymbol{\sigma} : \mathbf{d} \, \mathrm{d}v, \end{aligned} \tag{2.52}$$

where the property that the inner product of a symmetric and antisymmetric tensor is equal to zero is used, i.e. $\boldsymbol{\sigma} : \mathbf{w} = \mathbf{0}$. The final expression for $\mathcal{P}_{\text{int}}$ is also called the stress power and in Eq. (2.52) is expressed in the spatial configuration. Alternative expressions in the material configuration can be obtained using Eqs. (2.8), (2.20) & (2.24) as

$$\mathcal{P}_{\text{int}} = \int_{\Omega} \boldsymbol{\sigma} : \left(\dot{\mathbf{F}}\mathbf{F}^{-1}\right) \mathrm{d}v = \int_{\Omega_0} J\boldsymbol{\sigma}\mathbf{F}^{-T} : \dot{\mathbf{F}} \mathrm{d}V = \int_{\Omega_0} \mathbf{P} : \dot{\mathbf{F}} \mathrm{d}V = \int_{\Omega_0} \mathbf{S} : \dot{\mathbf{E}} \mathrm{d}V, \tag{2.53}$$

and with $\dot{\mathbf{E}}$ given as

$$\dot{\mathbf{E}} = \frac{1}{2}\dot{\overline{\mathbf{F}^T\mathbf{F}}} = \frac{1}{2}\left(\dot{\overline{\mathbf{F}^T}}\mathbf{F} + \mathbf{F}^T\dot{\mathbf{F}}\right) = \frac{1}{2}\left(\mathbf{F}^T\boldsymbol{\ell}^T\mathbf{F} + \mathbf{F}^T\boldsymbol{\ell}\mathbf{F}\right) = \mathbf{F}^T\frac{1}{2}\left(\boldsymbol{\ell}^T + \boldsymbol{\ell}\right)\mathbf{F} = \mathbf{F}^T\mathbf{d}\mathbf{F}. \tag{2.54}$$

Balance of thermal energy (First law of thermodynamics) The first law of thermodynamics states that the rate of change of total energy (both kinetic $\mathcal{K}$ and internal $\mathcal{E}$) of a thermodynamic system equals the rate at which external mechanical work $\mathcal{P}_{\text{ext}}$ is done on that system plus the rate at which thermal work $\mathcal{Q}$ is done by heat fluxes and heat sources. The internal energy of a body $\mathcal{E}$ is defined as:

$$\mathcal{E}(t) = \int_{\Omega} e_{\text{c}}(\mathbf{x}, t) \mathrm{d}v, \tag{2.55}$$

where e_{c} is the internal energy at a current position and time. The rate of thermal work $\mathcal{Q}$ is defined as the sum of the work of heat fluxes and heat sources

$$\mathcal{Q}(t) = \int_{\partial\Omega} q_{\text{n}} \mathrm{d}a + \int_{\Omega} r \mathrm{d}v = \int_{\partial\Omega_0} Q_{\text{N}} \mathrm{d}A + \int_{\Omega_0} R \mathrm{d}V, \tag{2.56}$$

where the heat flux is denoted by q_{n} or Q_{N} and the heat source by r or R in the spatial and current

configurations, respectively. The heat fluxes can further be defined as

$$q_{\mathrm{n}}(\mathbf{x},t,\mathbf{n}) = -\mathbf{q}(\mathbf{x},t)\cdot\mathbf{n}, \quad Q_{\mathrm{N}}(\mathbf{X},t,\mathbf{N}) = -\mathbf{Q}(\mathbf{X},t)\cdot\mathbf{N}. \tag{2.57}$$

The balance of thermal energy can be expressed as

$$\frac{\mathrm{D}}{\mathrm{D}t}\mathcal{K}(t) + \frac{\mathrm{D}}{\mathrm{D}t}\mathcal{E}(t) = \mathcal{P}_{\text{ext}}(t) + \mathcal{Q}(t), \tag{2.58}$$

and substituting Eqs. (2.49), (2.50) and (2.56) into Eq. (2.58) the first law of thermodynamics in the spatial description is obtained as

$$\frac{\mathrm{D}}{\mathrm{D}t}\int_{\Omega}\left(\frac{1}{2}\rho\mathbf{v}^2 - e_{\mathrm{c}}\right)\mathrm{d}v = \int_{\partial\Omega}\left(\mathbf{t}\cdot\mathbf{v} + q_{\mathrm{n}}\right)\mathrm{d}a + \int_{\Omega}\left(\mathbf{b}\cdot\mathbf{v} + r\right)\mathrm{d}v. \tag{2.59}$$

Employing Eq. (2.57) and the divergence theorem in Eq. (2.56), and expressing $\mathcal{P}_{\text{int}} = \mathcal{P}_{\text{ext}} - \frac{\mathrm{D}}{\mathrm{D}t}\mathcal{K}$ from Eq. (2.52), a reduced global form of balance of energy in the spatial description can be obtained as

$$\frac{\mathrm{D}}{\mathrm{D}t}\int_{\Omega} e_{\mathrm{c}}\mathrm{d}v = \int_{\Omega}\left(\boldsymbol{\sigma} : \mathbf{d} - \mathrm{div}\mathbf{q} + r\right)\mathrm{d}v. \tag{2.60}$$

The reduced global form in the material description can be expressed as

$$\frac{\mathrm{D}}{\mathrm{D}t}\int_{\Omega_0} e\mathrm{d}V = \int_{\Omega_0}\left(\mathbf{P} : \dot{\mathbf{F}} - \mathrm{Div}\mathbf{Q} + R\right)\mathrm{d}V, \tag{2.61}$$

where e is the internal energy in the material description. Since the reference volume V is independent of time, the local form of the balance of energy can be expressed from Eq. (2.61) as

$$\dot{e} = \mathbf{P} : \dot{\mathbf{F}} - \mathrm{Div}\mathbf{Q} + R. \tag{2.62}$$

Entropy inequality principle (Second law of thermodynamics) When observing thermomechanical processes it can be seen that they posses a direction in which they occur. E.g., heat flows spontaneously from a warmer body to a colder body, the inverse does not happen. Also, consider an object that has to get from point A to point B and has the option of two routes, one longer and one shorter. If friction is taken into account then the work needed to cover these two routes is not the same even though the same endpoint is reached. In order to fulfil such conditions, an additional state variable called entropy is introduced. Rubber-like materials are polymer materials and are made of long intertwined molecular chains. During deformation these chains straighten which reduces the disorder in the material. Entropy is also referred to as a measure of disorder or randomness. It will be denoted with $\eta(\mathbf{x},t)$ and $N(\mathbf{X},t)$ in the spatial and material configurations, respectively. The entropy of an entire body will be denoted with $\mathcal{S}$ and

is defined with the expression

$$\mathcal{S}(t) = \int_{\Omega} \eta(\mathbf{x}, t) \mathrm{d}v = \int_{\Omega_0} N(\mathbf{X}, t) \mathrm{d}V. \tag{2.63}$$

The rate of entropy input into a body (or region) consists of the entropy transferred through the boundary of the body and the amount that it generated or destroyed within the body. It is denoted with $\tilde{\mathcal{Q}}$ and defined as

$$\tilde{\mathcal{Q}} = -\int_{\partial\Omega} \mathbf{h} \cdot \mathbf{n} \mathrm{d}a + \int_{\Omega} \tilde{r} \mathrm{d}v = -\int_{\partial\Omega_0} \mathbf{H} \cdot \mathbf{N} \mathrm{d}A + \int_{\Omega_0} \tilde{R} \mathrm{d}V, \tag{2.64}$$

with $\mathbf{h}$ and $\mathbf{H}$ denoting the entropy fluxes, and $\tilde{r}$ and $\tilde{R}$ denoting the entropy sources in the spatial and material descriptions respectively. Since entropy is a quantity introduced to observe the direction of a process it is required that its total value in an observed system can not decrease, i.e. the total entropy can only be increased (or in the case of reversible processes remain the same). This increase, also called the total production of entropy, can be expressed using the symbol $\Gamma(t)$ as

$$\Gamma(t) = \frac{\mathrm{D}}{\mathrm{D}t}\mathcal{S}(t) - \tilde{\mathcal{Q}}(t) \geq 0, \tag{2.65}$$

asserting that entropy can only increase with time. The entropy fluxes and sources can be connected to the heat fluxes and sources through the factor of $\frac{1}{\Theta}$, where Θ is the absolute temperature measured in Kelvin [K]:

$$\mathbf{h} = \frac{\mathbf{q}}{\Theta}, \quad \mathbf{H} = \frac{\mathbf{Q}}{\Theta}, \quad \tilde{r} = \frac{r}{\Theta}, \quad \tilde{R} = \frac{R}{\Theta}. \tag{2.66}$$

Thus, substituting Eqs. (2.63) and (2.64) into Eq. (2.65), and having in mind the relations in Eq. (2.66), yields the following expression

$$\Gamma(t) = \frac{\mathrm{D}}{\mathrm{D}t} \int_{\Omega} \eta \mathrm{d}v + \int_{\partial\Omega} \frac{\mathbf{q}}{\Theta} \cdot \mathbf{n} \mathrm{d}a - \int_{\Omega} \frac{r}{\Theta} \mathrm{d}v \geq 0, \tag{2.67}$$

resulting in what is called the Clausius-Duhem inequality. In the material configuration it takes the form of

$$\Gamma(t) = \frac{\mathrm{D}}{\mathrm{D}t} \int_{\Omega_0} N \mathrm{d}V + \int_{\partial\Omega_0} \frac{\mathbf{Q}}{\Theta} \cdot \mathbf{N} \mathrm{d}A - \int_{\Omega_0} \frac{R}{\Theta} \mathrm{d}V \geq 0. \tag{2.68}$$

Applying the divergence theorem on the boundary part of Eq. (2.68) and substituting it back into the equation, the localized form of the inequality can be obtained as

$$\dot{N} + \frac{1}{\Theta}\mathrm{Div}\mathbf{Q} - \frac{1}{\Theta^2}\mathbf{Q} \cdot \mathrm{Grad}\Theta - \frac{R}{\Theta} \geq 0, \tag{2.69}$$

and by multiplying the equation with the temperature and using Eq. (2.62) to replace heat source,

the alternative local form of the inequality is obtained as

$$\mathbf{P} : \dot{\mathbf{F}} - \dot{e} + \Theta\dot{N} - \frac{1}{\Theta}\mathbf{Q} \cdot \text{Grad}\Theta \geq 0. \tag{2.70}$$

The Helmholtz free energy per unit reference volume is introduced as $\psi = e - \Theta N$. Note that later when considering an isothermal setting (that is when neglecting temperature and entropy) the function ψ will be referred to as the strain-energy function. With the expression for ψ in mind and assuming a homogenous temperature distribution where $\text{Grad}\Theta = 0$, the Clausius-Duhem inequality reduces to the form

$$\mathbf{P} : \dot{\mathbf{F}} - \dot{\psi} - N\dot{\Theta} \geq 0. \tag{2.71}$$

The Helmholtz free energy depends on the deformation gradient $\mathbf{F}$ and the temperature Θ (it is possible that it depends on other quantities but they are not considered in this work) so that the time derivative $\dot{\psi}(\mathbf{F}, \Theta)$ can be expressed as

$$\dot{\psi}(\mathbf{F}, \Theta) = \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \mathbf{F}} : \dot{\mathbf{F}} + \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \Theta}\dot{\Theta}. \tag{2.72}$$

By replacing $\dot{\psi}$ from Eq. (2.71) with Eq. (2.72) the following expression is obtained

$$\left(\mathbf{P} - \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \mathbf{F}}\right) : \dot{\mathbf{F}} - \left(N + \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \Theta}\right)\dot{\Theta} \geq 0. \tag{2.73}$$

If the temperature is assumed to be constant, i.e. $\dot{\Theta} = 0$, then the expression leads to

$$\left(\mathbf{P} - \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \mathbf{F}}\right) : \dot{\mathbf{F}} \geq 0, \tag{2.74}$$

and since $\dot{\mathbf{F}}$ can take any arbitrary values (positive or negative) then the definition of the 1^{st} Piola-Kirchhoff stress follows as

$$\mathbf{P} - \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \mathbf{F}} = 0 \rightarrow \mathbf{P} = \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \mathbf{F}}. \tag{2.75}$$

At this point it should be reiterated that a hyperelastic material is described by a strain energy function $\psi(\mathbf{F})$ and through the constitutive relation that has been given in Eq. (2.75).

Similarly to inequality in Eq. (2.74), assuming that there is no change in deformation, i.e. $\dot{\mathbf{F}} = 0$, but that the temperature may vary so that $\dot{\Theta}$ can take any arbitrary value (positive or negative), then Eq. (2.73) leads to the definition of entropy as

$$N + \frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \Theta} = 0 \rightarrow N = -\frac{\partial \psi(\mathbf{F}, \Theta)}{\partial \Theta}. \tag{2.76}$$

In the following discussion isothermal behaviour is assumed ($\dot{\Theta} = 0$) and the effect of the temperature is neglected for simplicity. Other stress tensors can be expressed, e.g. the 2^{nd}

Piola-Kirchhoff stress tensor **S**. First, the strain energy ψ is expressed in terms of the right Cauchy-Green deformation tensor **C** through the time derivative $\dot{\psi}$ as

$$\dot{\psi} = \frac{\partial \psi(\mathbf{F})}{\partial \mathbf{F}} : \dot{\mathbf{F}} = \text{tr}\left[\left(\frac{\partial \psi(\mathbf{F})}{\partial \mathbf{F}}\right)^{\text{T}} \dot{\mathbf{F}}\right], \tag{2.77}$$

$$\begin{aligned} \dot{\psi} &= \frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}} : \dot{\mathbf{C}} = \text{tr}\left[\frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}} \dot{\mathbf{C}}\right] = \text{tr}\left[\frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}} \left(\dot{\overline{\mathbf{F}^T}}\mathbf{F} + \mathbf{F}^T\dot{\mathbf{F}}\right)\right] = \\ &= 2 \text{ tr}\left[\frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}} \mathbf{F}^{\text{T}} \dot{\mathbf{F}}\right], \end{aligned} \tag{2.78}$$

where the useful property of the trace $\text{tr}\,(\mathbf{A}^{\text{T}}\mathbf{B}) = \text{tr}\,(\mathbf{B}^{\text{T}}\mathbf{A})$ (i.e. the diagonal is always the same in the trace regardless of the transpose) is used. Also, note that **C** is symmetric, so the derivative of ψ with respect to **C** is also symmetric. From Eqs. (2.77) and (2.78) the relation between the strain energy defined in terms of **F** and **C** can be obtained as

$$\left(\frac{\partial \psi(\mathbf{F})}{\partial \mathbf{F}}\right)^{\text{T}} = 2\frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}} \mathbf{F}^{\text{T}}. \tag{2.79}$$

Substituting Eqs. (2.79) and (2.75) into Eq. (2.26) the expression for the 2^{nd} Piola-Kirchhoff stress tensor is defined using the strain energy as

$$\mathbf{S} = 2\, \frac{\partial \psi(\mathbf{C})}{\partial \mathbf{C}}. \tag{2.80}$$

This can be further expressed in terms of the principal invariants of **C** as

$$\begin{aligned} \mathbf{S} &= 2\, \frac{\partial \psi(I_1, I_2, I_3)}{\partial \mathbf{C}} = 2\left[\frac{\partial \psi}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{C}} + \frac{\partial \psi}{\partial I_2}\frac{\partial I_2}{\partial \mathbf{C}} + \frac{\partial \psi}{\partial I_3}\frac{\partial I_3}{\partial \mathbf{C}}\right] = \\ &= 2\left[\left(\frac{\partial \psi}{\partial I_1} + I_1\frac{\partial \psi}{\partial I_2}\right)\mathbf{I} - \frac{\partial \psi}{\partial I_2}\mathbf{C} + I_3\frac{\partial \psi}{\partial I_3}\mathbf{C}^{-1}\right], \end{aligned} \tag{2.81}$$

which will later be used. Furthermore, the material tangent $\mathbb{c}$ or $\mathbb{C}$ (spatial or material configuration) is necessary for calculations involving the 1^{st} derivatives of the stresses and can be computed by further deriving the stresses with respect to the appropriate deformation measure. The material tangent in the material configuration is given as

$$\mathbb{C} = 2\, \frac{\partial \mathbf{S}}{\partial \mathbf{C}} = 4\, \frac{\partial^2 \psi(\mathbf{C})}{\partial \mathbf{C} \partial \mathbf{C}}, \tag{2.82}$$

and in the spatial configuration it can be obtained through a push-forward transformation as

$$\mathbb{c} = J^{-1}\boldsymbol{\chi}_*(\mathbb{C}) \rightarrow \mathbb{c}_{\text{abcd}} = J^{-1} F_{aA} F_{bB} F_{cC} F_{dD} \mathbb{C}_{ABCD}. \tag{2.83}$$

A note on incompressibility In the case of incompressibility the stress can be determined up to

an extra component (cf. [66], Sections 30. and 80.) as already presented in Eq. (2.27). In terms of the strain energy this can be expressed with Eqs. (2.24) and (2.75) as

$$\mathbf{P} = \frac{\partial \psi\left(\mathbf{F}\right)}{\partial \mathbf{F}} - Jp\mathbf{F}^{-\mathrm{T}}. \tag{2.84}$$

The value of the extra component p is determined either through equilibrium conditions (e.g. in a plane stress setting) or with the use of specialised numerical techniques. The extra component p can also be viewed as a Lagrange multiplier that enforces the incompressibility constraint that no volume change should occur, i.e. $\det(\mathbf{C}) = 1$.

A note on the behaviour of rubber-like materials Rubber-like materials are polymers and consist of intertwined molecular chains. Their behaviour is often characterised as hyperelastic since they are often described through a strain energy function $\psi(\mathbf{F})$ and the relation Eq. (2.75). Some characteristic properties which are common to all rubber-like materials include:

- Near-incompressibility - although some volume change may occur during deformation [7,52] these materials are often times regarded as incompressible since their bulk modulus is several orders of magnitude greater than their shear modulus [50]. Due to this large difference incompressibility is considered as a valid assumption when modelling such materials.

- Stress in these materials is caused by a change in entropy with deformation, due to the straightening of the aforementioned intertwined molecular chains, so these materials are referred to as *entropic*, whereas in metals the entropy does not change with deformation.

- A piece of rubber first cools and then warms upon stretching, this is shown later in Sec. 3.3 Fig. 3.3. Also, when a piece of stretched rubber is warmed it contracts and if it is cooled it extends, contrary to the usual behaviour of materials.

These are some of the characteristics which must be considered when modelling the behaviour of rubber materials. In the majority of this thesis the hyperelastic behaviour is assumed to be incompressible. Appropriate measures of deformation must be considered when modelling hyperelasticity and an investigation is made between different admissible measures. Additionally, the influence of self-heating during deformation is investigated.

3 Implementation of Conventional Neural Networks

Neural networks are a potent mathematical tool inspired by the way the brain functions. They were first proposed as a model for the brain by neurophysiologist Warren McCulloch and mathematician Walter Pitts in 1943 [47]. Since then they slowly found their way into the field of solid mechanics with papers on various applications in computational mechanics in the late 1970's and in the 1980's [71], and in 1989 they were proposed as universal approximators in the work of Hornik [28]. With the advent of larger computational capabilities and the recent advancements in other sciences, machine learning applications gathered a strong interest in the computational mechanics community. For a thorough explanation of neural networks the reader is referred to [22]. In this work the machine learning library TensorFlow [2] is used to develop and train neural networks.

3.1 Feedforward Neural Networks

In this chapter the *Feedforward Neural Network* (FNN) (also called a *MultiLayer Perceptron*, MLP) is explained and the way in which it will be used to model hyperelasticity is shown. In Fig. 3.1 an illustration of a FNN is given. It is comprised of multiple sequentially ordered layers that are filled with neurons. The depth of a neural network is the number of hidden layers it has and the width of a layer is the number of neurons inside it. The neurons are depicted as circles, while the layers are enclosed with a dashed line. Values are passed from the input layer to the hidden layer while at the same time multiplying the weights between the layers. An activation function can also be applied to each layer, passing the neurons through the activation function before forwarding their values to the next layer (in Fig. 3.1 the activation function is not depicted). This is repeated between all layers in a forward manner (depicted by the arrow in Fig. 3.1) until the output layer is reached. These chained operations can be described with the expression on how to calculate the values of an arbitrary hidden layer:

$$\boldsymbol{h}^{(l)} = g^{(l)}\left(\boldsymbol{W}^{(l)\mathrm{T}}\boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \quad l = 1, ..., n. \tag{3.1}$$

In the equation given above $\boldsymbol{h}^{(l)}$ is the l-th layer of a FNN, $g^{(l)}$ is the activation function corresponding to that layer, $\boldsymbol{W}^{(l)}$ are the weights between the current $\boldsymbol{h}^{(l)}$ and previous layer $\boldsymbol{h}^{(l-1)}$, $\boldsymbol{b}^{(l)}$ are the biases added to the neurons of the current layer, l is the number of the layer with $\boldsymbol{h}^{(0)}$ being the input layer, and $\boldsymbol{h}^{(n)}$ the output layer. The total number of layers after the input layer is denoted with n.

A neural network is used in this work as a regression model with many parameters (e.g. the weights in $\boldsymbol{W}^{(l)}$) which can be optimised to better describe certain behaviour. This is done through minimising an objective function, also called a loss function. For example, let $L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

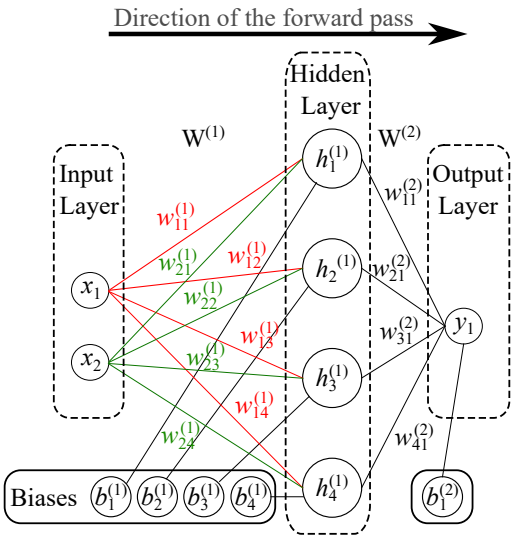


Figure 3.1: Illustration of a *Feedforward Neural network* (FNN).

be such an objective function in the form of

$$L\left(\hat{\boldsymbol{y}}, \boldsymbol{y}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\left(\boldsymbol{x}; \boldsymbol{w}\right)\right)^2, \tag{3.2}$$

where $\hat{\boldsymbol{y}}$ are the "true" (or target) values from the data that represent the function to be approximated, $\boldsymbol{y}\left(\boldsymbol{x}; \boldsymbol{w}\right)$ are the predicted values of the neural network with $\boldsymbol{x}$ being the inputs to the network and $\boldsymbol{w}$ all the trainable parameters of the network, and N is the number of training samples. This presented function is called the *mean squared error* (MSE). Starting from some initial values of the trainable parameters of the neural network, the objective is to find such parameters $\boldsymbol{w}$ that minimise the value of a loss function $L(\boldsymbol{w})$, which is the same loss function given before just rewritten to state the dependency of the loss to the tranable parameters. In order to find the direction in which the loss function decreases the fastest the problem changes to finding the minimum of the directional derivative

$$\min_{\boldsymbol{u}, \boldsymbol{u}^T\boldsymbol{u}=1} \frac{\partial}{\partial \alpha}\left[L(\boldsymbol{w} + \alpha\boldsymbol{u})\right]|_{\alpha=0}, \tag{3.3}$$

where $\boldsymbol{u}$ is the unit vector defining the direction with α being equal to 0. The directional derivative leads to

$$\frac{\partial}{\partial \alpha}\left[L(\boldsymbol{w} + \alpha\boldsymbol{u})\right]|_{\alpha=0} = \left[\frac{\partial L(\boldsymbol{w} + \alpha\boldsymbol{u})}{\partial(\boldsymbol{w} + \alpha\boldsymbol{u})} \cdot \frac{\partial(\boldsymbol{w} + \alpha\boldsymbol{u})}{\partial \alpha}\right]_{\alpha=0} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} \cdot \boldsymbol{u} = \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) \cdot \boldsymbol{u}, \tag{3.4}$$

which is a scalar product. Expanding the last part of the previous equation to the definition of a scalar product and substituting it into Eq. (3.3) leads to

$$\min_{\boldsymbol{u}, \boldsymbol{u}^T\boldsymbol{u}=1} |\nabla_{\boldsymbol{w}} L(\boldsymbol{w})|\ |\boldsymbol{u}| \cos\theta \tag{3.5}$$

and since $|\boldsymbol{u}| = 1$ and $|\nabla_{\boldsymbol{w}} L(\boldsymbol{w})|$ does not depend on $\boldsymbol{u}$, the expression further simplifies to $\min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1} \cos\theta$, for which the minimum is achieved when the unit vector $\boldsymbol{u}$ is pointing in the opposite direction of the gradient $\nabla_{\boldsymbol{w}} L(\boldsymbol{w})$, i.e. when $\cos\theta = -1$. This leads to the conclusion that the loss function decreases most when moving in the negative direction of its gradients leading to the relation

$$\boldsymbol{w} := \boldsymbol{w} - \epsilon \nabla_{\boldsymbol{w}} L(\hat{\boldsymbol{y}}, \boldsymbol{y}), \tag{3.6}$$

where ϵ is a small number called the learning parameter (or step size), and the operator ":=" denotes the variable update in an algorithm. The gradients of the loss function with respect to the trainable parameters are calculated using automatic differentiation built into TensorFlow. Apart from the basic version of the stochastic gradient descent algorithm shown in Eq. (3.6), there are multiple versions which incorporate some improvements. The variant that proved to be most effective in this work for training the presented neural networks is the *Adam* optimizer [35] which is described in Algorithm 1.

Algorithm 1 Adam optimizer.

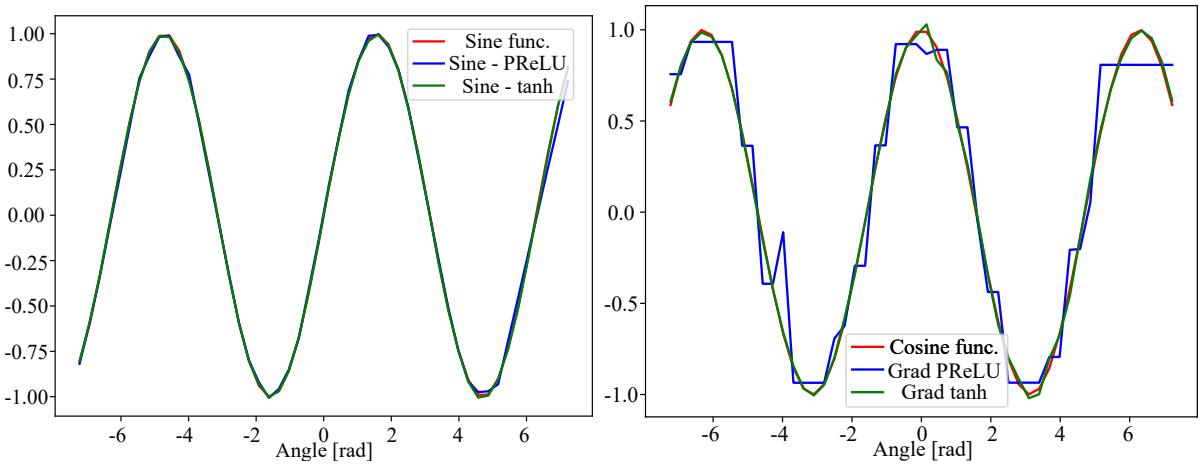
Adam introduces 2 moments during training, the first moment estimate $\boldsymbol{s}$ and the second moment estimate $\boldsymbol{r}$. The required variables are β_1 and β_2 (decay rates), and the parameter η used in this algorithm to prevent division by zero. The learning rate ϵ is borrowed from Eq. (3.6). This algorithm is performed during each training step. Variables are initialised at the start of training, common values are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\eta = 10^{-7}$, $\epsilon = 0.001$.

1: $t := t + 1$ ▷ Update the step during training.
2: $\boldsymbol{g} = \nabla_{\boldsymbol{w}} L(\boldsymbol{w})$ ▷ Get gradients.
3: $\boldsymbol{s} := \beta_1 \boldsymbol{s} + (1 - \beta_1)\boldsymbol{g}$ ▷ Update biased first moment estimate.
4: $\boldsymbol{r} := \beta_2 \boldsymbol{r} + (1 - \beta_2)\boldsymbol{g}^2$ ▷ Update biased second moment estimate.
5: $\hat{\boldsymbol{s}} := \boldsymbol{s}/(1 - \beta_1^t)$ ▷ Correct bias in first moment.
6: $\hat{\boldsymbol{r}} = \boldsymbol{r}/(1 - \beta_2^t)$ ▷ Correct bias in second moment.
7: $\boldsymbol{w} := \boldsymbol{w} - \epsilon \cdot \hat{\boldsymbol{s}}/(\sqrt{\hat{\boldsymbol{r}}} + \eta)$

With this the basic function of the neural network is explained as well as the training algorithm.

3.2 Choice of the Activation Function

Neural networks can be looked at as simple regression models that can be relatively easily trained on a large number of samples while having a large number of trainable parameters. They should be able to capture a wide range of functions simply by expanding the width (number of neurons per layer) [28]. Also, using the automatic differentiation tool built into modern machine learning libraries such as TensorFlow, the neural network can easily be incorporated into finite element calculations. For example, learning the stress-strain relationship via a neural network enables the calculation of the material tangent. Having a neural network that calculates the stresses and material tangent makes it an ideal drop-in replacement for conventional models in finite element software such as Abaqus.



(a) (b)

Figure 3.2: Prediction of the sine function using an FNN with PReLU and tanh activation functions, and the gradients of the respective FNNs.

In Section 3.1 the activation function is mentioned but not studied in detail. The importance of the activation function is considerable when attempting to model any type of behaviour. In this work the underlying hyperelastic behaviour is always nonlinear so it is important to choose an activation function that can correctly capture nonlinearities. The sine is used as a toy example to highlight the importance of choosing an adequate activation function. An overview of the possible candidate functions that are widely used is given in [42]. To illustrate the importance of the choice 2 functions are studied, the Parametric Rectified Linear Unit (PReLU) and the hyperbolic tangent (tanh). The PReLU function is defined as

$$f(x) = \begin{cases} \alpha x & \text{if} \quad x < 0 \\ x & \text{if} \quad x \geq 0, \end{cases}, \qquad \frac{\partial f(x)}{\partial x} = \begin{cases} \alpha & \text{if} \quad x < 0 \\ 1 & \text{if} \quad x \geq 0, \end{cases} \tag{3.7}$$

whereas the tanh is a well known function defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \frac{\partial \tanh(x)}{\partial x} = 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right)^2. \tag{3.8}$$

Comparing the functions it can be seen that the main difference is that the PReLU is linear and discontinuous at 0 with a sharp jump in its derivative, while the tanh is nonlinear and continuous at 0. Based solely on this the tanh seems to be the proper candidate to describe the sine function. Plotting the predictions of the sine function in Fig. 3.2a both FNNs correctly capture the sine and the choice of function seems arbitrary. However, looking at the gradients of the FNNs (i.e. the cosine function) in Fig. 3.2b it can be seen that the gradients of the FNN using PReLU are constants that somewhat approximate the cosine curve, while the FNN using the tanh correctly captures the gradients rendering it the preferred activation function for such a task.

When training an NN the loss function is a key element. Although it is intuitive that the loss

function contains the quantity it predicts, it can be formulated that it contains the derivatives of the NN. In the case of solid mechanics this would mean that if the NN predicts the strain energy the loss function would contain the stresses. This will be explored in the later sections of this work.

3.3 Adiabatic Thermoelasticity

In order to properly asses the abilities of NNs they are firstly applied to capture the difference between isothermal rubber behaviour and adiabatic thermoelasticity, incompressibility is assumed. This behaviour is chosen because the differences are small and occur during various loading conditions [23]. When considering thermoelasticity the deformation gradient can be decomposed into a mechanical and thermal part, i.e. $\mathbf{F} = \mathbf{F}_{\mathrm{M}}\mathbf{F}_{\theta}$. In the incompressible case $\det(\mathbf{F}_{\mathrm{M}}) = J_{\mathrm{M}} = 1$. The thermal volume change can be calculated as $\det(\mathbf{F}_{\theta}) = J_{\theta} = \exp[3\alpha_0\,(\Theta - \Theta_0)]$, where α_0 is the coefficient of thermal expansion, Θ is the current temperature and Θ_0 is the reference temperature. The volume change from Eq. (2.7) can be defined as $J = J_{\mathrm{M}}J_{\Theta}$.

In the non-isothermal case the strain energy must be augmented by a thermal contribution here denoted as $T(\Theta)$ with the expression

$$T(\Theta) = c_0 \left[(\Theta - \Theta_0) - \Theta \ln\left(\frac{\Theta}{\Theta_0}\right) \right], \tag{3.9}$$

where c_0 is the specific heat capacity. The expanded expression is referred to as the free energy function $\psi(\mathbf{F}, \Theta)$. In order to somewhat simplify the behaviour to only include changes in temperature due to deformation an adiabatic process is considered, i.e. the entropy N is kept constant during the process.

In this work the principal stretches $(\lambda_a,\ a = 1, 2, 3)$ based Ogden hyperelastic model [51] is taken as the base hyperelastic behaviour and then expanded to account for the thermal contribution [26]. The specific free energy function now takes the form

$$\psi\left(\lambda_1, \lambda_2, \lambda_3, \Theta\right) = \underbrace{\sum_{p=1}^{3} \frac{\mu_p(\Theta)}{\alpha_p}(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_3^{\alpha_p} - 3)}_{\psi_{\text{Ogden}}} + \underbrace{c_0 \left[(\Theta - \Theta_0) - \Theta \ln\left(\frac{\Theta}{\Theta_0}\right) \right]}_{T(\Theta)}. \tag{3.10}$$

The Ogden model is defined by the shear moduli μ_p and dimensionless constants α_p. All the values for the material parameters are given in Table 1. The reference temperature Θ_0 is taken as 293.15 K.

Additionally, it is assumed that the shear moduli takes linear variation with temperature [12, 26, 53]:

$$\mu_p(\Theta) = \mu_p(\Theta_0)\frac{\Theta}{\Theta_0}. \tag{3.11}$$

It should be noted that the linearity of shear moduli as a function of temperature is valid

Table 1: Material parameters.

μ	μ_1	μ_2	μ_3
[MPa]	0.63	0.0012	-0.01
α	α_1	α_2	α_3
$[-]$	1.3	5	-2
α_0	22.3333e-5 K^{-1}	c_0	1.83 $\frac{\mathrm{Nmm}}{\mathrm{kgK}}$

exclusively for small and medium strains. This results from the application of Gaussian statistical theory to long-chain molecules, which provides a considerable simplification. Behaviour of rubber at extreme strains requires the application of non-Gaussian theory. Nevertheless, in order to provide comparison to results of other authors, the linearity assumption is also applied to the large strain regime. The reader's attention is drawn to the fact that this may lead to discrepancies with the real behaviour of rubber.

As mentioned earlier the entropy is kept constant and is defined in Eq. (2.76). The thermoelastic behaviour is demonstrated on uniaxial tension where $\lambda_2 = \lambda_3 = \left(\frac{J_\Theta}{\lambda_1}\right)^{0.5}$, this comes from Eq. (2.7). Also, in the uniaxial case λ_1 is referred to simply as λ. For the uniaxial case the free energy function now takes the form

$$\psi(\lambda, \Theta) = \sum_{p=1}^{3} \frac{\mu_p(\Theta)}{\alpha_p} \left[\lambda^{\alpha_p} + 2 \left(\frac{J_\Theta}{\lambda} \right)^{\frac{\alpha_p}{2}} - 3 \right] + c_0 \left[(\Theta - \Theta_0) - \Theta \ln \left(\frac{\Theta}{\Theta_0} \right) \right], \quad (3.12)$$

and the expression for the entropy is obtained as

$$N(\lambda, \Theta) = -\frac{1}{\Theta_0} \sum_{p=1}^{3} \frac{\mu_p(\Theta_0)}{\alpha_p} \left[\lambda^{\alpha_p} + (2 + 3\alpha_0 \Theta \alpha_p) \left(\frac{J_\Theta}{\lambda} \right)^{\frac{\alpha_p}{2}} - 3 \right] + c_0 \ln \left(\frac{\Theta}{\Theta_0} \right). \quad (3.13)$$

Given the isentropic condition $N = \text{const.}$ the expression for the temperature Θ is obtained as:

$$\Theta(\lambda) = \Theta_0 \exp \left\{ c_0^{-1} \left[N(1, \Theta_0) + \Theta_0^{-1} \sum_{p=1}^{3} \left[\lambda^{\alpha_p} + (2 + 3\alpha_0 \Theta \alpha_p) \left(\frac{J_\Theta}{\lambda} \right)^{\frac{\alpha_p}{2}} - 3 \right] \right] \right\}, \quad (3.14)$$

with $N(1, \Theta_0)$ being the entropy at the beginning of the deformation process. The evolution of temperature during uniaxial loading is shown in Fig. 3.3 for a stretch of $\lambda = 1.6$ in order to show the temperature inversion effect where the during the initial loading the temperature begins to fall and afterwards increases.

The expression for the 1st Piola-Kirchhoff stress is obtained by differentiating the free energy function w.r.t. the stretch λ. The expression is obtained as

$$P_1 = \sum_{p=1}^{3} \frac{\mu_p(\Theta)}{\lambda} \left[\lambda^{\alpha_p} - \left(\frac{J_\Theta}{\lambda} \right)^{\frac{\alpha_p}{2}} \right]. \quad (3.15)$$

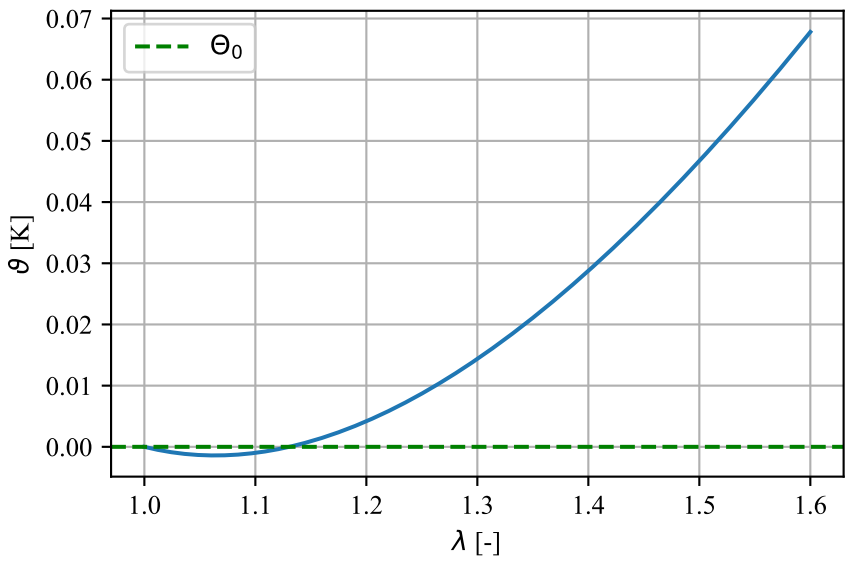


Figure 3.3: Temperature change for a uniaxially loaded specimen.

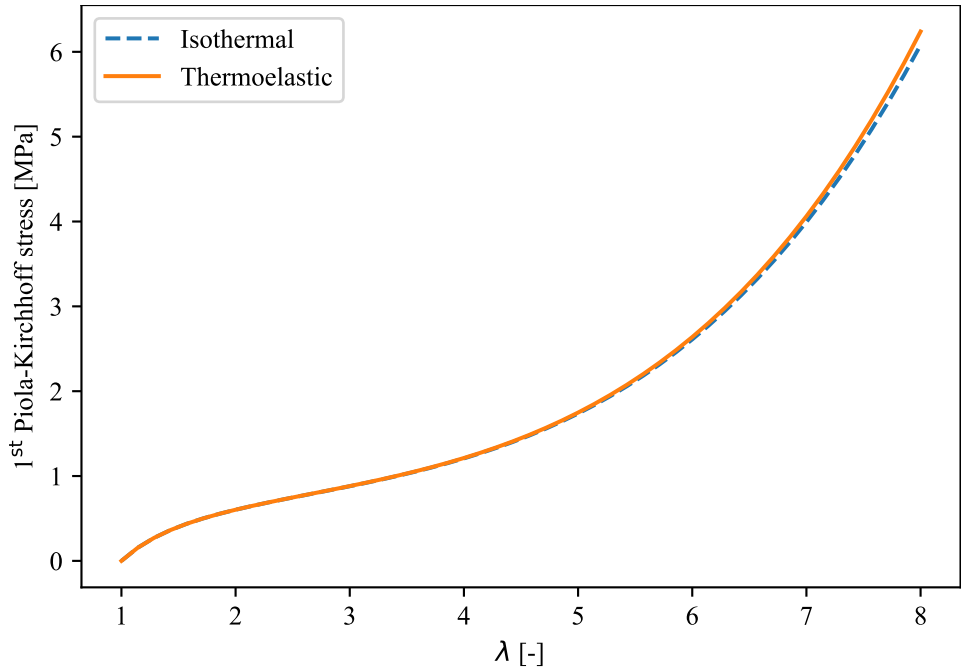
The isothermal and thermoelastic stress responses are shown in Fig. 3.4a, the difference in stresses is quite small as seen in the figure, only manifesting near the very end of loading. For $\lambda = 8$ the stress difference is 2.61%. In Fig. 3.4b the temperature changes are shown, and the highest is just below 8 K for a stretch of 8, i.e. 700% deformation. Note that since entropic materials shrink during heating the stress values for the thermoelastic case are slightly higher. Given that the differences are small and produced by introducing a real-world behaviour rather than just slightly adjusting the material parameters, these two behaviours serve as a baseline to test how well can neural networks capture small differences in data. That is, if neural networks can properly capture two different but very similar behaviours.

In order to test the approximation properties of NNs, plane stress conditions are assumed which allow to easily calculate the Lagrange multiplier p from the condition $\sigma_3 = 0$. An NN is constructed that predicts components of the Cauchy stress tensor from logarithmic strain. These two were chosen because because they are used by the FE software Abaqus thus implementing the NNs within the UMAT (User MATerial) subroutine of Abaqus is easier. In order to generate a large and diverse amount of data a number of deformation gradients of the following form were generated:

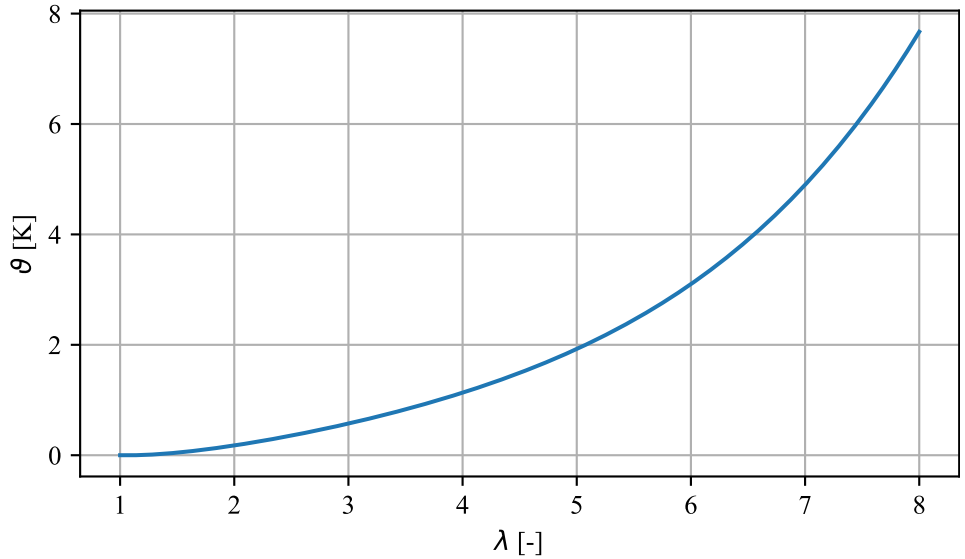
1. Uniaxial deformation

$$\mathbf{F} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & (\frac{J_\Theta}{\lambda})^{0.5} & 0 \\ 0 & 0 & (\frac{J_\Theta}{\lambda})^{0.5} \end{bmatrix},$$

2. Biaxial deformation



(a)



(b)

Figure 3.4: Isothermal and thermoelastic response during uniaxial loading are shown in the top figure, the temperature change during the deformation is shown in the bottom figure.

$$\mathbf{F} = \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & J_\Theta/\lambda_x\lambda_y \end{bmatrix},$$

3. Simple shear - 2 variations

$$\mathbf{F} = \begin{bmatrix} \lambda_x & \gamma & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & J_\Theta/\lambda_x\lambda_y \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} \lambda_x & 0 & 0 \\ \gamma & \lambda_y & 0 \\ 0 & 0 & J_\Theta/\lambda_x\lambda_y \end{bmatrix},$$

4. General shear

$$\mathbf{F} = \begin{bmatrix} \lambda_x & \gamma/2 & 0 \\ \gamma/2 & \lambda_y & 0 \\ 0 & 0 & \lambda_z \end{bmatrix},$$

5. No deformation

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

all of these five types of deformation gradients were generated in equal parts. The values for generating the training dataset were randomly chosen from $\lambda \in [0.5, 8]$, which would correspond to the maximum values in Treloars experiment [65], and the shear deformation γ was chosen from $\gamma \in [-0.5, 0.5]$. The logarithmic strain tensor is calculated from the left stretch tensor $\mathbf{V}$ as

$$\boldsymbol{\varepsilon}^L = \ln(\mathbf{V}) = \sum_{i=1}^{3} \ln(\lambda_i)\mathbf{n}_i\mathbf{n}_i^T. \tag{3.16}$$

An FNN is used to predict the output consisting of three stress components $\boldsymbol{\sigma}_{\text{NN}} = [\sigma_x, \sigma_y, \tau_{xy}]^T$ from the input consisting of three logarithmic strain components $\boldsymbol{\varepsilon}^L = [\varepsilon_x, \varepsilon_y, 2\varepsilon_{xy}]$. The NN architecture consists of the following:

- 3 hidden layers, 60 neurons per layer, referred to as (3-60-60-60-1) - for thermoelastic behaviour
- 2 hidden layers, 60 neurons per layer, referred to as (3-60-60-1) - for isothermal hyperelastic behaviour
- 3 separate neural networks, 1 for calculating each stress component
- no biases are included
- tanh used as the activation function
- Adam optimizer used during training
- Glorot initialisation used for the weights [21]

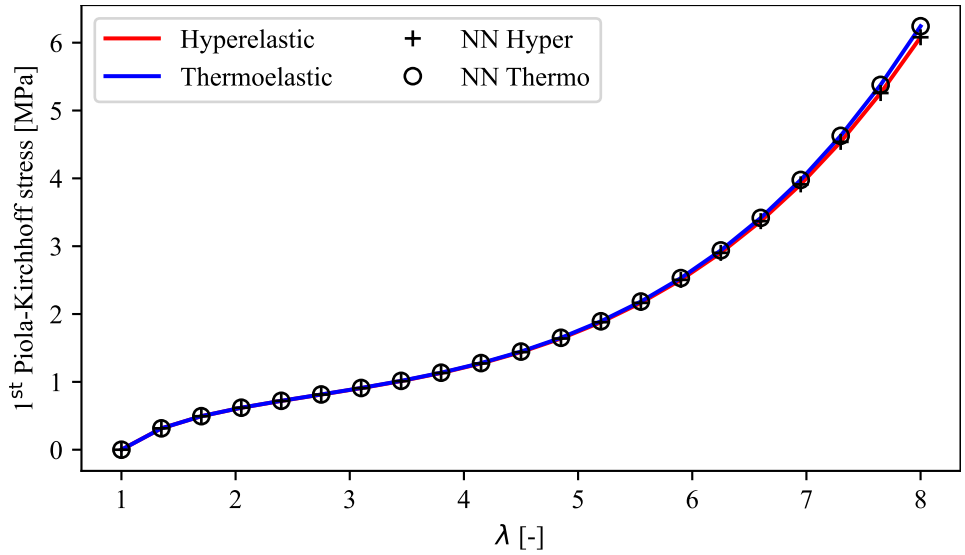


Figure 3.5: Thermoelastic and isothermal uniaxial responses, comparison of reference solutions to the NN solutions.

By omitting the biases, and using the tanh with the property $\tanh(0) = 0$, for no deformation ($\boldsymbol{\varepsilon}^L = \mathbf{0}$) the stresses will be 0, therefore the normalisation condition for the stresses is satisfied *a priori* with the NN construction. The loss function takes the form of:

$$L(\boldsymbol{\varepsilon}^L, \boldsymbol{\sigma}) = \frac{1}{N} \sum_{i=1}^{N} \left(\sigma_{\mathrm{NN},k} \left(\boldsymbol{\varepsilon}_i^L \right) - \sigma_{i,k} \right)^2 , \tag{3.17}$$

where the loss function is reused 3 times for each neural network and the index k stands for the possible predicted stresses, i.e. $k = x, y, xy$.

Results for the uniaxial case are shown in Fig. 3.5 and they show good agreement with an error for the isothermal case of 0.008% and for the thermoelastic case of 0.08%. The results are obtained from finite element analysis using Abaqus with both the NN and analytical solution implemented via UMAT. The corresponding temperature change during uniaxial deformation is shown in Fig. 3.4b.

Given that the NN was trained on uniaxial data the good results are expected. A simple test to see if the NN captured the material behaviour would be to test it on a deformation mode it did not see during training. One such mode of deformation is planar tension defined by the following deformation gradient:

$$\mathbf{F} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{J_\Theta}{\lambda} \end{bmatrix} . \tag{3.18}$$

The results are shown in Fig. 3.6 and the relative error for the isothermal case is 0.005%, and for the thermoelastic case the error is 0.015%. Although a simple test, this shows that the NN captured the underlying behaviour.

It must be noted that 1 000 000 examples were generated and split into a 75/25 train/test split.

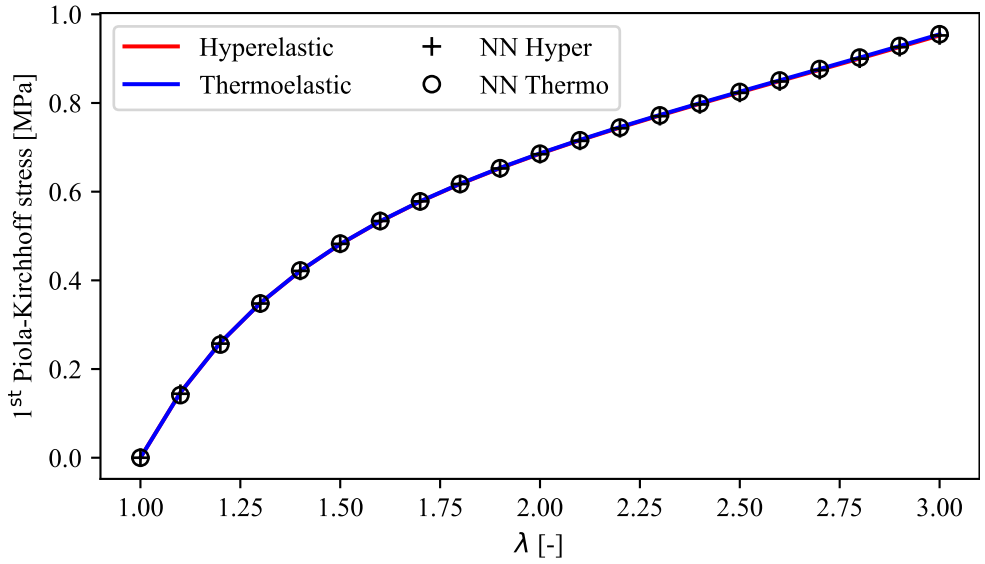


Figure 3.6: Thermoelastic and isothermal planar tension responses, comparison of reference solutions to the NN solutions.

Such a high number of examples was necessary not to capture simple modes of deformation such as uniaxial or biaxial loading, but rather in order to capture more complex modes of deformation. The presented NN is not objective by design, it does not distinguish between rotated states. By enriching the database with many samples this problem is alleviated.

3.4 Cook Membrane Example

In order to further verify the NN model it was tested on more complex problems such as the standard Cook's membrane, geometry shown in Fig. 3.7. The dimensions are given in millimetres, the small letter t notes the traction on the right boundary, the red dot marks the corner where stress results were taken and the blue dot where the displacement results were taken.

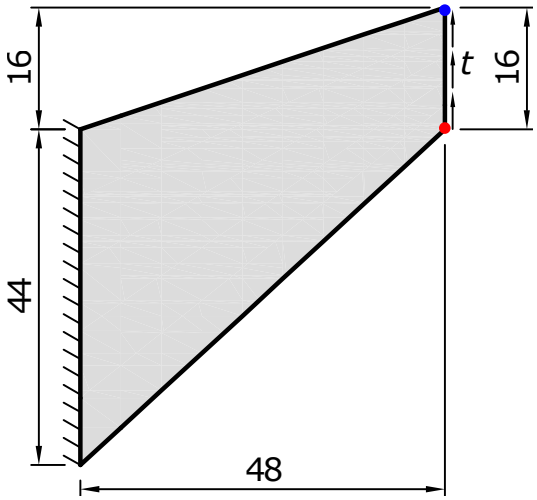


Figure 3.7: Geometry of Cook's membrane problem.

The neural network does not take the temperature as a separate input since in the adia-

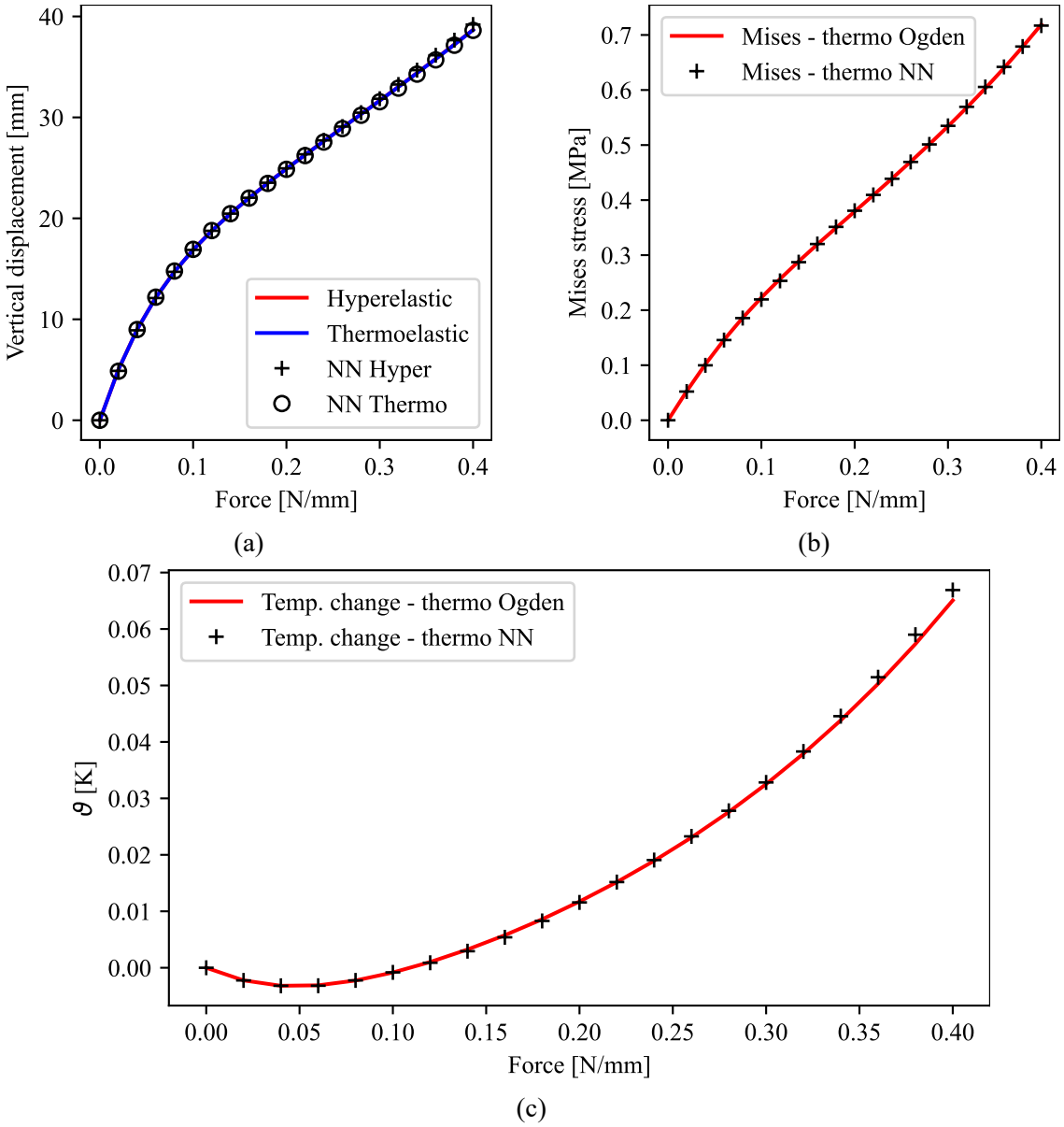


Figure 3.8: Diagrams showing the displacements at the upper right corner (blue dot), and the von Mises stress and temperature evolutions at the lower right corner (red dot).

batic case the temperature itself is a function of the stretch λ, i.e. the deformation, as shown in Eq. (3.14). Inversely, the temperature can be obtained by post-processing the results to see what temperature would correspond to a certain state of deformation. In this manner it can further be checked if the NN has captured the desired behaviour.

The results for the displacements, von Mises stress and temperature are shown in Fig. 3.8. The results show good agreement for all the quantities.

3.5 Rubber Seal Numerical Example

Cook's membrane is a standard benchmark test, but the validity of the approach can be tested on other numerical examples. One such example is taken from real-world applications and is

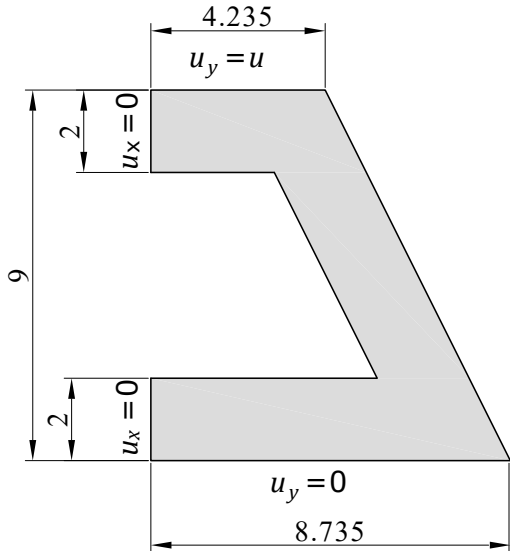


Figure 3.9: Geometry of the rubber seal example.

the rubber seal example used in [56]. It consists of one half of a rubber seal with a symmetry boundary condition on its left edges and fixed in the vertical direction at the bottom edge. The seal is then loaded on the top edge with a prescribed displacement mimicking conditions in operation. The geometry is shown in Fig. 3.9 and is given in millimetres with the prescribed displacement u shown at the top. The downward displacement u is equal to 2.2 mm. Plane stress conditions are assumed and the FE mesh size is 0.5 mm.

In Fig. 3.10, a plot of the reaction force at the top edge versus the displacement is shown. It can be seen that both the hyperelastic and thermoelastic behaviors are well described. The final relative error for the hyperelastic model is 0.6% and 1.4% for the thermoelastic model. The stretch is relatively small, so the effect of adiabatic heating on the total reaction force is not noticeable. The accuracy of the model may be better evaluated in the region with the highest stress at the top edge.

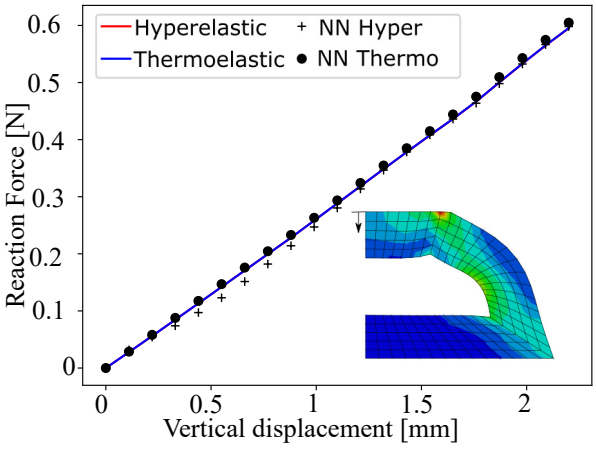
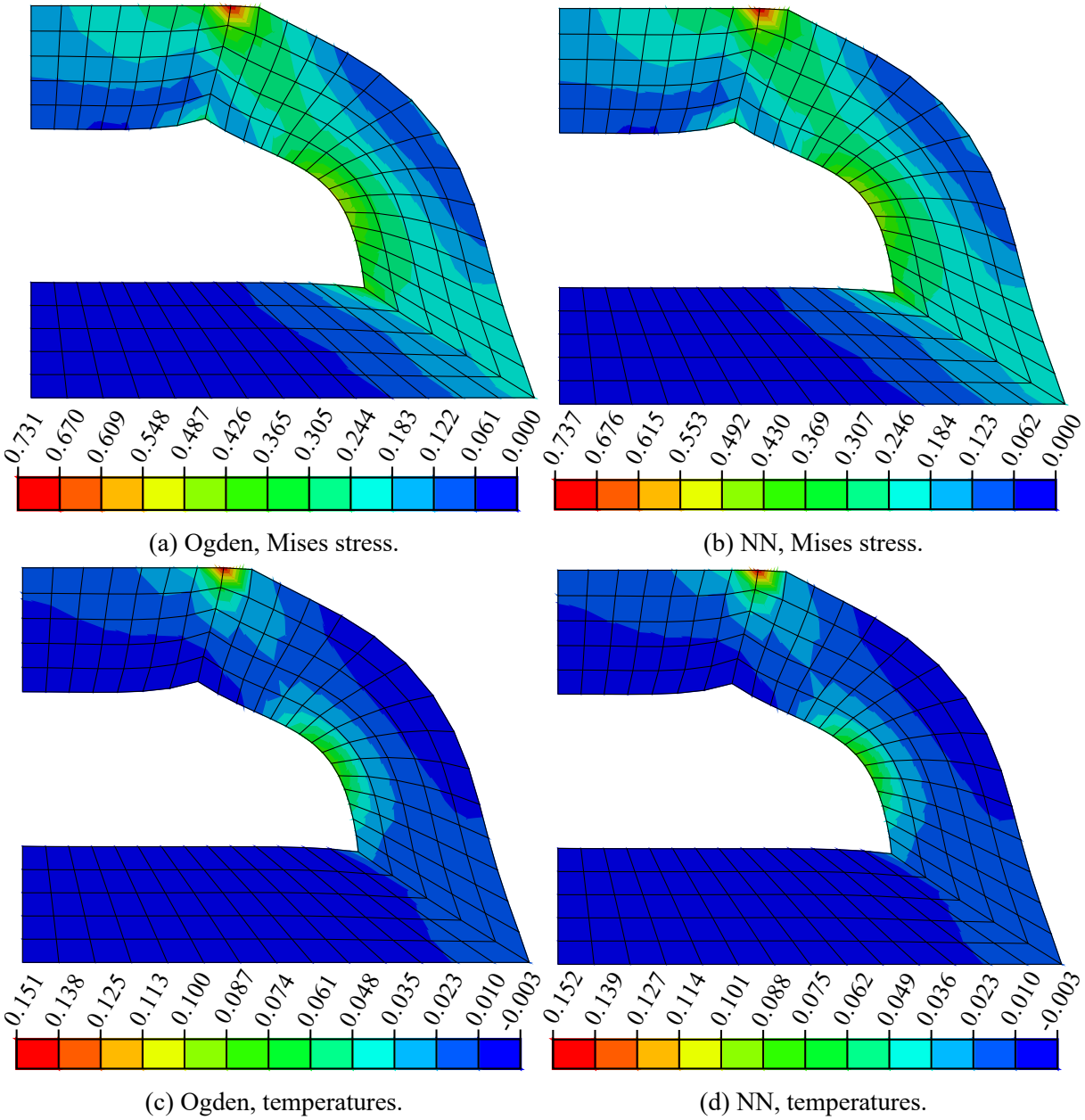


Figure 3.10: Reaction force at the top edge where the displacement is prescribed. An insert is shown of the deformed rubber seal FE mesh with the direction an position of the prescribed vertical direction shown.

The results over the entire FE mesh domain are given in Fig. 3.11 comparing the thermally expanded Ogden model and the thermoelastic NN. The results are in good agreement with very little difference between the stress and temperature distributions.



(a) Ogden, Mises stress.

(b) NN, Mises stress.

(c) Ogden, temperatures.

(d) NN, temperatures.

Figure 3.11: Plots of the von Mises stresses and temperatures over the deformed FE mesh. The thermally expanded Ogden and thermoelastic NN solutions are given side by side for comparison.

In summary, NNs can be used for hyperelastic behaviour and can capture even minute differences in behaviours, as shown by the results so far. This is done with classical NN activation functions without custom loss functions, i.e. everything can be done with already existing tools in machine learning libraries.

3.6 Introducing Invariants as Inputs

Although the results presented so far have shown good agreement with the reference analytical models, the resulting NN models are still classical NNs without taking benefit of any knowledge of solid mechanics. For example, a large database is needed to train the NNs containing many examples of different states of deformation. Additionally, a material is not defined by its stress-strain relation, but the presented 3-60-60-1 or 3-60-60-60-1 NN models simply give us this relation. As a possible improvement, the inputs can be replaced with invariants of the right Cauchy-Green deformation tensor $\mathbf{C}$ instead of using the strain tensor, an example of the invariant domain is given in Fig. 3.12. This would reduce the number of examples needed to train the NN as many states of deformation would be similar or have the same invariants since the invariants do not change with a rotation of a reference system.

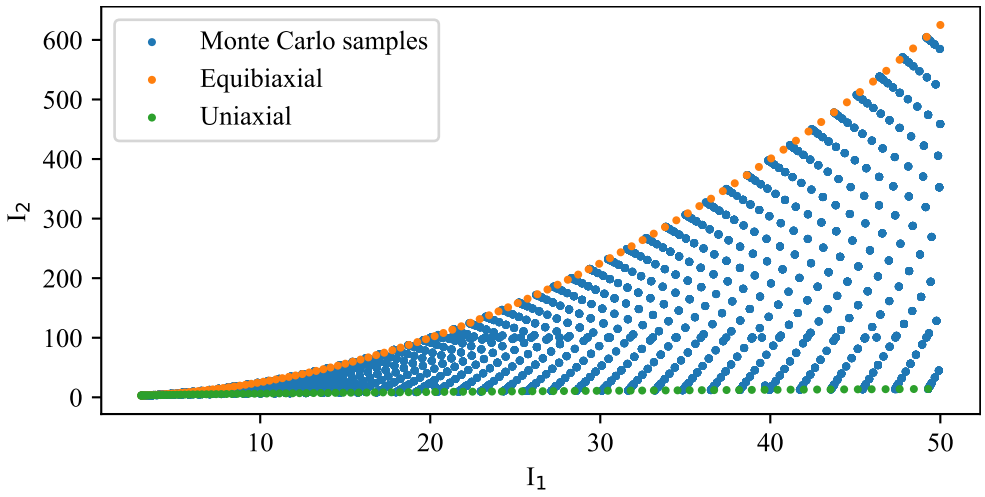


Figure 3.12: Invariant domain for an uniaxial stretch up to $\lambda = 7$, and equibiaxial stretch up to $\lambda = 5$.

Additionally, as shown before the material behaviour is defined through the strain energy or the free energy in case of thermoelasticity. Thus changing the neural network so that it has the invariants of $\mathbf{C}$ as inputs and the free energy ψ_{NN} as output gives an objective function, i.e. it does not change regardless of the change with the change of basis. This means that now only one quantity would need to be predicted, the free energy, reducing the amount of NNs to train to only one. The stress tensor and material tangent matrix can be constructed using the derivatives of the energy w.r.t. the invariants. It can be easily implemented in Abaqus through the existing UHYPER (User HYPERelastic) subroutine. Thus the new invariant based (IB) NN architecture details are as follows:

- invariants used as input, but adapted to be $(I_1 - 3, I_2 - 3)$ so that in the undeformed state the energy is equal to 0
- 2 hidden layers, 60 neurons per layer, referred to as (2-60-60-1) - for both isothermal hyperelastic and thermoelastic behaviour

- no biases are included
- tanh used as the activation function
- Adam optimizer used during training
- Glorot initialisation used for the weights [21]

Again, the hyperbolic tangent was used without biases since for an undeformed state the energy should be zero, i.e. $\psi_{\text{NN}}(\mathbf{I}) = 0$. With the inputs taken as $(I_1 - 3, I_2 - 3)$ and the fact the invariants are equal to 3 in an undeformed state the predicted energy is zero for an undeformed state. When training the IB NN only 30 000 training samples were needed to obtain an accurate NN. This is more than 30 times less data than for the stress-strain (SS) trained NNs presented earlier (NNs 3-60-60-60-1 and 3-60-60-1). Also, the 3-60-60-60-1 NNs had in total 22 320 trainable parameters across the 3 trained NNs, whereas the IB approach 2-60-60-1 NN has only 3 780 trainable parameters which is approximately 5.9 times less. The loss function would now take the form of:

$$L = \frac{1}{N} \sum_{i}^{N} \left(\psi_{\text{NN},i} - \psi_i \right)^2 , \tag{3.19}$$

where $\psi_{\text{NN},i}$ is the energy predicted by the NN for a sample, and ψ_i is the true value known during training.

3.6.1 Sheet with a Hole Numerical Example

To test and compare the performance between the SS and IB NNs other numerical examples are introduced. The first one is the 2D one quarter of a sheet with a hole used in [40, 55, 69]. The geometry and boundary conditions of the problem are given in Fig. 3.13. Symmetry boundary conditions are applied to the left and bottom edge, and the displacement u of 100 mm is prescribed at the right edge. The mesh consists of 8×12 quadrilateral FEs, plane stress conditions are assumed.

In Fig. 3.14 the reaction force is shown at the right edge where the displacement was prescribed. The results are for the SS NNs, but the differences are minute and the IB approach would give the same curve. In order to compare the 2 approaches in more detail plots over the entire domain may prove to be more informative.

Fig. 3.15 shows the temperature change at the end of the simulation and it is from the SS NN. It illustrates the magnitude of the temperature change that occurs. The temperature errors are shown in Fig. 3.16. The upper figures show the relative errors compared to the thermodynamically expanded Ogden reference model. In Fig. 3.16a it can be seen that the maximum errors reach 30% in an area where the temperature values are moderate, i.e. between the maximum and minimum temperature values. This is counterintuitive as the largest errors are expected at the extremes. This may indicate that although a large database was used to train the stress-strain

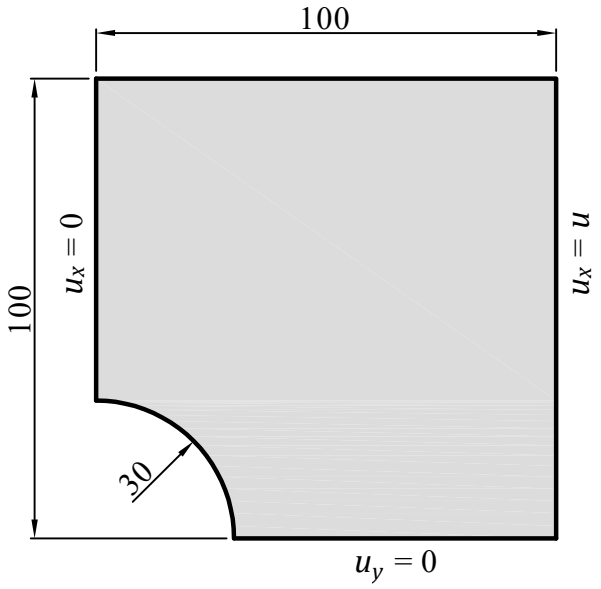


Figure 3.13: Geometry of the sheet with a hole in the middle. One quarter shown, symmetry boundary conditions are applied.

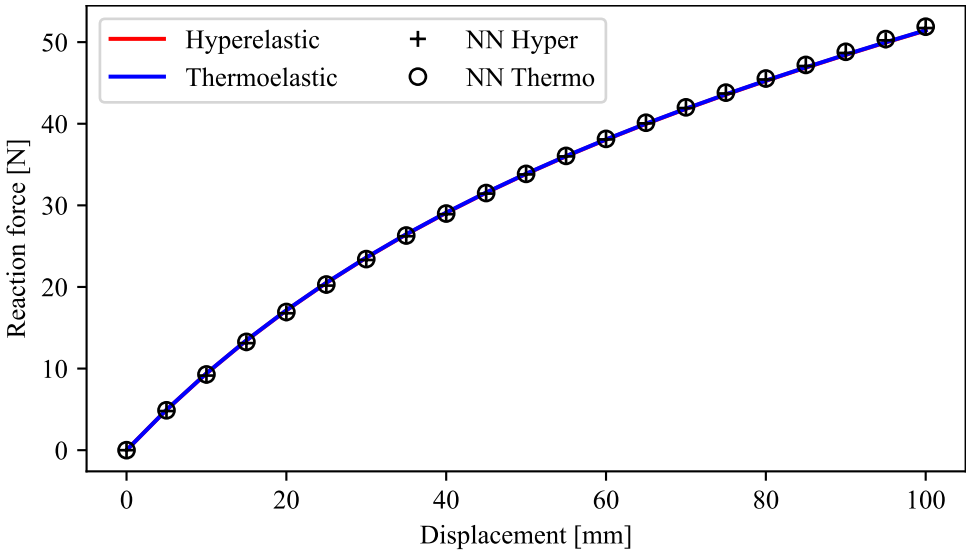


Figure 3.14: Reaction force at right edge where the displacement is prescribed.

neural networks it is not sufficient to obtain consistently accurate results. On the other hand, Fig. 3.16b shows the relative errors of the invariant based neural network with the largest error of 2.9% at the bottom edge where the temperature change values are the smallest. This means that the highest relative error is achieved in the area where the absolute values are the smallest, so that small absolute errors lead to large relative errors, which is to be expected. This is also confirmed when looking at the absolute errors of the IB NN in Fig. 3.16c, where it can be seen that the absolute errors are smallest at the bottom edge, where the relative errors are largest. This example leads to the conclusion that in addition to the fact that IB NNs are numerically less burdening they are also more accurate and require less data to be trained on. This is still one example so further investigation is necessary.

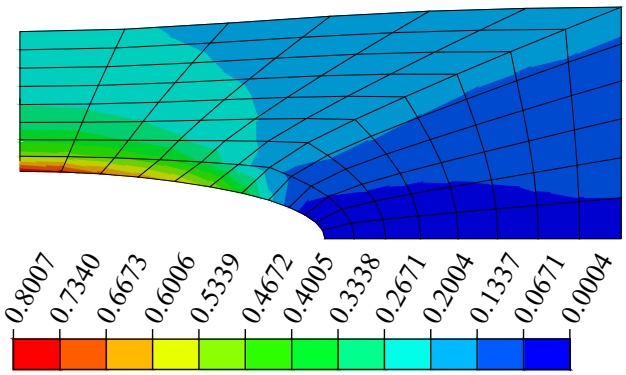
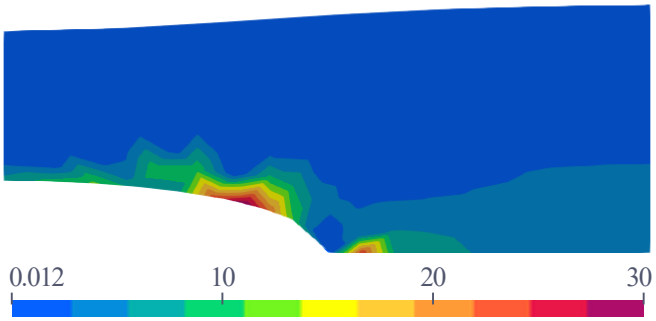
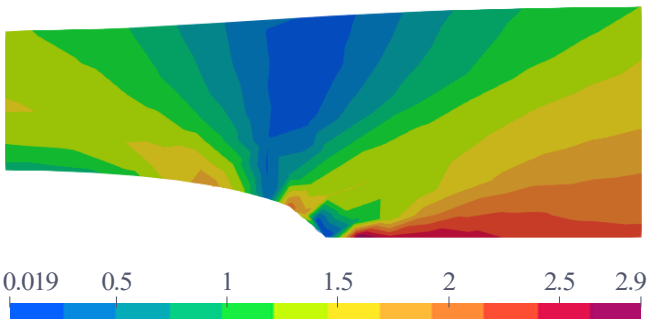


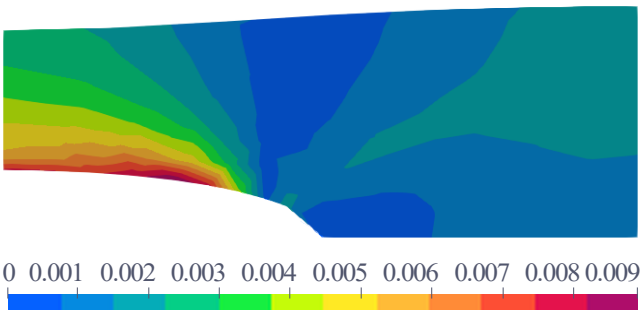
Figure 3.15: Temperature change [K], results obtained using the SS (3-60-60-60-1) NN.



(a) SS, relative error [%].



(b) IB, relative error [%]



(c) IB, absolute error [K].

Figure 3.16: Relative errors of the SS (3-60-60-60-1) and IB (2-60-60-1) NNs shown in the top 2 figures. The bottom figure presents the absolute error of the IB NN.

3.6.2 Punch Problem Numerical Example

The second additional numerical example is the 2D punch problem benchmark [56, 67], a 3D version is also available [61]. The geometry is shown in Fig. 3.17. Horizontal displacement is constrained on the left and top edge, while vertical displacements are constrained on the bottom edge. Half of the top edge is loaded with a downward facing distributed load q which is set to 2 N/mm. The problem is different from the other problems with half the domain being compressed whilst the other half can freely expand. Also, it is the first numerical example where the loading is compressive.

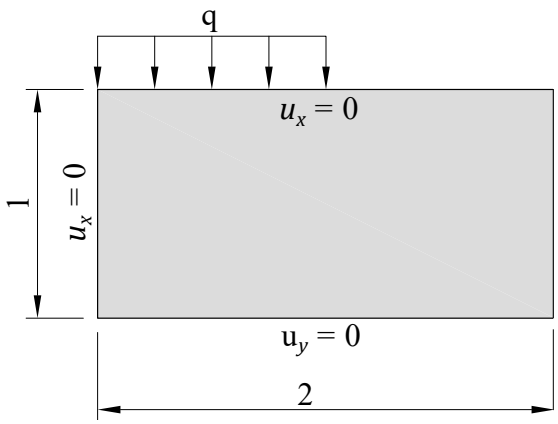
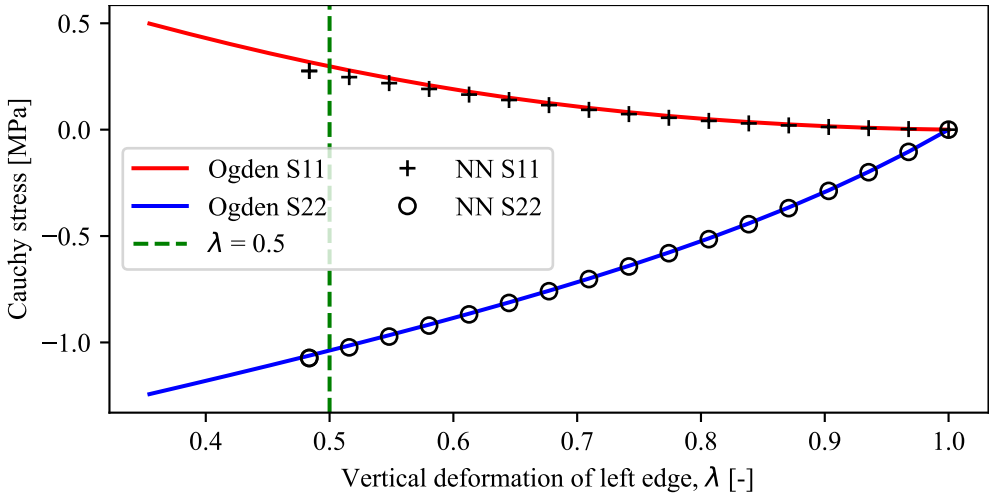


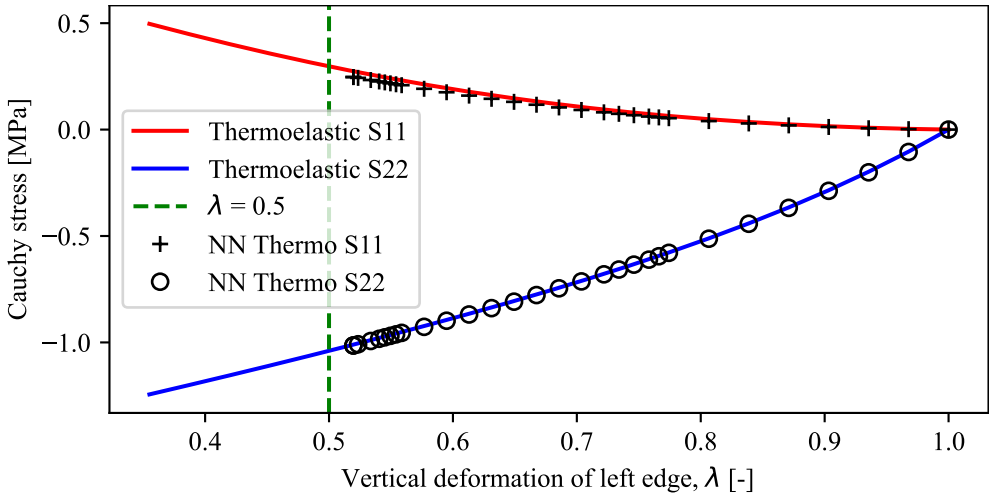
Figure 3.17: Geometry of the 2D punch problem.

Fig. 3.18a shows the evolution of the σ_x and σ_y (S11 and S22 in Abaqus) at the bottom left corner using isothermal hyperelastic behaviour. The horizontal axis shows the stretch λ, i.e. the deformation of the left edge of the domain. An interesting event occurs where the NN driven simulation does not fully complete the calculation but breaks shortly after passing the 50% compression mark that is where $\lambda = 0.5$. Coincidentally that is also the lower bound of the generated training data. This shows the expected behaviour from NNs where they cannot correctly predict values outside of the range they were trained on which is also demonstrated here. However, the solutions are accurate until that point. When adding thermoelastic behaviour the quality only further decreases and the analysis breaks a bit before the value of $\lambda = 0.5$ as shown in Fig. 3.18b. However, looking at the postprocessed results for the temperature change in Fig. 3.20 the material behaviour has been correctly captured.

On the other hand, the results of the IB NN are quite opposite as shown in Fig. 3.21 where the analysis not only finished, but has done so without any numerical difficulties unlike Fig. 3.18b where step times were cut back. This is attributed to the fact that the IB NN is an objective function that takes the invariants as input and thus has effectively seen the data but in the invariant domain shown in Fig. 3.12, where the uniaxial and biaxial deformation modes bound the entire domain, and the lowest value of invariants in the incompressible case occur in the undeformed state when $I_1 = I_2 = 3$. This is a major highlight of the IB NN since they can be trained on one set of data which may not be rich in different modes of deformation, but nevertheless adequately

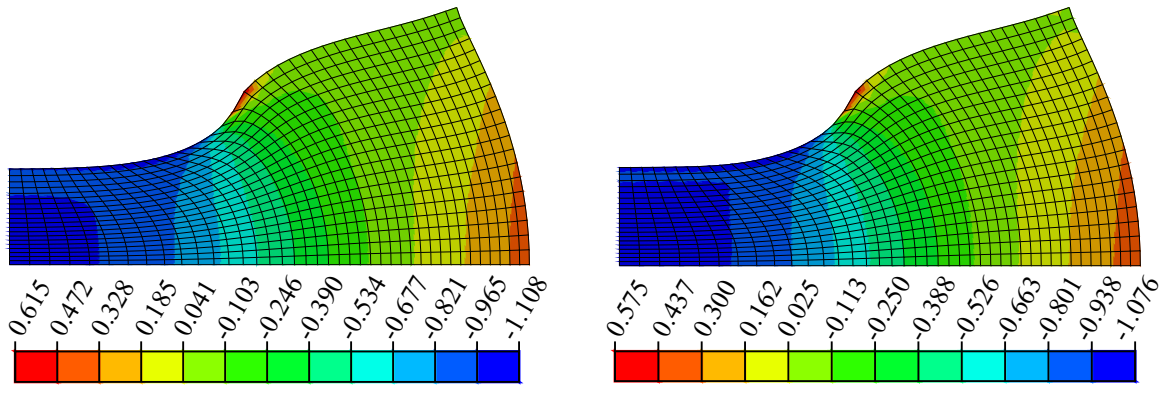


(a) hyperelastic (3-60-60-1).



(b) Thermoelastic (3-60-60-60-1).

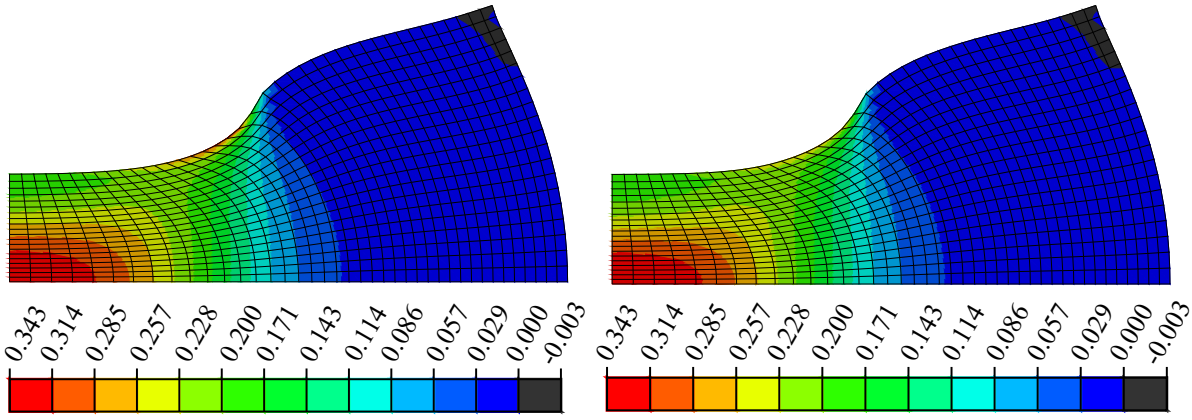
Figure 3.18: Stress evolution at the bottom left corner of the punch problem. Only results for the SS (3-60-60-1) and (3-60-60-60-1) NNs shown. The vertical green line represents the lower bound of the generated training data.



(a) Thermoelastic Ogden, σ_y [MPa].

(b) Thermoelastic SS NN, σ_y [MPa].

Figure 3.19: Distribution of σ_y (S22) stresses at the point where the analysis breaks for the SS NN model.



(a) Thermoelastic Ogden, ϑ [K]. (b) Thermoelastic SS NN, ϑ [K].

Figure 3.20: Distribution of temperature changes ϑ at the point where the analysis breaks for the SS NN model.

captures material behaviour outside of the given deformation modes as shown in Fig. 3.21. Additionally, the stress distribution over the domain is given in Fig. 3.22 where the solutions can be directly compared and it can be confirmed that the material behaviour is correctly captured.

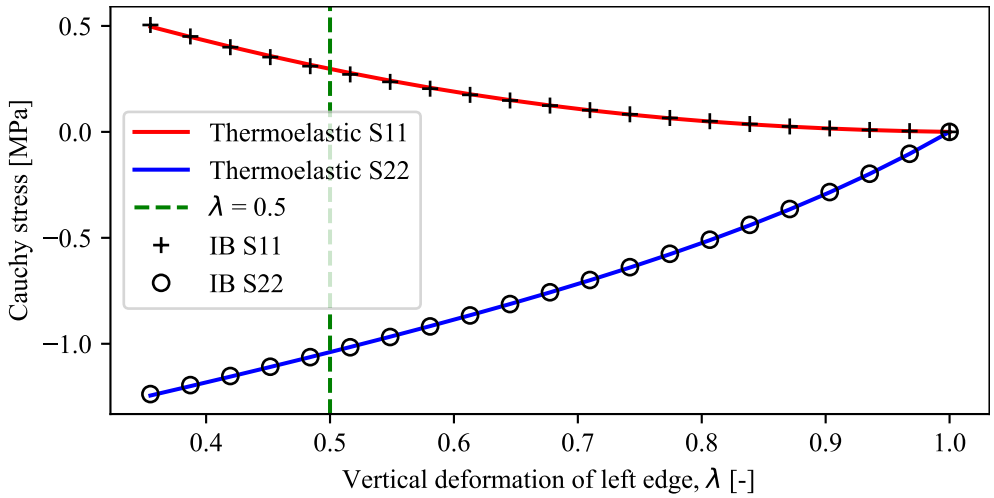
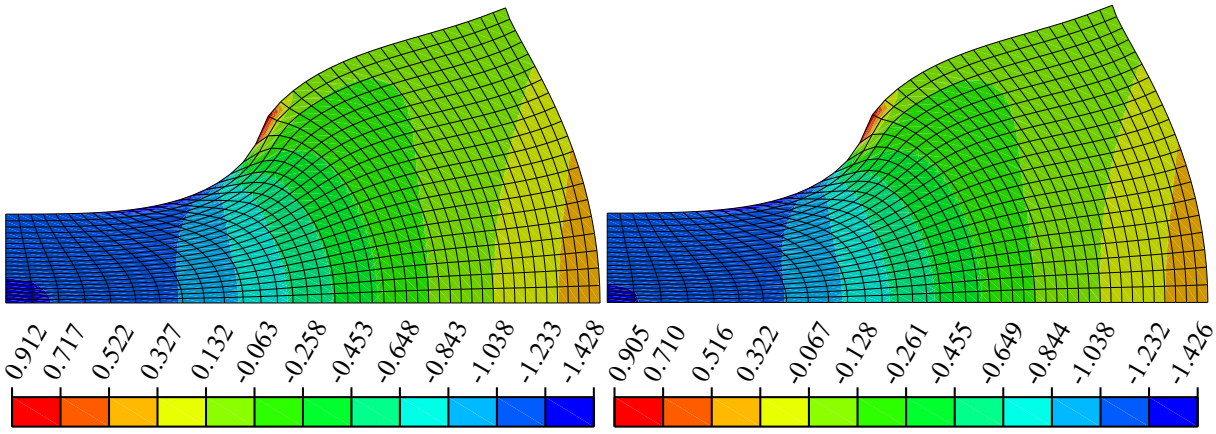


Figure 3.21: Stress evolution at the bottom left corner of the punch problem. Results of the IB (2-60-60-1) NN shown. The vertical green line represents the lower bound of the generated training data.

So far all the numerical examples were calculated under the assumption of plane stress conditions. This is indeed a limiting factor for the SS NN model since it cannot predict outside of the data it was trained on. The neural network based on invariants however is not bound by such limitations as the predicted energy and invariants are not restricted by the assumption of plane stress in the same manner the strain or stress tensors are.



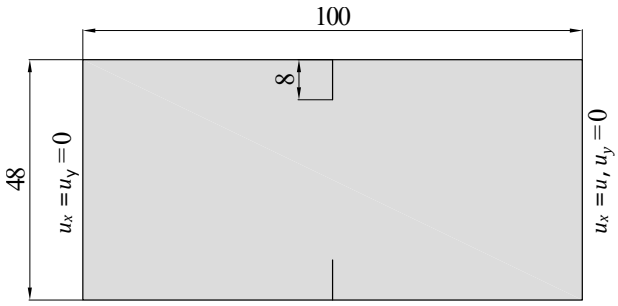
(a) Thermoelastic Ogden, σ_y [MPa]. (b) Thermoelastic IB NN, σ_y [MPa].

Figure 3.22: Distribution of σ_y (S22) stresses at the end of analysis with the IB NN model.

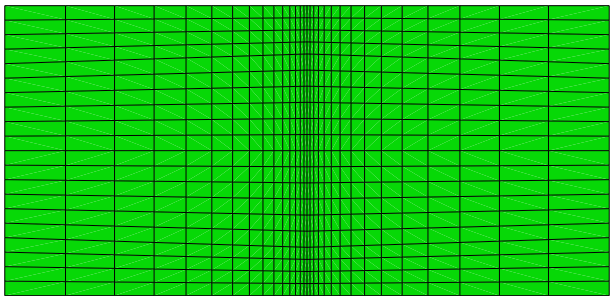
3.6.3 Cracked Bar - a 3D Numerical Example

In order to test the behaviour of the IB model in a very different type of analysis, a full 3D example with a hybrid finite element formulation is studied. This is also the last numerical example in this section and it was taken from [26], the cracked bar example. It consists of a tensile test of a bar with cracks added at the middle, the geometry is shown in Fig. 3.23a. The FE mesh is shown in Fig. 3.23b, only one eighth was modelled as in the original paper, and symmetry boundary conditions were imposed. The model here is treated as an incompressible one while the original one presented in [26] was treated as compressible. The prescribed load u shown on Fig. 3.23a was set to 175 mm. Results for the reaction force, von Mises stress and temperature change are given in Fig. 3.24. In Fig. 3.24a the characteristic S shape can seen, similar to the one from a simple uniaxial test in Fig. 3.4a although less pronounced since the stretch of the bar at the centre node is not as high. Also in Fig. 3.24c at the beginning the small inversion effect can be seen, the same one shown in Fig. 3.3. Looking more closely at the crack and comparing with the original work of [26], the maximum stretch was 796% which agrees with the result from the original of about 800%. The highest temperature change was 6.88 K which is a difference of 9.6% from the original one of 6.21 K. At the outer ends of the crack convex curvatures occur which also agrees with the original work, this can be seen in Fig. 3.25.

In summary, neural networks are capable of correctly capturing even very small differences in material behaviour. Using conventional architectures and readily available activation functions from well established machine learning libraries was sufficient to reproduce the desired behaviour with excellent agreement. However, adopting the invariant based neural network opens up other possibilities. If simply considering invariants to fulfil objectivity has greatly improved the performance of the NN then how much would the consideration and introduction of other existing knowledge from solid mechanics impact the behaviour of NNs when applied to hyperelasticity? This is the area of investigation in the rest of this thesis.

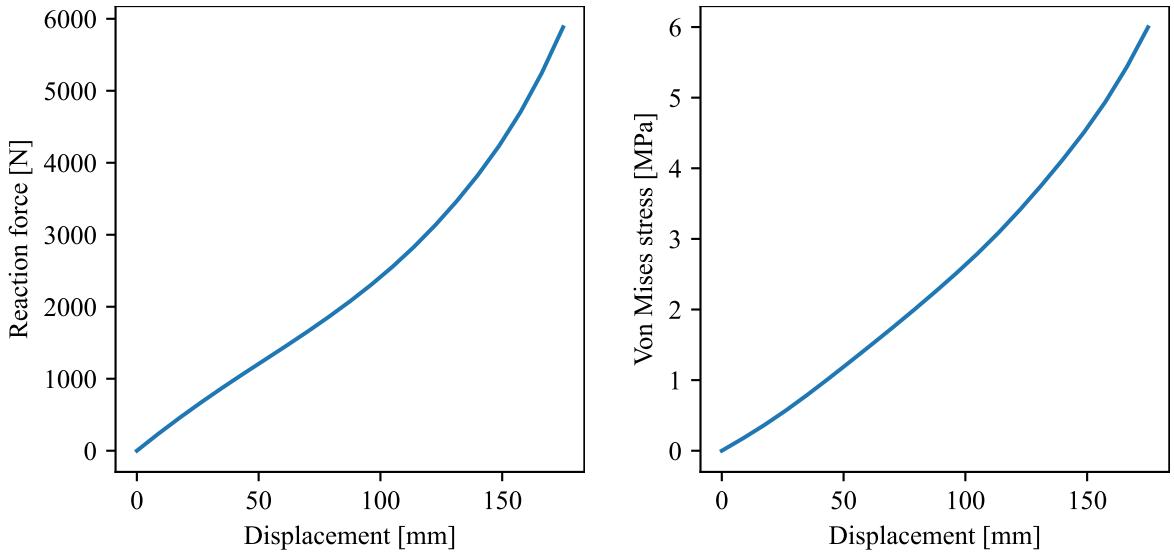


(a) Geometry.

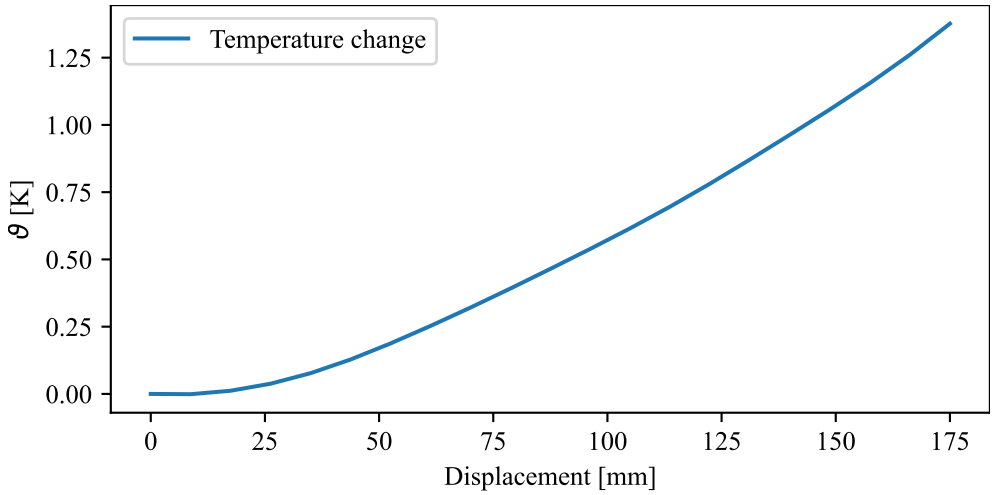


(b) FE mesh.

Figure 3.23: Geometry and FE mesh of the cracked bar example, thickness perpendicular to the plane is 10 mm. The mesh consists of 3D hybrid formulation (type C3D8H in Abaqus) finite elements.



(a) (b)



(c)

Figure 3.24: Diagrams of the reaction force (a), von Mises stress (b) and temperature change evolution (c) for the cracked bar. Results of the stress and temperature evolution are shown for the centre node of the cracked bar (not at the crack tip).

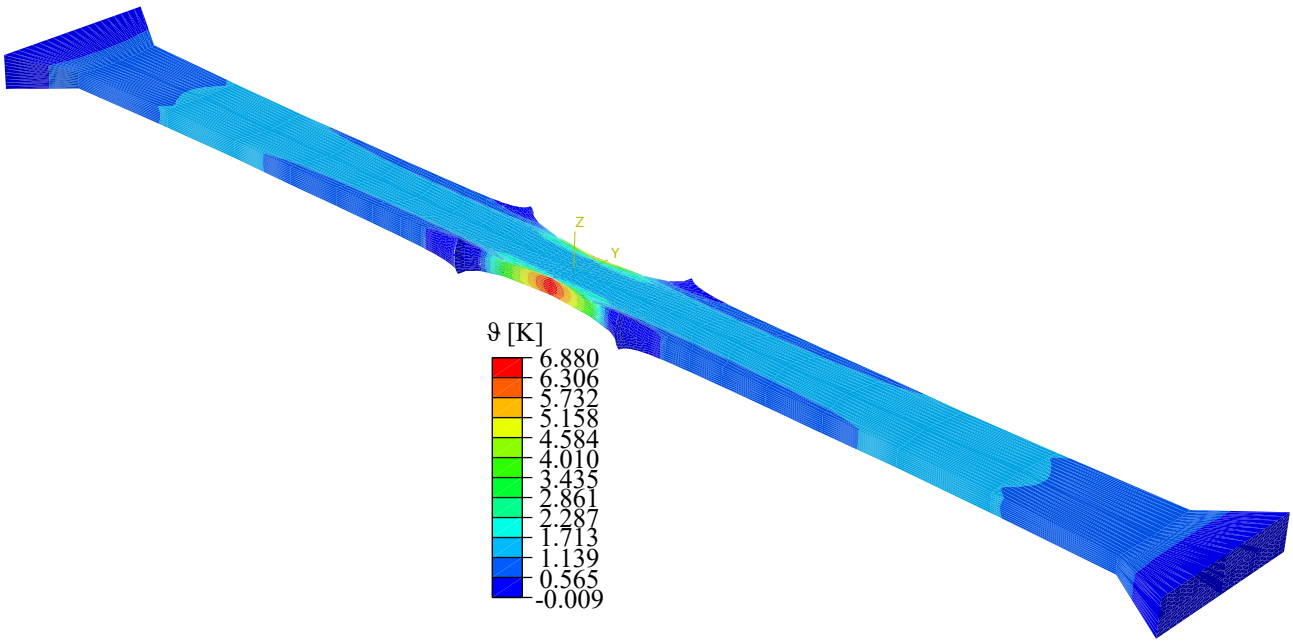


Figure 3.25: Temperature change distribution for the deformed cracked bar.

4 Physically Augmented Neural Networks for Hyperelasticity

In order to answer the question which was posed at the end of the previous section the concept of a Physics-Augmented Neural Network (PANN) is introduced, the term was suggested by [44]. This type of NN implies that the architecture of the NN has been changed in such a way that it satisfies a number of conditions from solid mechanics in an exact way. Some of these conditions might be:

1. *thermodynamic consistency*
2. *symmetry of the stress tensor*
3. *objectivity*
4. *normalisation of energy*
5. *normalisation of stress*
6. *non-negativity of strain energy*
7. *polyconvexity*
8. *growth condition*

These conditions can vary given that different assumptions can be made during analysis or modelling, e.g. the *symmetry of the stress tensor* is meaningful only if a symmetric stress tensor is used, the 1^{st} Piola-Kirchhoff tensor is not symmetric by default while the 2^{nd} Piola-Kirchhoff or Cauchy tensors are, the *growth condition* is associated with compressibility and it does not exist if incompressibility is assumed, and so on.

Thermodynamic consistency is satisfied by fulfilling Eq. (2.75), i.e. the stress tensor should be derived from the energy with respect to a deformation measure. If invariants of the right Cauchy-Green tensor are employed to obtain the 2^{nd} Piola-Kirchhoff stress tensor $\mathbf{S}$ from the energy predicted by the NN then Eq. (2.81) can simply be rewritten to

$$\begin{aligned}\mathbf{S} &= 2\,\frac{\partial \psi_{\text{NN}}(I_1, I_2, I_3)}{\partial \mathbf{C}} = 2\left[\frac{\partial \psi_{\text{NN}}}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{C}} + \frac{\partial \psi_{\text{NN}}}{\partial I_2}\frac{\partial I_2}{\partial \mathbf{C}} + \frac{\partial \psi_{\text{NN}}}{\partial I_3}\frac{\partial I_3}{\partial \mathbf{C}}\right] = \\ &= 2\left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + I_1\frac{\partial \psi_{\text{NN}}}{\partial I_2}\right)\mathbf{I} - \frac{\partial \psi_{\text{NN}}}{\partial I_2}\mathbf{C} + I_3\frac{\partial \psi_{\text{NN}}}{\partial I_3}\mathbf{C}^{-1}\right],\end{aligned} \tag{4.1}$$

or in the incompressible case in the form

$$\mathbf{S} = 2\left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + I_1\frac{\partial \psi_{\text{NN}}}{\partial I_2}\right)\mathbf{I} - \frac{\partial \psi_{\text{NN}}}{\partial I_2}\mathbf{C}\right] - p\mathbf{C}^{-1}, \tag{4.2}$$

which will be used later throughout this work when training the NNs.

It should be noted that the principal invariants of $\mathbf{C}$ can be defined in terms of the deformation gradient $\mathbf{F}$:

$$I_1 = \mathbf{F} : \mathbf{F}, \quad I_2 = \frac{1}{2}\big[I_1^2 - \underbrace{\big(\mathbf{F}^T\mathbf{F}\big) : \big(\mathbf{F}^T\mathbf{F}\big)}_{\text{tr}(\mathbf{C}^2)=\mathbf{C}:\mathbf{C}}\big], \quad I_3 = \det(\mathbf{F}^T\mathbf{F}) = J^2, \tag{4.3}$$

so that Eq. (2.75) can also be defined in terms of the principal invariants of $\mathbf{C}$, as was implemented in [46]. Since the derivatives can be calculated as

$$\frac{\partial I_1}{\partial \mathbf{F}} = 2\mathbf{F}, \quad \frac{\partial I_2}{\partial \mathbf{F}} = 2[I_1\mathbf{F} - \mathbf{F}\mathbf{F}^T\mathbf{F}], \quad \frac{\partial I_3}{\partial \mathbf{F}} = 2I_3\mathbf{F}^{-T}, \tag{4.4}$$

then $\mathbf{P}$ can be defined as

$$\begin{aligned} \mathbf{P} = \ & \frac{\partial \psi_{\text{NN}}(I_1, I_2, I_3)}{\partial \mathbf{F}} = \frac{\partial \psi_{\text{NN}}}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{F}} + \frac{\partial \psi_{\text{NN}}}{\partial I_2}\frac{\partial I_2}{\partial \mathbf{F}} + \frac{\partial \psi_{\text{NN}}}{\partial I_3}\frac{\partial I_3}{\partial \mathbf{F}} = \\ & = 2\left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + I_1\frac{\partial \psi_{\text{NN}}}{\partial I_2}\right)\mathbf{F} - \frac{\partial \psi_{\text{NN}}}{\partial I_2}\mathbf{F}\mathbf{F}^T\mathbf{F} + I_3\frac{\partial \psi_{\text{NN}}}{\partial I_3}\mathbf{F}^{-T}\right], \end{aligned} \tag{4.5}$$

or in the incompressible case

$$\mathbf{P} = 2\left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + I_1\frac{\partial \psi_{\text{NN}}}{\partial I_2}\right)\mathbf{F} - \frac{\partial \psi_{\text{NN}}}{\partial I_2}\mathbf{F}\mathbf{F}^T\mathbf{F}\right] - p\mathbf{F}^{-T}. \tag{4.6}$$

Thermodynamic consistency is fulfilled since the model is trained on its derivatives using auto-differentiation. In that way the loss function takes the form

$$L(\mathbf{S}_{\text{NN}}, \mathbf{S}) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{S}_{\text{NN},i} - \mathbf{S}_i\right)^2, \tag{4.7}$$

where tensors with 6 independent components are used, unlike before when in Eq. (3.17) only an individual stress component was used in the loss function.

Symmetry of the stress tensor is not necessarily needed to be fulfilled since different stress tensors can be used in the loss. E.g., in [46] the 1st Piola-Kirchhoff stress tensor is used which is not symmetric. In this work the 2nd Piola-Kirchhoff stress tensor is used which is a symmetric tensor and this is guaranteed through construction, see Eq. (4.2).

Objectivity is fulfilled by using the invariants of the right Cauchy-Green deformation tensor as a measure of deformation since these are principal values of a characteristic polynomial and thus not affected by arbitrary rotations of the reference. An example of a body that is simply rotated after being deformed is shown in Fig. 4.1.

Normalisation of energy means that for an undeformed state the strain energy should be equal to zero. This was briefly mentioned previously when describing the choice of the inputs for the invariant based NN. This can be satisfied by choosing an appropriate activation function and by

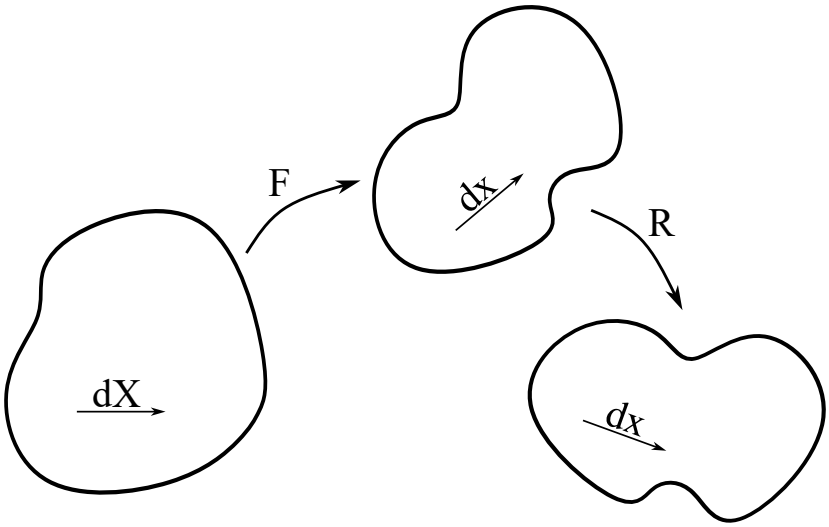


Figure 4.1: An arbitrary body is deformed and then rotated. The deformation itself is not affected by the rotation. An objective strain energy should not give a different results for a rotated body.

adapting the inputs. It is known that for an undeformed state the invariants I_1 and I_2 are equal to 3, therefore when using them as inputs to the NN they were subtracted by 3, i.e. (I_1-3, I_2-3), and when passed to the tanh activation function this would yield zero for an undeformed state. In [45] many activation functions were proposed for modelling hyperelasticity, of which the exponential linear unit was chosen as a possible candidate for an activation function instead of the conventional tanh, sigmoid or ReLU:

$$h(x) = e^{\alpha x} - 1, \tag{4.8}$$

where α is a trainable parameter. In [46] the function was proposed and used without the trainable parameter α being a part of the activation unction. The linear exponential function is used in different forms when modelling hyperelasticity and also satisfies the normalisation condition by using the adapted input (I_1-3, I_2-3) since for an undeformed state $h(0) = 0$. To satisfy the normalisation of energy biases should be omitted in the NN architecture. A visual illustration of the normalized and non-normalized functions is given in Fig. 4.2.

Normalisation of stress means that for an undeformed state there should be no stress in the body. In the incompressible case this is *a priori* satisfied from the Lagrange multiplier p. The stress tensor **S** can be expressed in the spectral form as

$$\mathbf{S} = \sum_{i=1}^{3} S_i \mathbf{N}_i \otimes \mathbf{N}_i, \tag{4.9}$$

with S_i being the principal stress. It can be expressed as

$$S_i = 2\left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + I_1 \frac{\partial \psi_{\text{NN}}}{\partial I_2}\right) - \frac{\partial \psi_{\text{NN}}}{\partial I_2}\lambda_i^2\right] - p\lambda_i^{-2}, \tag{4.10}$$

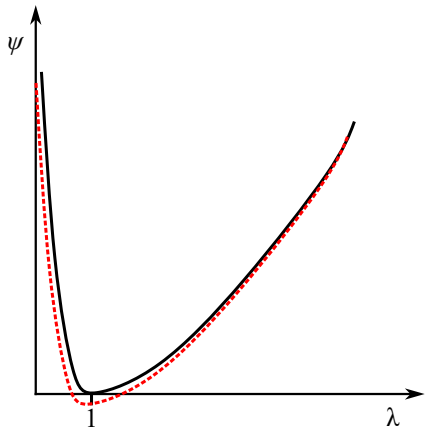


Figure 4.2: A non-negative function normalized at the undeformed state ($\mathbf{F} = \mathbf{I}$) is shown in black, while an undesirable function with negative values that is not normalized at the undeformed state is shown in dashed red. The stretch λ is shown on the horizontal axis and the strain energy ψ is shown on the vertical axis.

and using Eq. $(2.15)_1$ and the plane stress condition $S_3 = 0$, the Lagrange multiplier p can be expressed in the isotropic case under plane stress conditions as

$$p = 2\lambda_3^2 \left[\frac{\partial \psi}{\partial I_1} + \left(\lambda_1^2 + \lambda_2^2\right) \frac{\partial \psi}{\partial I_2} \right]. \tag{4.11}$$

Combining Eqs. (4.11) and (4.2) and assuming the undeformed state where $\mathbf{C} = \mathbf{I}$ (and by extension $\lambda_i = 1$) leads to the following

$$\mathbf{S}_{|\mathbf{C}=\mathbf{I}} = 2 \left[\left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + 3 \frac{\partial \psi_{\text{NN}}}{\partial I_2} \right) \mathbf{I} - \frac{\partial \psi_{\text{NN}}}{\partial I_2} \mathbf{I} \right] - \underbrace{2 \left(\frac{\partial \psi}{\partial I_1} + 2 \frac{\partial \psi}{\partial I_2} \right)}_{p} \mathbf{I} = \mathbf{0}, \tag{4.12}$$

thus showing that for the incompressible case the normalisation of stress is always satisfied in an exact way. For completeness the necessary treatment is presented in case of compressibility. For the compressible case a stress normalisation term is needed to satisfy $\mathbf{S}(\mathbf{I}) = \mathbf{0}$. As presented in the works of [33, 38, 44] an additional term for the stress correction in the form of

$$\psi_{\text{str}} = -\mathfrak{n}(J - 1) \tag{4.13}$$

is added to the energy calculated by the NN with the constant $\mathfrak{n}$ defined as

$$\mathfrak{n} = 2 \left(\frac{\partial \psi_{\text{NN}}}{\partial I_1} + \frac{\partial \psi_{\text{NN}}}{\partial I_2} + \frac{\partial \psi_{\text{NN}}}{\partial I_3} + \frac{\partial \psi_{\text{NN}}}{\partial I_3^*} \right) \Bigg|_{\mathbf{F}=\mathbf{I}}, \tag{4.14}$$

where I_3^* is an additional polyconvex invariant defined as $I_3^* = -2\sqrt{I_3} = -2J$ introduced to represent negative stresses.

An example illustration for a normalized and non-normalized stress response is shown in Fig. 4.3, with the stretch $\lambda = 1$ representing the undeformed state $\mathbf{F} = \mathbf{I}$.

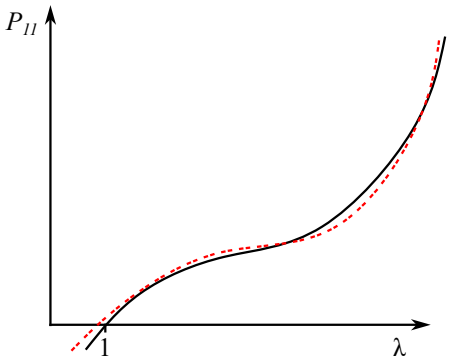


Figure 4.3: Illustration for the stress normalization during uniaxial tension. The black curve represents a normalized stress function where stress is zero for the undeformed state ($\mathbf{F} = \mathbf{I}$). The dashed red curve represents a non-normalized curve, i.e. a non-zero stress in the undeformed state. The stretch λ is shown on the horizontal axis and the 1^{st} Piola-Kirchhoff stress is shown on the vertical axis.

Non-negativity of strain energy means that the strain energy can not be negative during the deformation process, i.e. $\psi(\mathbf{F}) \geq 0$. This is ensured by using the exponential linear unit as the activation function since it is zero for an undeformed state, and since in an incompressible setting the smallest value of the invariants is 3 the exponent is strictly non-negative, if the values of the previous weights $w_{1,i}^{[1]}$ and $w_{2,i}^{[1]}$ are enforced to be non-negative. An example of a non-negative strain energy response is shown in Fig. 4.2 with a black curve, while the dashed red curve shows a similar energy response that contains negative values.

Polyconvexity is a condition that ensures the existence of at least one minimum of energy, i.e. at least one possible equilibrium solution. It is proposed in [4] in order to replace the condition of quasi-convexity proposed by [31]. The condition of polyconvexity is thoroughly discussed in [24], where exponential functions are also given as possible polyconvex functions that can be implemented for modelling hyperelasticity. It can be stated that a function is polyconvex if it is convex in $(\mathbf{F}, \text{adj}(\mathbf{F}), \det \mathbf{F})$, illustration given in Fig. 4.4. Note that the first and second invariants can be expressed as $I_1 = ||\mathbf{F}||^2$ and $I_2 = ||\text{adj}(\mathbf{F})||^2$, hence they themselves are convex in $\mathbf{F}$ and $\text{adj}(\mathbf{F})$.

The growth condition is presented for the sake of completeness and is only needed in case of compressibility. The growth condition requires the following:

$$\psi(\mathbf{F}) \to \infty \quad \text{as} \quad \det \mathbf{F} \to 0^+ \quad \text{or} \quad \det \mathbf{F} \to \infty. \tag{4.15}$$

This can be simply satisfied by adding a growth term to the energy which can be as simple as

$$\psi(J) = \left(J + \frac{1}{J} - 2 \right)^2, \tag{4.16}$$

which also satisfies polyconvexity. Other growth terms can be used, some are presented and investigated in regards to the polyconvexity condition in [24]. The growth of energy towards

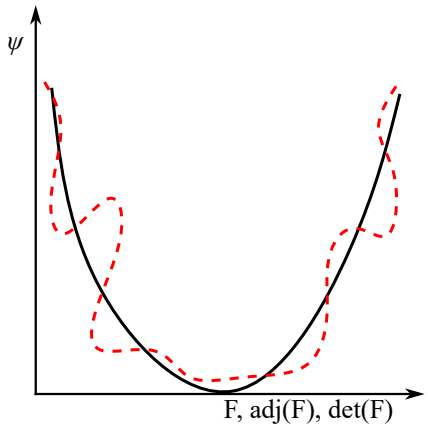


Figure 4.4: Illustration of a polyconvex black curve that is convex in $(\mathbf{F}, \text{adj}(\mathbf{F}), \det \mathbf{F})$ and a non-polyconvex dashed red curve that is not convex in the same arguments.

infinity is shown in Fig. 4.2 where the energy starts significantly increasing as the stretch λ tends towards zero or infinity.

The proposed neural network has the same base architecture as the FNN shown in Fig. 3.1 without the biases, with the input layer consisting of the invariants (I_1-3, I_2-3), the activation functions

$$h_i(I_1-3, I_2-3) = \exp\left\{\alpha_i\left[w_{1,i}^{[1]}(I_1-3) + w_{2,i}^{[1]}(I_2-3)\right]\right\} - 1, \tag{4.17}$$

with α_i being the additional trainable parameter within each neuron, and the expression for the output strain energy can be written as

$$\psi_{\text{NN}} = \sum_{i=1}^{n} w_{i,1}^{[2]} h_i. \tag{4.18}$$

Having included all the proposed changes into the architecture of the NN it would take the form shown in Fig. 4.5. In the figure the NN consists of 5 neurons but the actual number of neurons can be arbitrarily chosen. The network is referred to as LINEXP-PANN (LINear Exponential Physics-Augmented Neural Network.)

The 1st order derivatives of ψ_{NN} are needed for training the NN, and the 2nd order derivatives are needed for the implementation into Abaqus. They are easily obtained as

$$\frac{\partial \psi_{\text{NN}}}{\partial I_j} = \sum_{i=1}^{n} w_{i,1}^{[2]} \frac{\partial h_i}{\partial I_j}, \quad \frac{\partial^2 \psi_{\text{NN}}}{\partial I_j \partial I_k} = \sum_{i=1}^{n} w_{i,1}^{[2]} \frac{\partial^2 h_i}{\partial I_j \partial I_k}, \quad j,k = 1,2, \tag{4.19}$$

and the partial derivatives of the activation functions are further obtained as

$$\begin{aligned} \frac{\partial h_i}{\partial I_j} &= \alpha_i w_{j,i}^{[1]} \exp\left\{\alpha_i\left[w_{1,i}^{[1]}(I_1-3) + w_{2,i}^{[1]}(I_2-3)\right]\right\}, \\ \frac{\partial^2 h_i}{\partial I_j \partial I_k} &= \alpha_i^2 w_{j,i}^{[1]} w_{k,i}^{[1]} \exp\left\{\alpha_i\left[w_{1,i}^{[1]}(I_1-3) + w_{2,i}^{[1]}(I_2-3)\right]\right\}. \end{aligned} \tag{4.20}$$

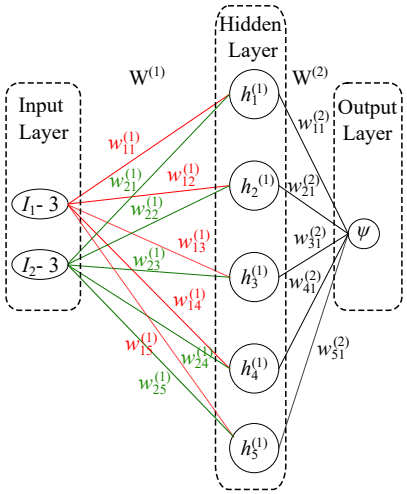


Figure 4.5: Illustration of the proposed NN architecture referred to as LINEXP-PANN.

It should be noted that a composition of functions is convex if all the functions are convex and non-decreasing. Also, a sum of convex functions is a convex function. From Eq. (4.20) it can be seen that the activation function is polyconvex (i.e. convex in $\mathbf{F}$ and cof($\mathbf{F}$))as long as the weights $w_{1,i}^{[1]}$, $w_{2,i}^{[1]}$ and the trainable parameters α_i are positive. Additionally, since the exponential function is convex and non-decreasing then a composition of functions such as e^{I_1} or e^{I_2} is also a polyconvex function. Finally, if the weights $w_{i,1}^{[2]}$ from Eq. (4.18) are positive then the strain energy ψ_{NN} predicted by the neural network is polyconvex.

The chosen LINEXP-PANN for modelling ordinary hyperelastic materials consists of 5 neurons in the hidden layer. The NN model contains a total of 20 parameters that can be trained. Unlike the model in Sec. 3.3, this model is trained solely on simpler tests such as in [65], i.e. uniaxial tension, equibiaxial tension and planar tension (pure shear).

The invariant space it was trained on is presented in Fig. 4.6, the data are gathered only from the curves representing each deformation mode thus the training space is rather sparse compared to the one from Sec. 3.3 shown in Fig. 3.12.

The network is trained on data generated artificially from several material models to mimic different materials, namely the neo-Hookean, Mooney-Rivlin and Ogden models were used. The coefficients for the neo-Hookean and Mooney-Rivlin models were taken from [63]. The neo-Hookean is the simplest one with linear dependence on only the first invariant I_1:

$$\psi_{\text{NH}} = \frac{\mu}{2}\left(I_1 - 3\right), \tag{4.21}$$

where μ is the shear modulus taken as 0.525 MPa. The Mooney-Rivlin model is a bit more complex with the inclusion of the second invariant I_2:

$$\psi_{\text{MR}} = c_{10}\left(I_1 - 3\right) + c_{01}\left(I_2 - 3\right), \tag{4.22}$$

with the coefficients $c_{10} = 0.2659$ MPa and $c_{01} = -0.0017$ MPa. Note that this material is not

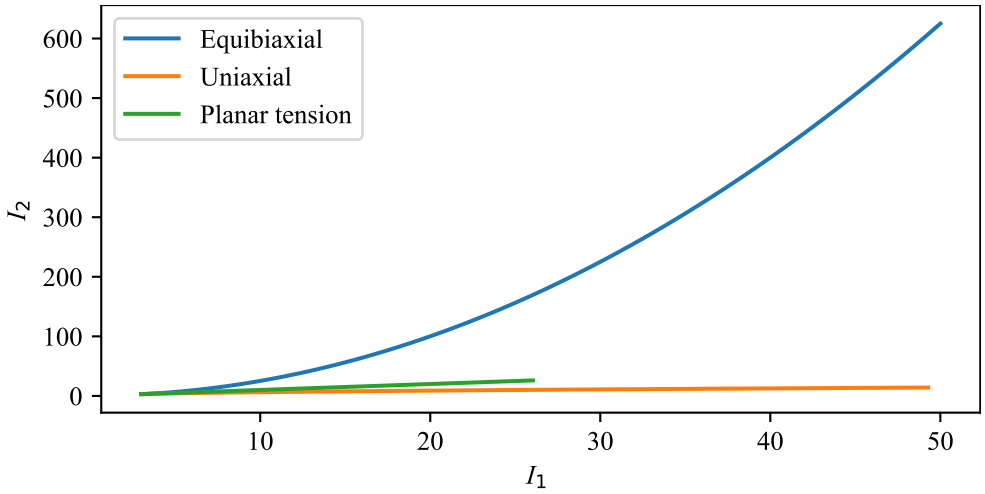


Figure 4.6: Diagrams showing the respective curves from which the proposed NN model was trained on. The number of samples can vary and is different for different material behaviours.

polyconvex since the coefficient c_{01} is negative. The Ogden material is reused from Sec. 3.3 with the strain-energy defined by Eq. $(3.10)_1$, i.e. the strain energy is of the form

$$\psi_{\text{Ogden}} = \sum_{p=1}^{3} \frac{\mu_p}{\alpha_p}(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_3^{\alpha_p} - 3), \tag{4.23}$$

with the material constants given in Table 1 and the temperature dependence is of the shear moduli is removed. Note that this model is polyconvex, this is shown explicitly in [4] for the same material parameters that were used in this work. The results for the simple tests in Figs. 4.7, 4.8 and 4.9 are presented with the 1st Piola-Kirchhoff stress as done in [51, 65].

4.1 Simple Tests

In Fig. 4.7 the results for the uniaxial (UT), equibiaxial (ET) and planar tension (PT) tests for the Neo-Hookean model are shown. The results show good agreement with a relative error of 3.15 %, 1.48 % and 0.05 %, respectively.

In Fig. 4.8 results UT, ET and PT for the Mooney-Rivlin model are shown. The results show good agreement with an error of 3.38%, 6.62% and 4.38%, respectively. The base model from which the data was generated is not polyconvex, nevertheless with the proposed NN a polyconvex model is obtained that correctly captures the desired behaviour.

In Fig. 4.9 results for UT, ET and PT are shown when the model is trained on data generated by the Ogden model. The results show poor agreement with an error of 32.01%, 25.77% and 20.38%, respectively. In the work [44] it has been noted that when polyconvexity is included in the NN *a priori* by limiting certain weights to be only positive the accuracy worsens. In that same work results are presented when considering the Neo-Hookean model, which is captured quite well by the PANN proposed in this work as seen in Fig. 4.7. In the work of [64] the used

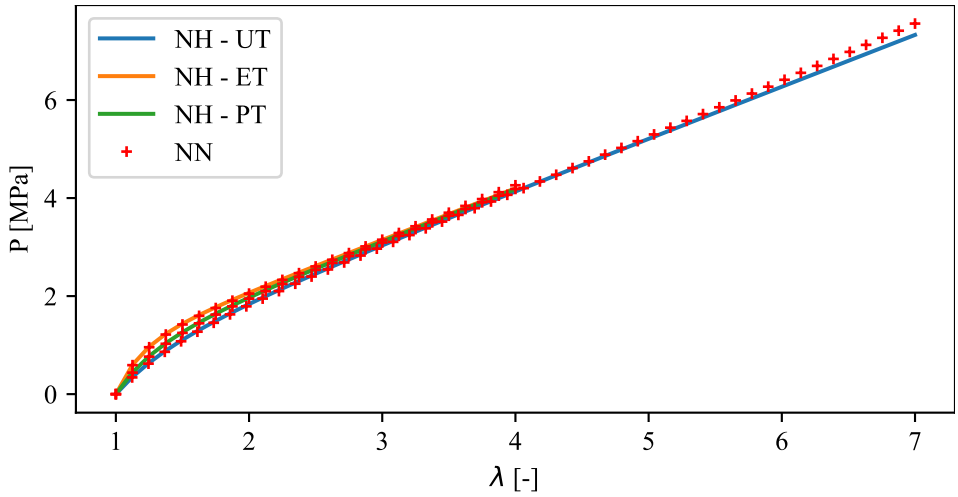


Figure 4.7: Results for the polyconvex model trained on data generated by the neo-Hookean model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

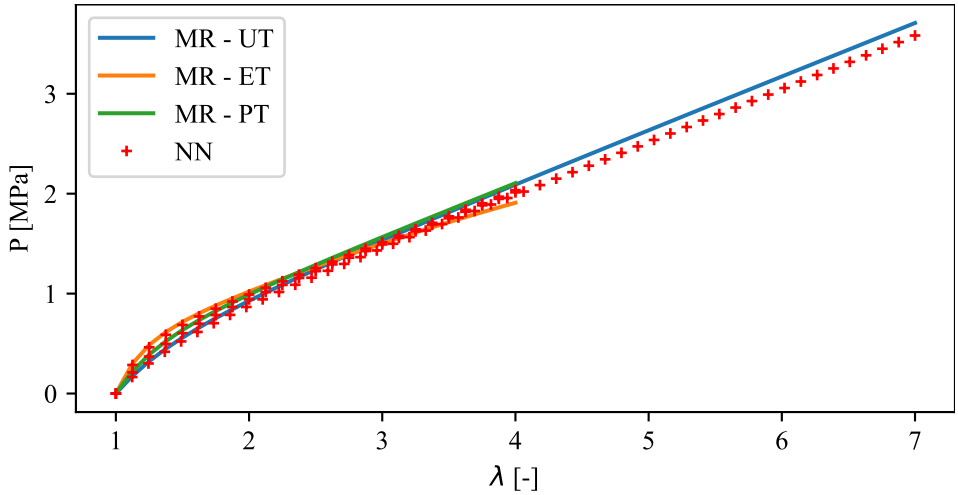


Figure 4.8: Results for the polyconvex model trained on data generated by the Mooney-Rivlin model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

Ogden model has only a single term (i.e. in Eq. (4.23) there is no summation but rather the summation index $p = 1$, and $\mu_p = \alpha_p = 1$) so a much simpler model then the one used in this work.

Following the results when using a polyconvex NN, the results are presented for the NN models where the polyconvexity constraint is not enforced, i.e. the weights are not restricted to be positive. When giving the NN more freedom it captures the desired material behaviours excellently regardless of the underlying material behaviour.

In Fig. 4.10 results UT, ET and PT are shown with the Neo-Hookean model used to generate the training data and a non-polyconvex NN model is used. The results show good agreement with an error of 1.03%, 0.07% and 0.21%, respectively.

In Fig. 4.11 results UT, ET and PT are shown with the Mooney-Rivlin model used to generate the training data and a non-polyconvex NN model is used. The results show excellent agreement

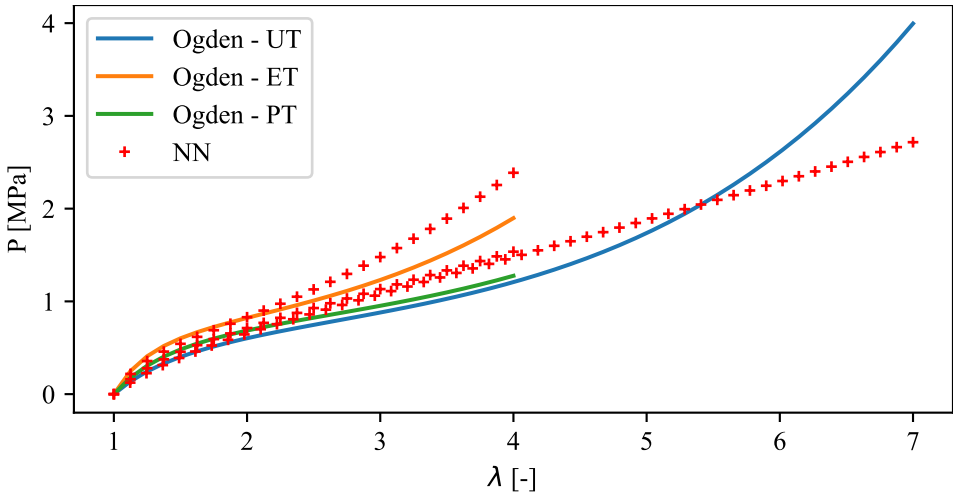


Figure 4.9: Results for the polyconvex model trained on data generated by the Ogden model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

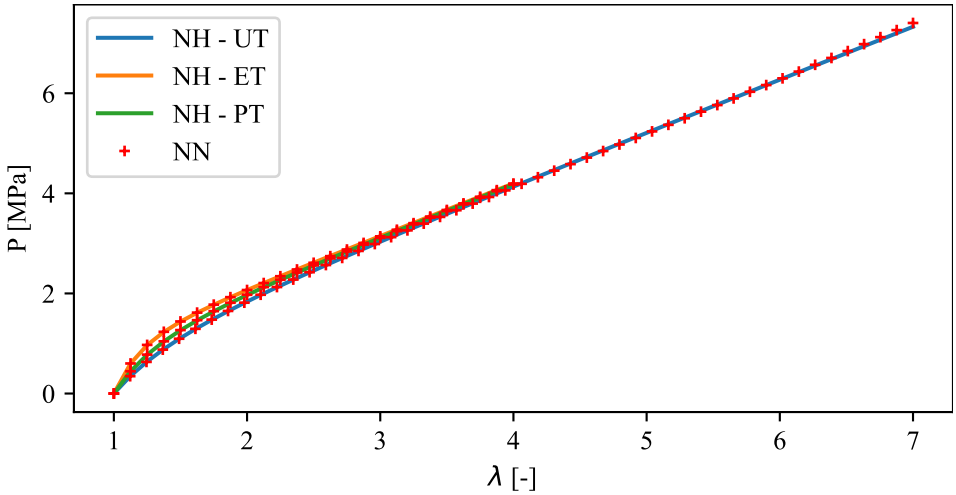


Figure 4.10: Results for the non-polyconvex model trained on data generated by the Neo-Hookean model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

with an error of 0.37%, 0.17% and 0.01%, respectively.

In Fig. 4.12 results UT, ET and PT are shown with the Ogden model used to generate the training data and a non-polyconvex NN model is used. The results show excellent agreement with an error of 1.04%, 0.06% and 0.54%, respectively.

Relaxing the polyconvexity constraint has shown to be beneficial in terms of capturing the stress curves. However, relaxing these constraint also allows the possibility that the activation functions take on negative values thus the *non-negativity of energy* is no longer strictly enforced. To somewhat alleviate this problem the NNs can be numerically tested on a domain, such as the one shown in Fig. 3.12, to see if it predicts negative values for the energy on an are of interest. When this simple test was done none of the non-polyconvex NNs predicted negative values for the strain energy. Although not strictly enforced, this result is attributed to the fulfilment of *thermodynamic consistency*, i.e. training the NNs on stresses which are the 1st derivatives thus

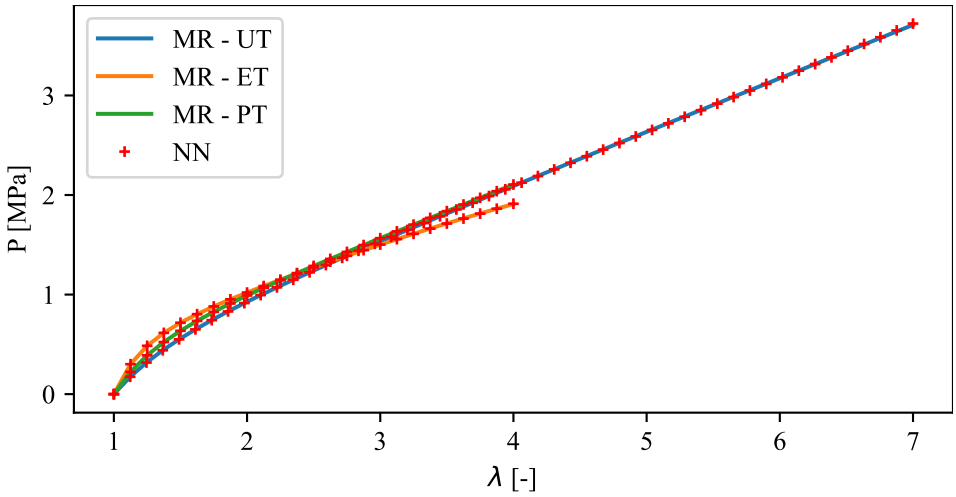


Figure 4.11: Results for the nonpolyconvex model trained on data generated by the Mooney-Rivlin model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

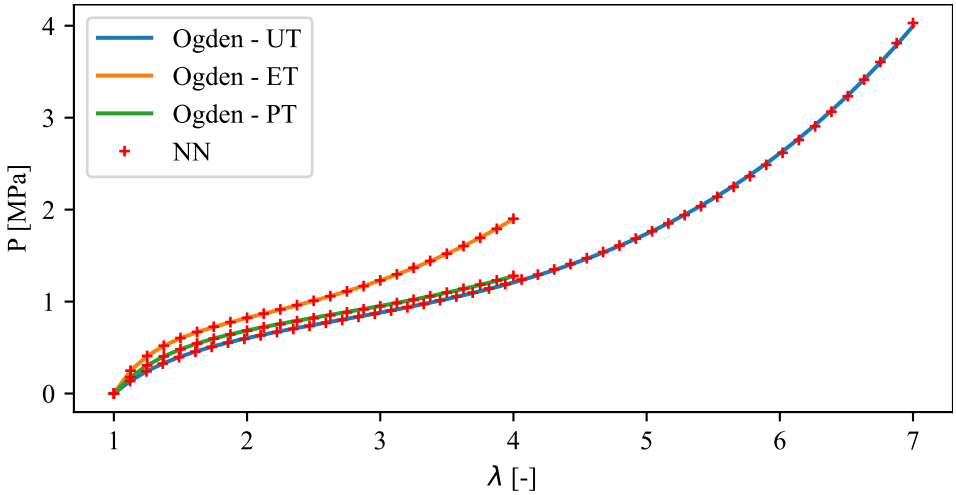
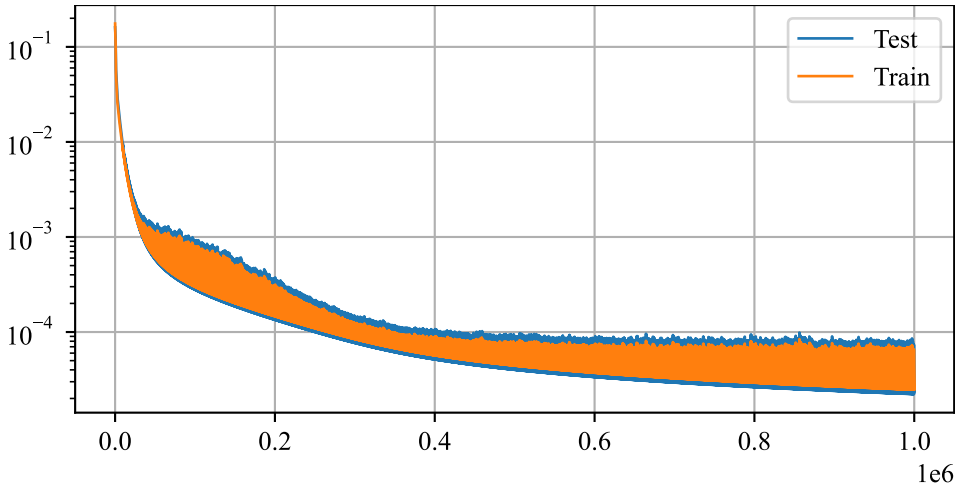


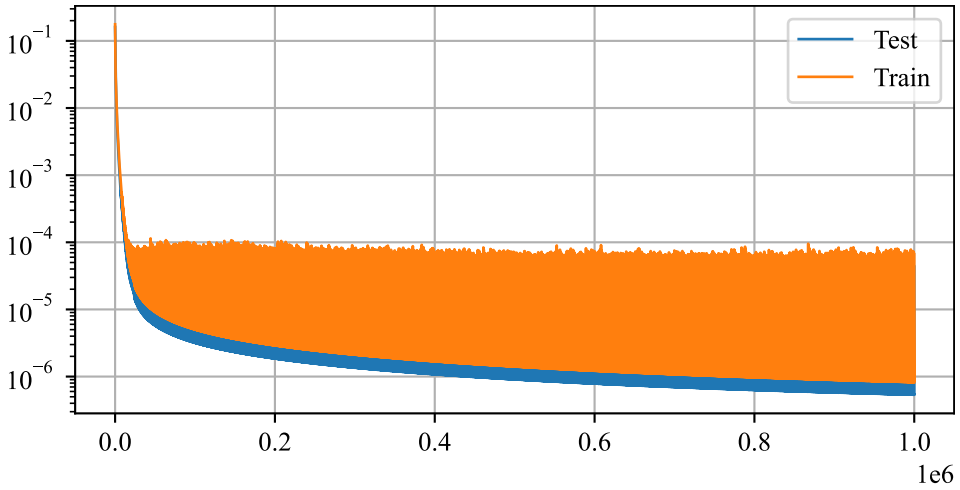
Figure 4.12: Results for the nonpolyconvex model trained on data generated by the Ogden model. Results include the uniaxial (UT), equibiaxial (ET) and planar (PT) tension tests.

weakly enforcing the NN model to grow around zero energy.

The polyconvexity condition also impacts the losses during training and the training time of the NNs. In Fig. 4.13 both the polyconvex and non-polyconvex models trained until the maximum pre-set limit of 1 000 000 epochs. However, there are two clear differences between the models, the non-polyconvex model achieved losses two orders of magnitude lower than the polyconvex model, and the error between epochs had a greater variance.



(a)



(b)

Figure 4.13: Graphs showing the loss function over a certain number of epochs until the training terminated for the NNs trained on data generated by the Neo-Hookean model. Note that the models were trained until the limit of 1 000 000 epochs and the loss is still decreasing. The top figure presents the train/test losses for the polyconvex NN model, and the bottom figure for the non-polyconvex NN model.

This training behaviour on the simpler Neo-Hookean model from Fig. 4.13 is contrasted by the training behaviour from the Mooney-Rivlin model in Fig. 4.14 where the training error and duration were comparable for the polyconvex model, but were much lower for the non-polyconvex model in Fig. 4.14b where the error is 3 orders of magnitude lower than for the polyconvex model and the training time is shortened by half, the model terminated once it reached the patience of 20 000 epochs. When the more complex Ogden model is used as the baseline then this is even more exaggerated, the differences between the training and validation errors is about 3 orders of magnitude and the training time is cut by more than half. In Fig. 4.15 the training took more about 10 times less, but this can vary between individual trainings.

Another quantity of interest would be the strain energy itself. Since the NN is trained on the

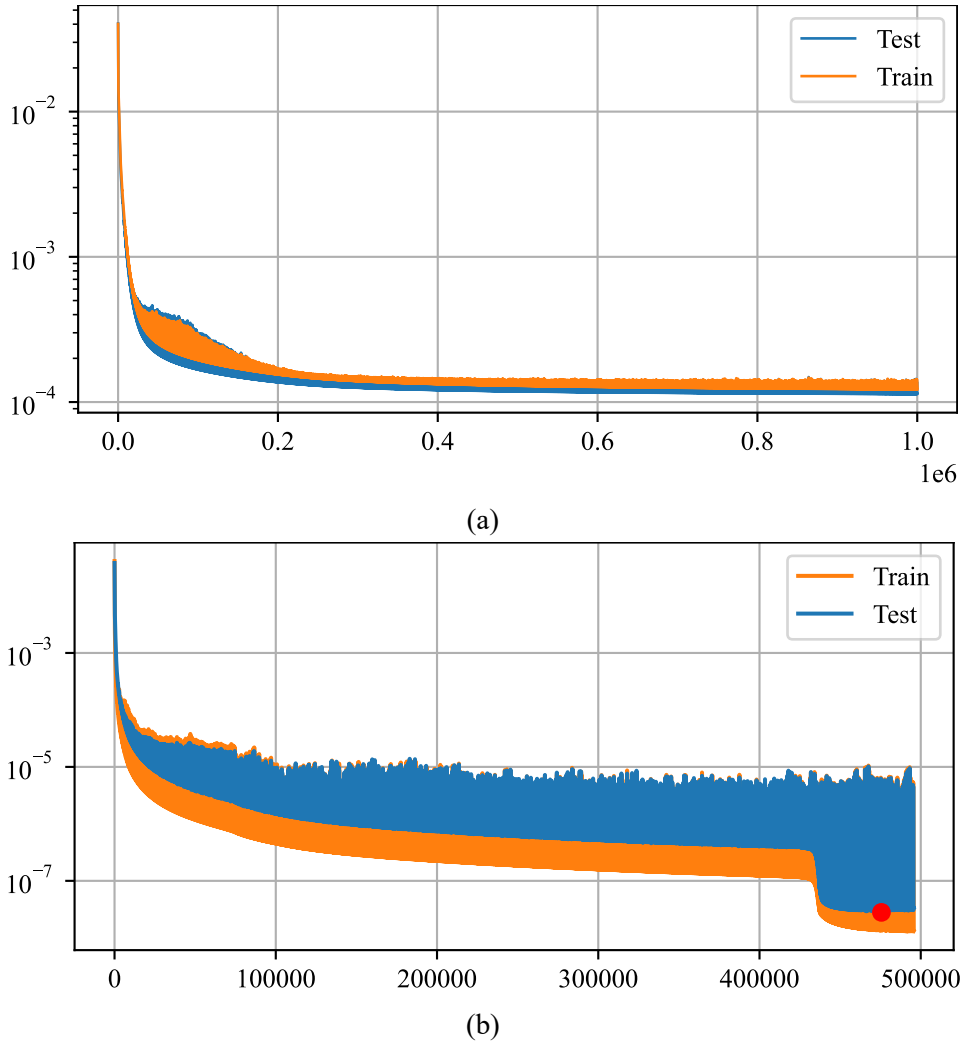


Figure 4.14: Graphs showing the loss function over a certain number of epochs until the training terminated for the NNs trained on data generated by the Mooney-Rivlin model. Note that the patience before termination was set to 20 000 epochs. The top figure presents the train/test losses for the polyconvex NN model, and the bottom figure for the non-polyconvex NN model. The red dot in the bottom figure represents the best validation loss obtained during training.

stresses, i.e. 1st derivatives of the strain energy, and is normalized at 0 for an undeformed state, it is expected to have correctly captured the evolution of the strain energy. Thus, it is expected that the energy predictions are as accurate as the stress predictions. Taking the simplest Neo-Hookean model would imply that the strain energy is correctly captured for both the polyconvex and non-polyconvex models, whereas the predictions of the NNs trained on Ogdens model are expected to be worse for the polyconvex model. The results are shown on Fig. 4.16 for the models trained on data generated by the Neo-Hookean model, and in Fig. 4.17 for the models trained on data generated by the Ogden model. The energy predictions behave as expected, with no noticeable difference for the Neo-Hookean NN models, but large differences for the Ogden NN models.

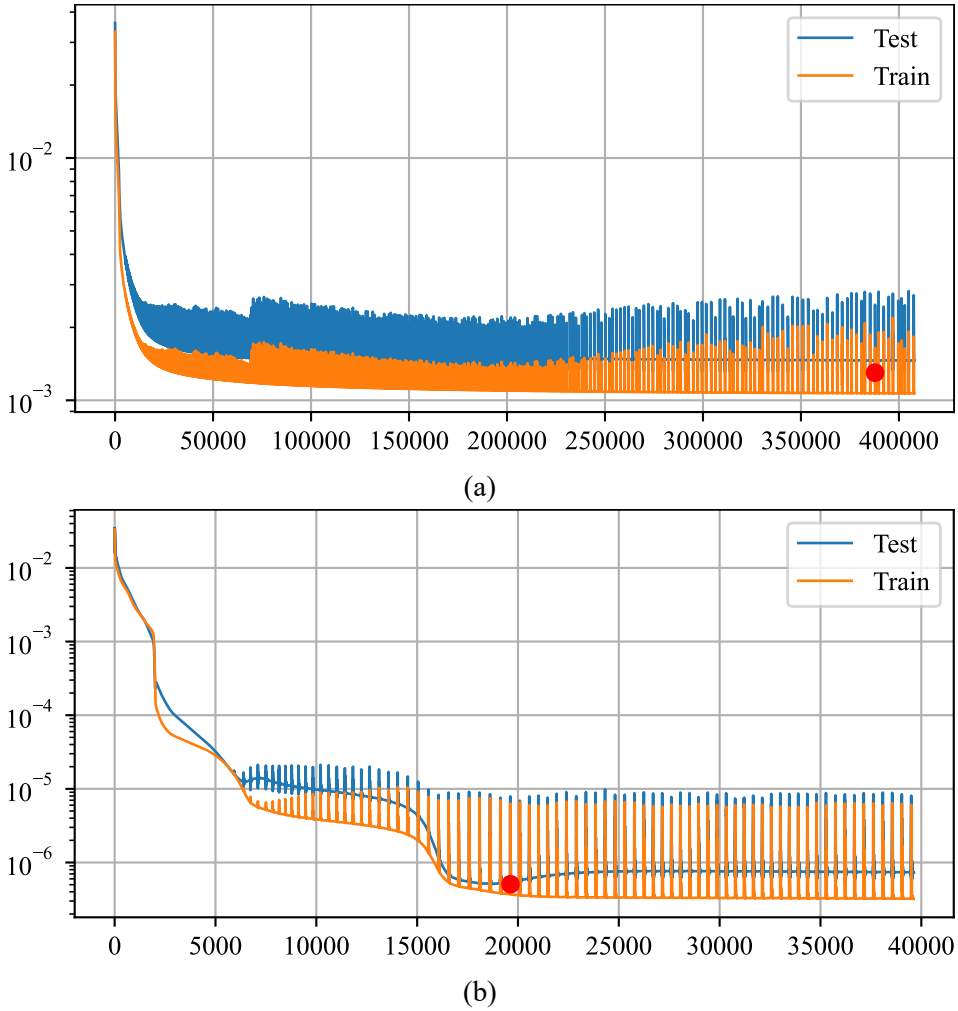
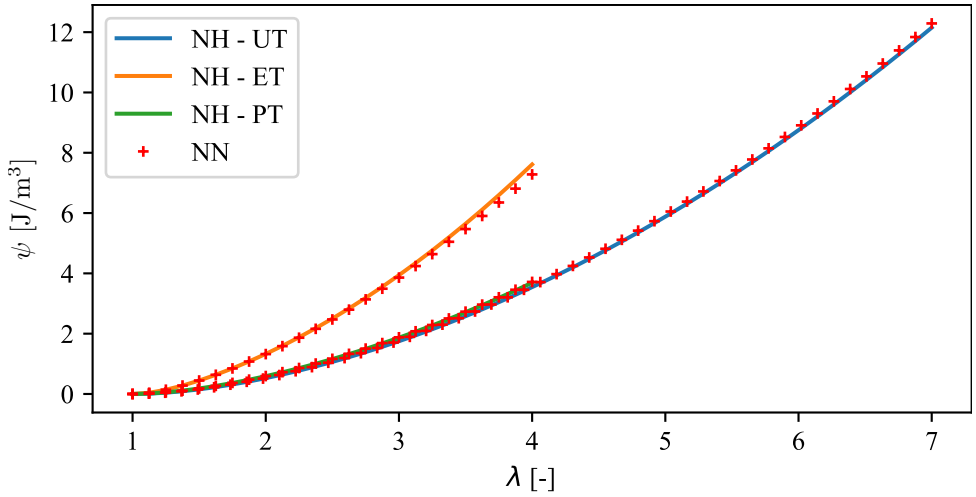
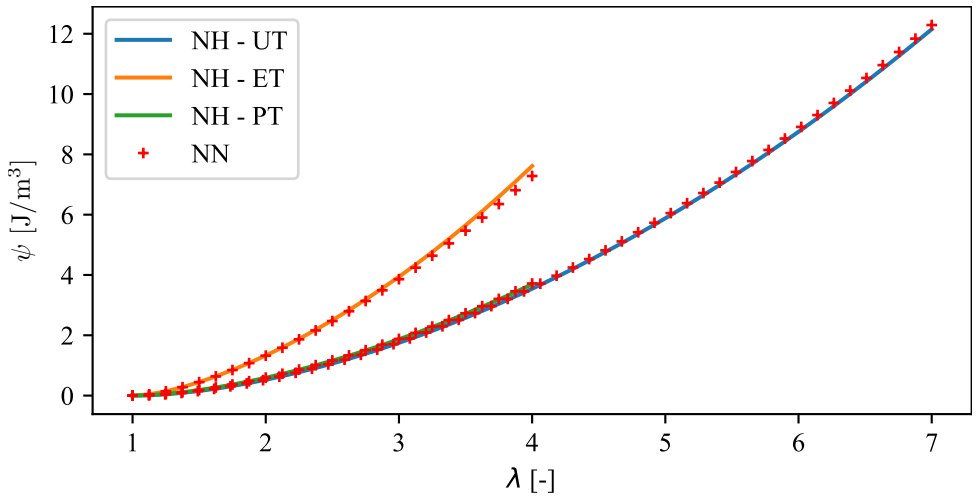


Figure 4.15: Graphs showing the loss function over a certain number of epochs until the training terminated for the NNs trained on data generated by the Ogden model. Note that the patience before termination was set to 20 000 epochs. The top figure presents the train/test losses for the polyconvex NN model, and the bottom figure for the non-polyconvex NN model. The red dot represents the best validation loss obtained during training of the respective models.

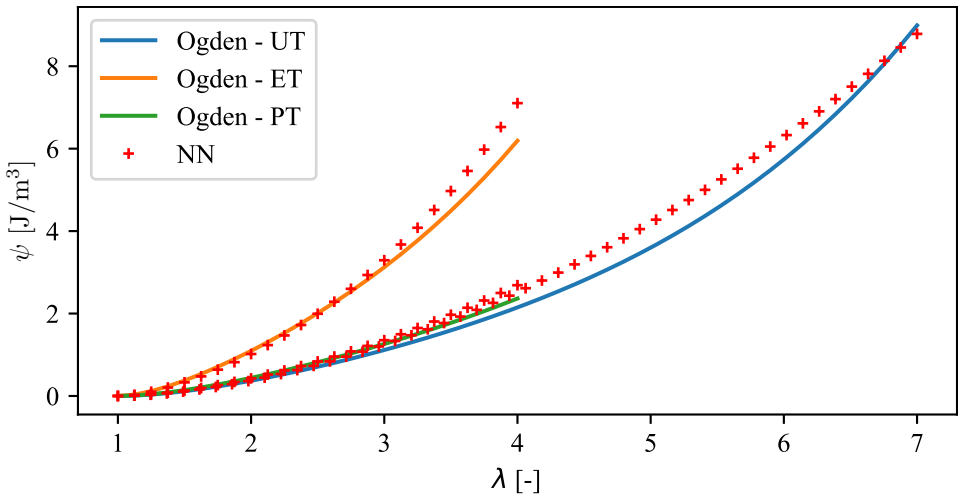


(a)

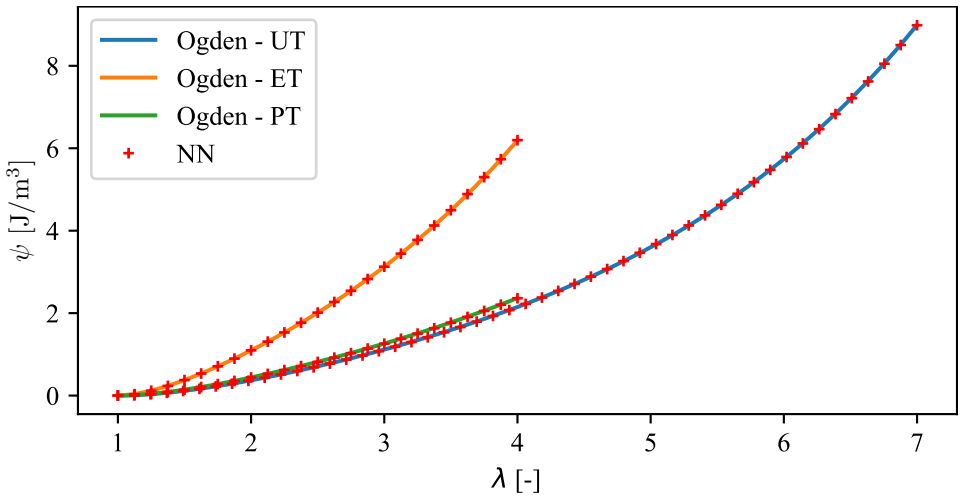


(b)

Figure 4.16: Strain energy predictions by the polyconvex (top) and non-polyconvex (bottom) NN models trained on data generated by the Neo-Hookean model.



(a)

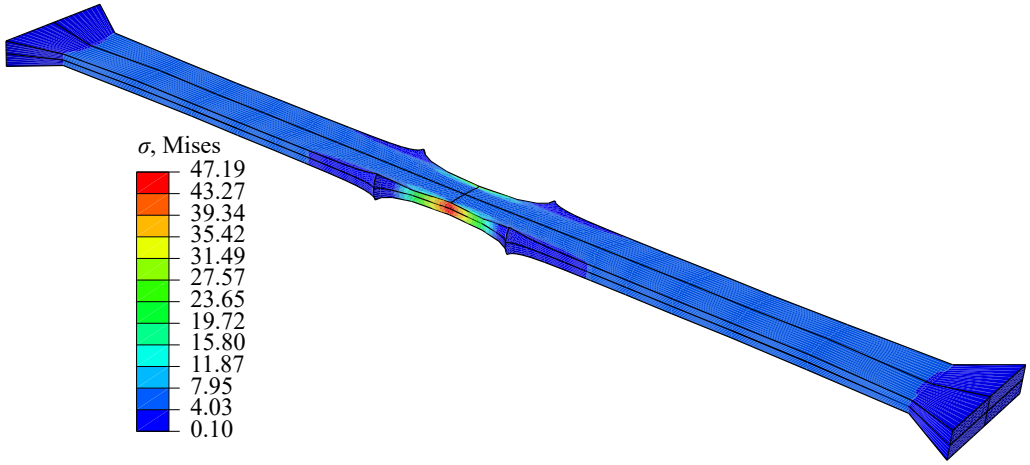


(b)

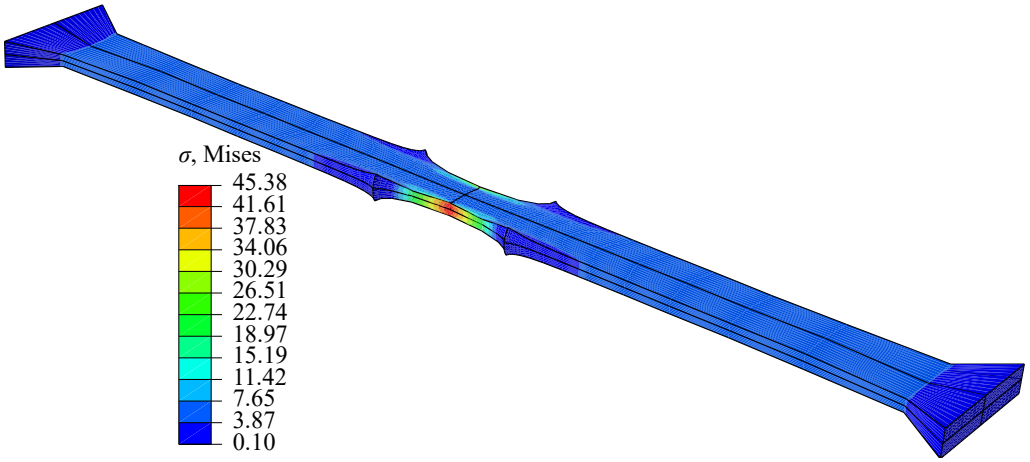
Figure 4.17: Strain energy predictions by the polyconvex (top) and non-polyconvex (bottom) NN models trained on data generated by the Ogden model.

4.2 Cracked Bar Numerical Example

The amount of data needed to train these NNs is also be significantly reduced compared to the first results presented in Sec. 3.6.3. Taking the cracked bar example from the same section, without the thermoelastic effect, and training the non-polyconvex NN on 15 examples in total (5 per uniaxial, equibiaxial and planar tension cases) the obtained results show good agreement with the base Ogden model, as shown in Fig. 4.18. The highest stress values differ by 3.84% whereas the minimum values are the same.



(a)



(b)

Figure 4.18: Solutions for the cracked bar example first shown in Sec. 3.3, without the thermoelastic effect. The top figure shows the stress plot with the NN model trained on 15 samples and the bottom figure shows the reference Ogden solution.

4.3 Torsion of a Cuboid Numerical Example

Another example with complex loading conditions is given in Fig. 4.19 where a brick (cuboid) is at the same time under a tensile and torsional load. Hybrid finite elements were used (type

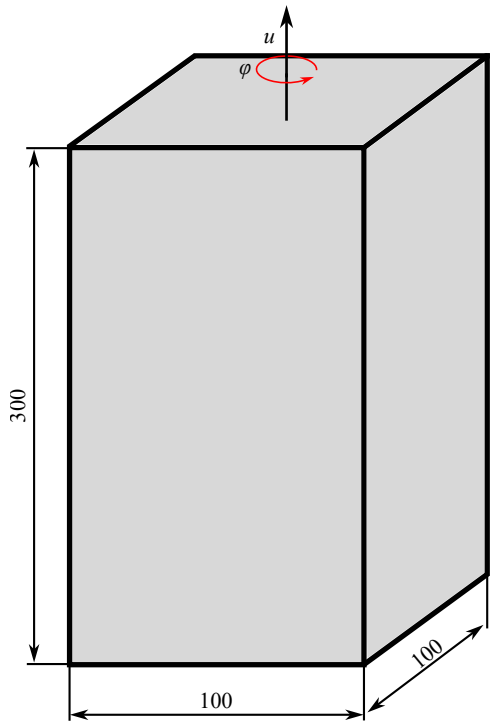
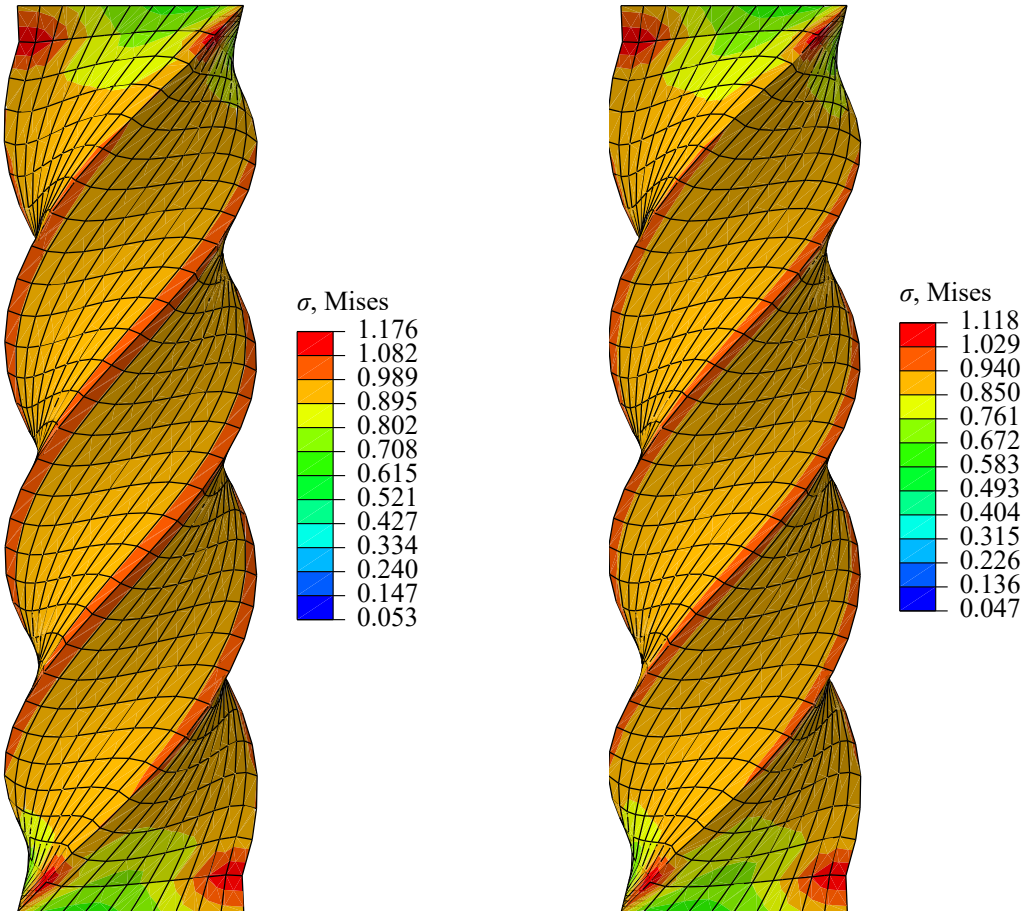


Figure 4.19: Geometry and loading conditions of the brick under torsion, dimensions given in millimetres. The displacement u is equal to 100 mm and the angle φ is equal to 2π, i.e. a full circle.

C3D8H in Abaqus), same as for the cracked bar example. The load is prescribed at the top surface via a reference point so the top surface is rigid, and the bottom surface is fixed. The final results with the stress plots are shown in Fig. 4.20 with the error of the maximum stress values of 5.2%.



(a) (b)

Figure 4.20: Plots of von Mises stress for the torsion brick example. The left figure shows the Ogden solution and the right figure shows the solution using a non-polyconvex NN.

5 Extension to the Mullins effect

The results presented until now support the LINEXP-PANN as a suitable candidate for the general modelling of hyperelastic behaviour for various behaviours. However, until now only basic hyperelastic behaviour has been covered. In this section the novel NN architecture is used to model the Mullins effect, a simple isotropic damage model for rubbers. Furthermore, a novel training method is presented which was first proposed in [74].

The Mullins effect is a strain-induced softening effect first described in [49]. It is characterized by two responses, a primary response during the loading cycle, and a secondary response during unloading and reloading. The effect is shown in Fig. 5.1 with two unloading phases.

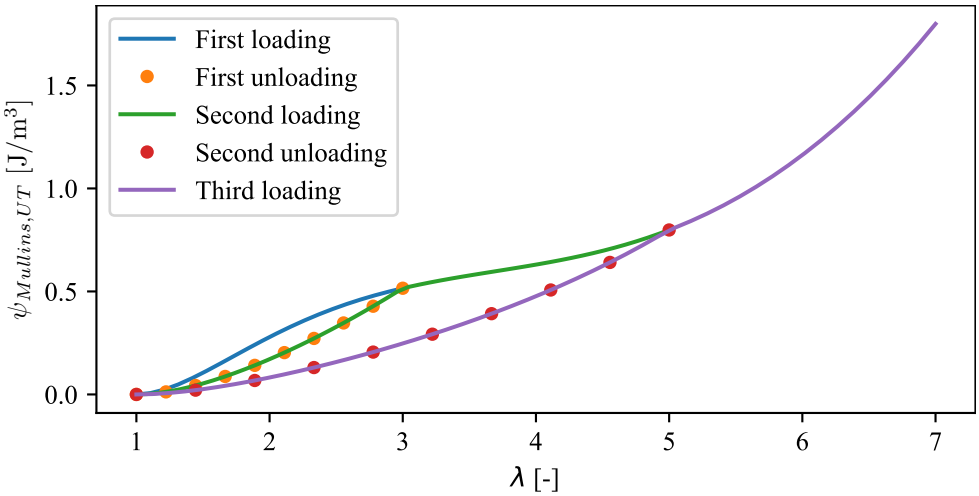


Figure 5.1: Illustration of the Mullins effect.

This effect is described by the introduction of damage in the simple form of

$$\psi_{\text{Mullins}} = (1 - \zeta(\gamma))\,\psi_0, \tag{5.1}$$

where ψ_0 is the base underlying energy described by the Neo-Hookean, Mooney-Rivlin, Ogden, or any other model, ζ is the damage parameter and γ is the historical variable used to calculate the damage parameter. The Mullins effect can be described with many expressions for the damage parameter (for a comprehensive overview the reader is referred to [16]), but in this work the one shown in [27] is taken. It has the form

$$\zeta(\gamma) = \zeta_\infty \left(1 - \exp\left(-\frac{\gamma}{\iota}\right)\right), \tag{5.2}$$

where ζ_∞ is the maximum attainable value of the damage parameter and ι is referred to as the saturation parameter. These are material constants that define the Mullins effect. The historical variable γ can be expressed as

$$\gamma = \max_{s\in[0,t]} \psi_0(s), \tag{5.3}$$

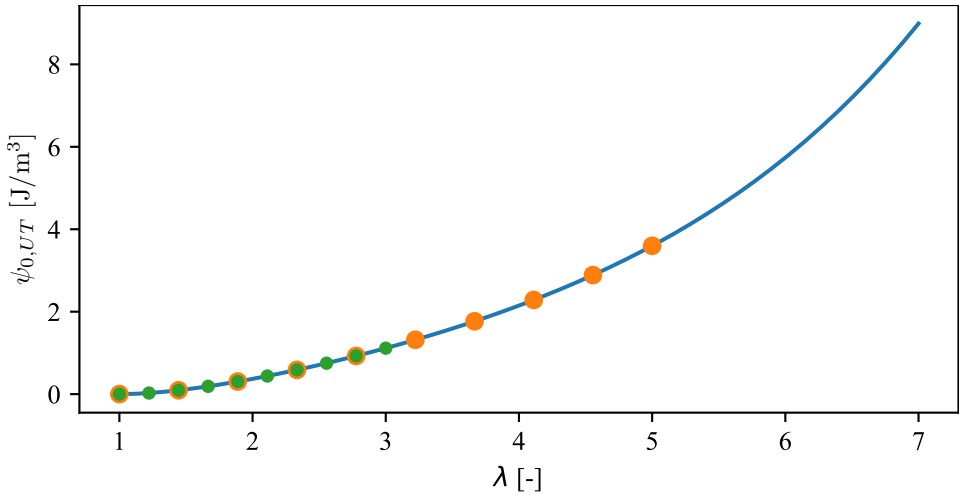


Figure 5.2: Evolution of the underlying undamaged strain energy ψ_0 from the deformation process shown in Fig. 5.1.

that is as the maximum value of the underlying undamaged strain energy attained during the entire observed deformation process until the current time t. In Fig. 5.1 the value of γ rises during the first loading until $\lambda = 3$, and during unloading it does not change since the values of the undamaged energy are lower. During the second loading the value of γ does not change until the point where the material was first unloaded is reached. The evolution of the underlying undamaged energy is shown in Fig. 5.2. The ordinary Ogden model with the material properties given in Table 1 is used with the parameters $\zeta_\infty = 0.8$ and $\iota = 1$.

The first Piola-Kirchhoff stress is defined as in Eq. (2.75) so when considering the Mullins effect it simply follows as

$$\mathbf{P}_{\text{Mullins}} = \frac{\partial \psi_{\text{Mullins}}}{\partial \mathbf{F}} = (1 - \zeta(\gamma)) \frac{\partial \psi_0}{\partial \mathbf{F}} = (1 - \zeta(\gamma)) \mathbf{P}_0, \tag{5.4}$$

where $\mathbf{P}_0$ is the stress of the undamaged material behaviour that is used in the previous sections. The evolution of Cauchy stress with the Mullins effect included is shown in Fig. 5.3a, while the undamaged Cuahcy stress evolution is shown in Fig. 5.3b. It should be noted that in case of incompressible behaviour it is defined as in Eq. (2.84)

$$\mathbf{P} = (1 - \zeta(\gamma)) \frac{\partial \psi_0(\mathbf{F})}{\partial \mathbf{F}} - Jp\mathbf{F}^{-\mathrm{T}} = (1 - \zeta(\gamma)) \mathbf{P}_0 - Jp\mathbf{F}^{-\mathrm{T}}, \tag{5.5}$$

where the damage does not impact the volumetric part of the stress tensor with the Lagrange multiplier p, but only the part that follows from the strain energy.

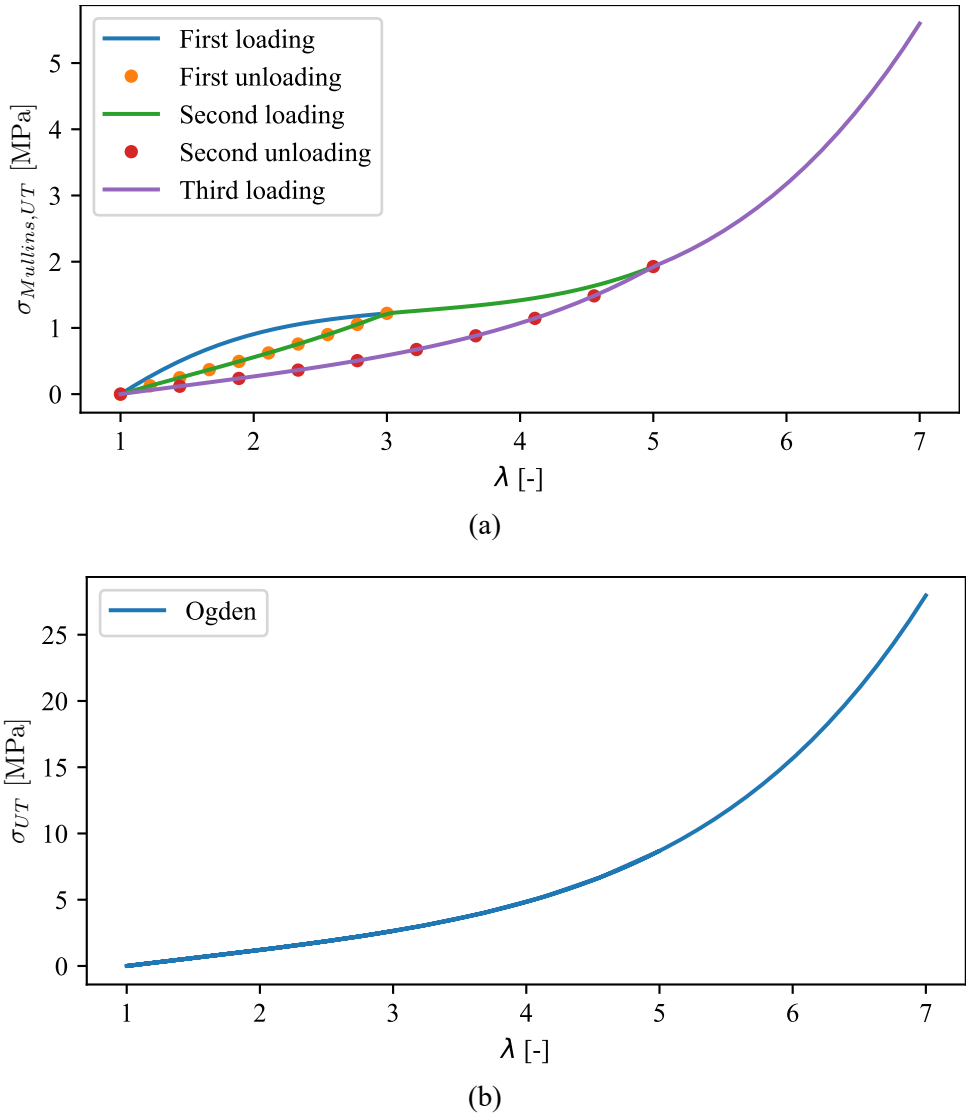


Figure 5.3: Cauchy stress evolution with the Mullins effect included is shown in the top figure, the results follow the loading in Fig. 5.1. The undamaged Cauchy stress evolution is shown in the bottom figure.

The Neural Network Architecture of the LINEXP-PANN can be reused and adapted for the Mullins effect. Drawing inspiration from the Constitutive Artificial Neural Network (CANN) presented in [46], the LINEXP-PANN can be used as a base energy prediction NN while the rest of the NN architecture can be directly taken from existing models and adapted in the NN framework. This means that the value of ψ_0 in Eq. (5.1) can be replaced by the LINEXP-PANN while retaining all other elements. The manner in which this is modified to include the existing description of the Mullins effect is illustrated in Fig. 5.4. Note that the NN block shown in the figure refers to the LINEXP-PANN presented in Fig. 4.5 and only one such block exists but is shared for the calculation of both ψ_0 and $\psi_{0,\text{max}}$, where $\psi_{0,\text{max}}$ is the variable used in the NN to note the value γ introduced in Eq. (5.1) and explicitly defined in Eq. (5.3). A more detailed illustration containing the names of the weights and better illustrating the fact that only one LINEXP-PANN is used is shown in Fig. 5.5. Additionally, the values ζ_∞ and ι used for modelling the Mullins

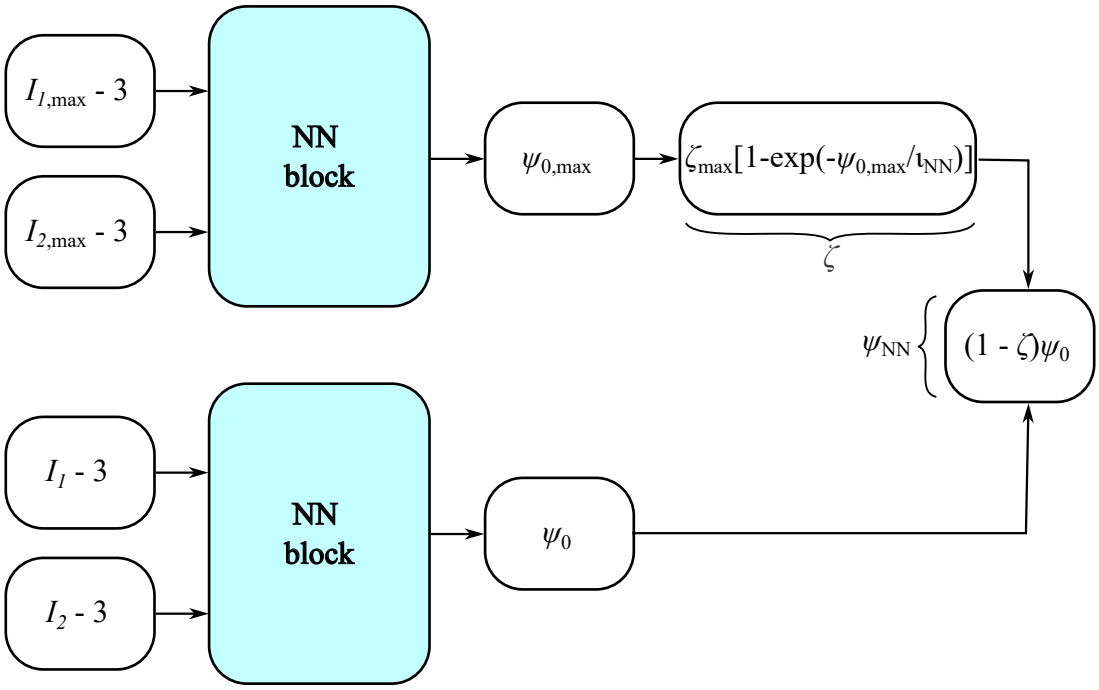


Figure 5.4: Illustration of the NN architecture for the Mullins effect. Only one NN block exists and its weights are used for calculating both $\psi_{0,\max}$ and ψ_0.

effect are included in the NN as $\zeta_{\max}$ and ι_{NN} and are trainable parameters.

Given that the parameter γ is the largest historical value of the undamaged strain energy (cf. Eq. (5.3)) then it can be effectively recalculated using the invariants at the time it occurred, hence the use of $I_{1,\max}$ and $I_{2,\max}$ as the input variables that serve as historical variables. In practice, when conducting an experiment, it is more straightforward to calculate the stress rather than the strain energy. Also, when simple tests are used such as uniaxial, equibiaxial and planar tension, then it is easy to tell when the maximum was reached, which is when the largest stretch is applied. Thus, all the data needed for training the NN is gathered in a more accurate manner since the invariants can be calculated from the prescribed stretch using Eq. (2.15), and the stresses can be defined with respect to the reference configuration, i.e. using the stress tensor **P**.

Following the detailed description in Fig. 5.5 the expression for the undamaged energy ψ_0 is

$$\psi_0 = \sum_{i=1}^{n} w_{i,1}^{[2]} h_i(I_1 - 3, I_2 - 3), \tag{5.6}$$

for the maximum value of the undamaged energy $\psi_{0,\max}$ is

$$\psi_{0,\max} = \sum_{i=1}^{n} w_{i,1}^{[2]} h_i(I_{1,\max} - 3, I_{2,\max} - 3), \tag{5.7}$$

with the activation function $h_i(I_1 - 3, I_2 - 3)$ being the same as shown in Eq. (4.17). Also, in Fig. 5.5 the weights are shown as arguments for each activation function to emphasize the fact that they are shared for calculating the values ψ_0 and $\psi_{0,\max}$. The expression for the damage

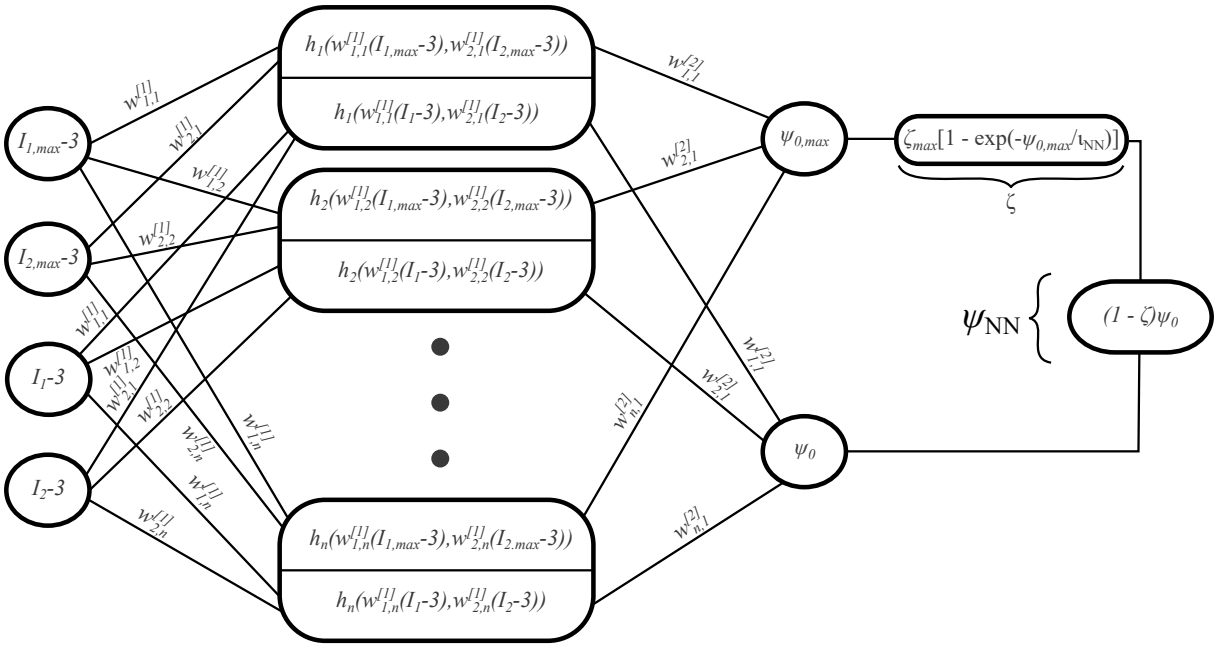


Figure 5.5: Detailed illustration of the NN architecture for the Mullins effect.

variable ζ is

$$\zeta = \zeta_{\text{max}} \left[1 - \exp\left(-\frac{\psi_{0,\text{max}}}{\iota_{\text{NN}}} \right) \right]. \tag{5.8}$$

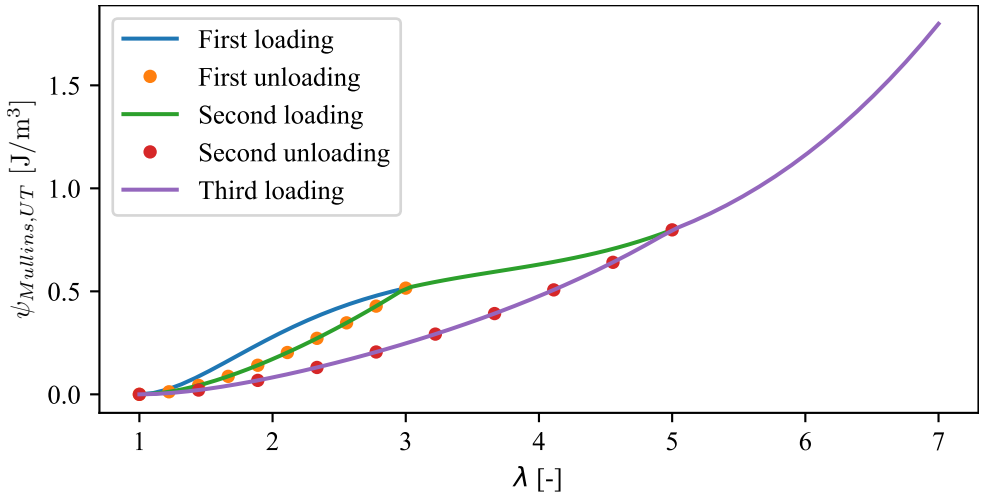
The final value for the energy ψ_{NN} is

$$\psi_{\text{NN}} = \left[1 - \zeta_{\text{max}} + \zeta_{\text{max}} \exp\left(-\frac{\sum_{i=1}^{n} w_{i,1}^{[2]} h_i(I_{1,\text{max}} - 3, I_{2,\text{max}} - 3)}{\iota_{\text{NN}}} \right) \right] \sum_{i=1}^{n} w_{i,1}^{[2]} g_i(I_1 - 3, I_2 - 3). \tag{5.9}$$

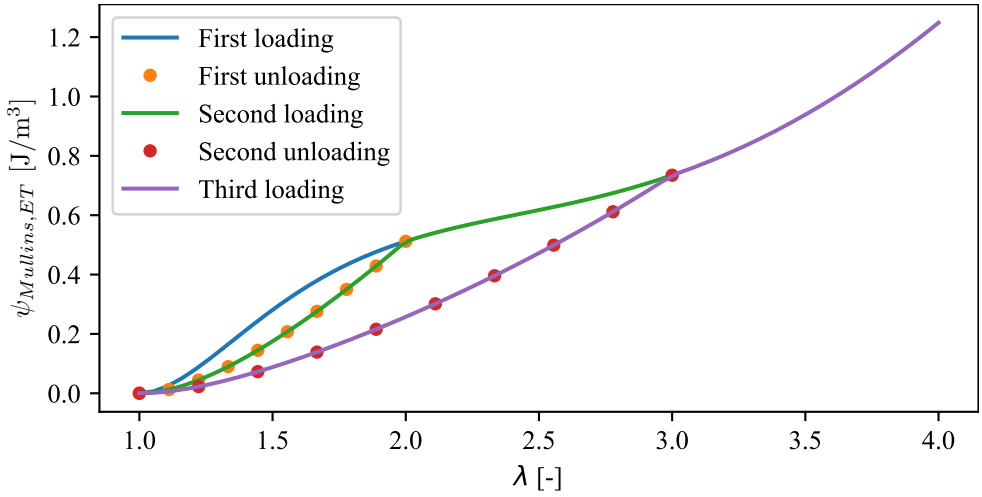
Given that the LINEXP-PANN is used as the basic building block all the physical conditions mentioned in Sec. 4 can be fulfilled in the same manner. The addition of the Mullins effect related variables such as $\psi_{0,\text{max}}$ and ζ does not affect any of the previously mentioned and incorporated conditions.

The training data is gathered from the artificial uniaxial, equibiaxial and planar tension tests, all plane stress cases. The energy responses can be viewed in Fig. 5.6 and the stress responses (that were used to train the NN) are shown in Fig. 5.7. The following loadings were performed for the training data:

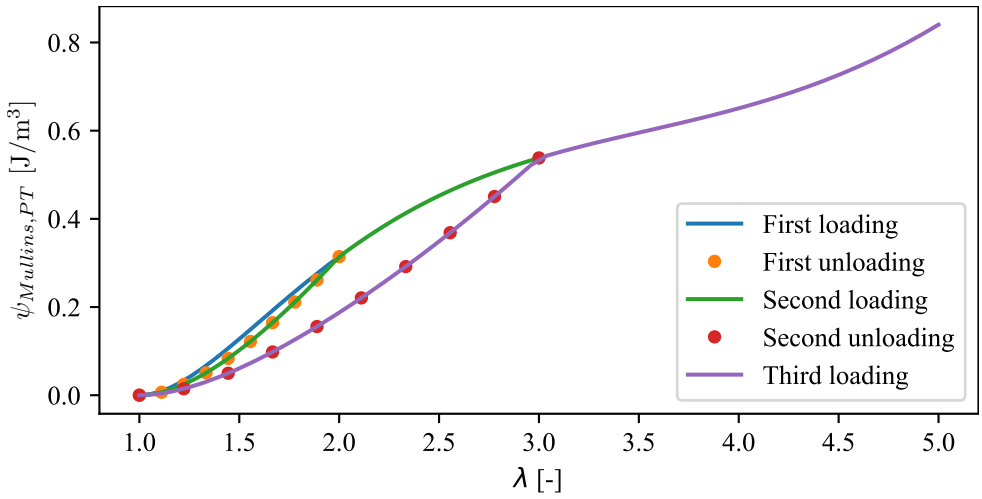
- uniaxial λ: $1 \xrightarrow{load} 3 \xrightarrow{unload} 1 \xrightarrow{load} 5 \xrightarrow{unload} 1 \xrightarrow{load} 7$
- equibiaxial λ: $1 \xrightarrow{load} 2 \xrightarrow{unload} 1 \xrightarrow{load} 3 \xrightarrow{unload} 1 \xrightarrow{load} 4$
- planar tension λ: $1 \xrightarrow{load} 2 \xrightarrow{unload} 1 \xrightarrow{load} 3 \xrightarrow{unload} 1 \xrightarrow{load} 5$



(a) Uniaxial tension

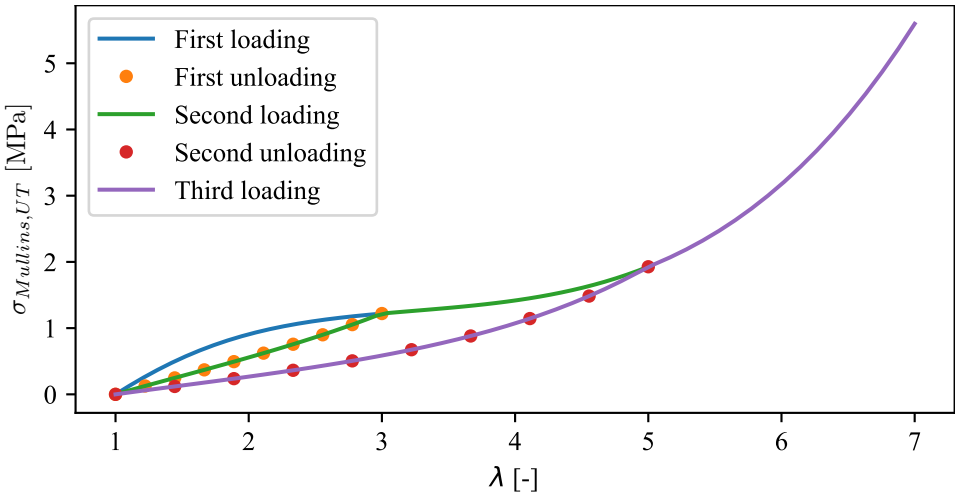


(b) Equibiaxial tension.

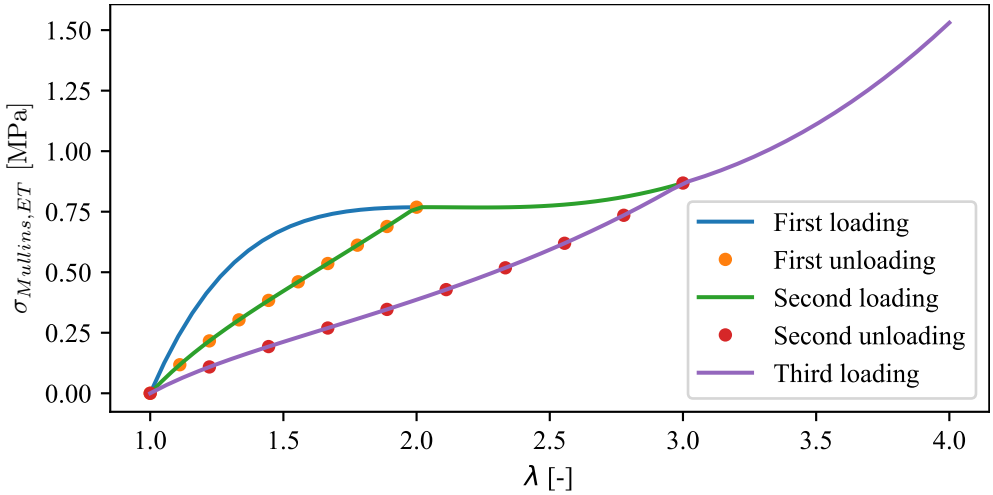


(c) Planar tension.

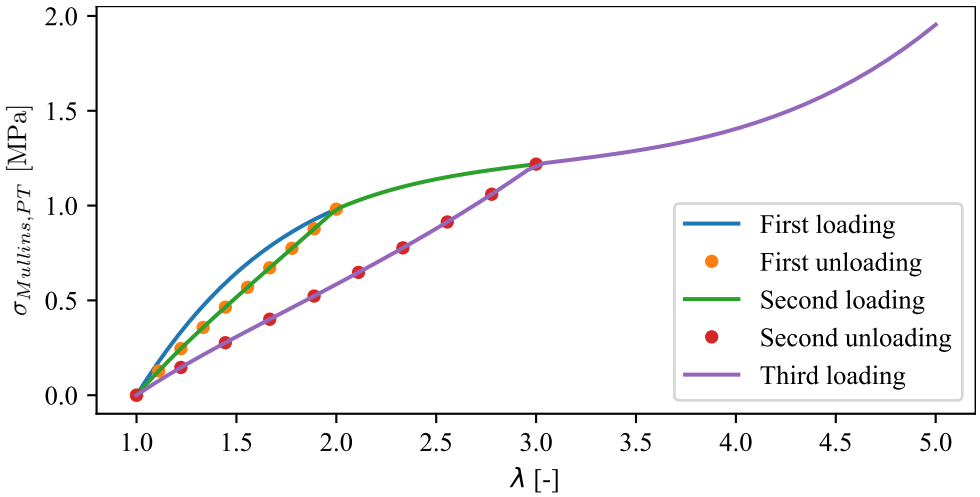
Figure 5.6: Energy responses of the of the training samples using the Ogden model with Mullins effect. These are shown for completeness and are not used during training.



(a) Uniaxial tension.



(b) Equibiaxial tension.



(c) Planar tension.

Figure 5.7: Cauchy stress responses of the of the training samples using the Ogden model with Mullins effect. These data are used during training.

Several other NN architectures or combinations of the NN parameters are introduced to compare other possible options of modelling the undamaged strain energy and to compare against the LINEXP-PANN. They are organised into 6 different cases as follows:

1. *polyconvex* LINEXP-PANN with the weights $w_{i,j}^{[l]}$ and α_i constrained to be non-negative, this NN fulfils all the conditions mentioned in Sec. 4.

2. LINEXP-PANN with only the weights $w_{i,j}^{[2]}$ constrained to be non-negative, this preserves convexity in the invariants rendering it an input convex neural network (ICNN), but does not *a priori* preserve *polyconvexity* or *non-negativity of the strain energy*.

3. LINEXP-PANN where the weights $w_{i,j}^{[1]}$ are constrained to be non-negative but $w_{i,j}^{[2]}$ are not, this NN does not preserve any convexity in any argument nor does it preserve *non-negativity of the strain energy* but is simply shown for completeness so all the combinations of parameters are investigated.

4. LINEXP-PANN without any constraints on the weights $w_{i,j}^{[l]}$ or α_i, *polyconvexity* and *non-negativity of the strain energy* are not *a priori* fulfilled, but this unconstrained network has the most adaptability.

5. An NN where the invariants are not summed in the exponent but passed into the hidden layer separately, effectively having two layers in the NN architecture that are later concatenated, see Fig. 5.8. This is done to test an NN with the linear exponential activation function while keeping the strain energy a sum of individual subfunctions $\psi(I_1, I_2) = \psi(I_1) + \psi(I_2)$, similar to [46].

6. The isotropic perfectly incompressible CANN model from [46] is used in the NN block to verify against another NN model .

The diagrams showing the losses during training are shown in Fig. 5.9. In Fig 5.9a, case 1, the *polyconvex* LINEXP-PANN results are shown, training reached the limit of 10^6 epochs with the best loss of $5.44 \cdot 10^{-5}$. In Fig. 5.9b, case 2, the ICNN version of the LINEXP-PANN where *convexity* with respect to the invariants is guaranteed (the weights $w_{i,j}^{[2]}$ are constrained to be non-negative), but *non-negativity of the strain energy* is not, the best loss was slightly lower at $5.3 \cdot 10^{-5}$. For the sake of completeness the NN where the weights $w_{i,j}^{[1]}$ are constrained, but $w_{i,j}^{[2]}$ are not, is shown in Fig. 5.9c with a higher loss at $5.7 \cdot 10^{-5}$. This particular NN is not suitable for use as neither *polyconvexity* nor *non-negativity of the strain energy* are guaranteed while at the same time performing worse than other cases. In Fig. 5.9d the unconstrained LINEXP-PANN immediately shows the best results of all the cases with the lowest loss of $7.24 \cdot 10^{-7}$ thus compensating for the fact that *polyconvexity* and *non-negativity of the strain energy* are no longer guaranteed. This is in line with results from other authors that used PANNs, as noted in [44] where relaxing the *polyconvexity* requirement rendered an NN with greater accuracy. It

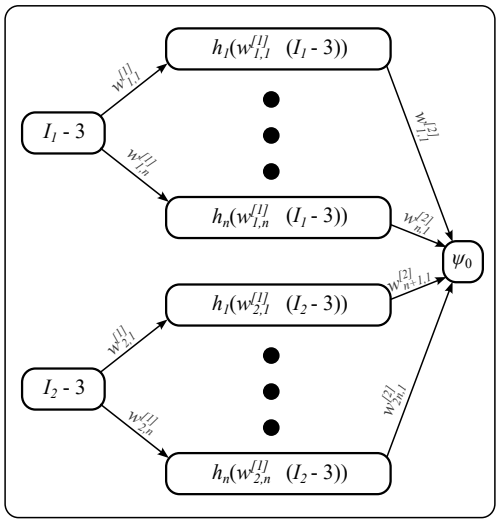
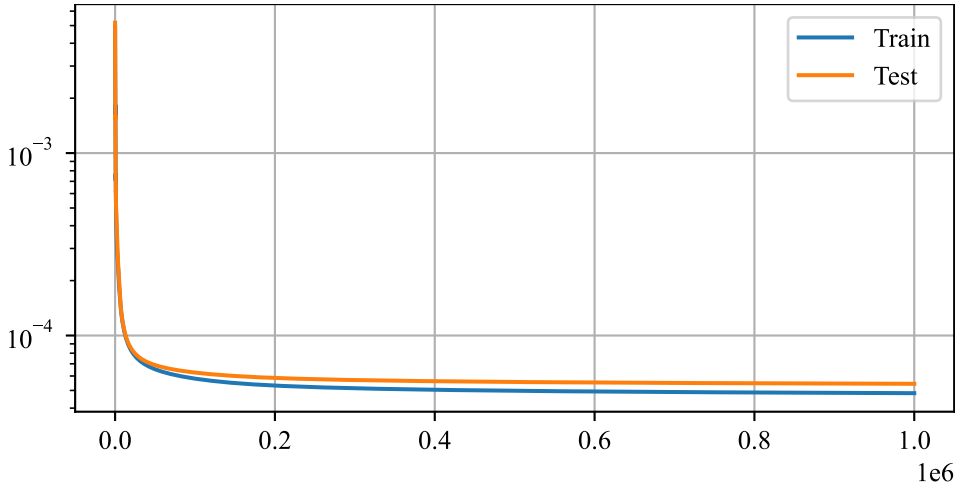


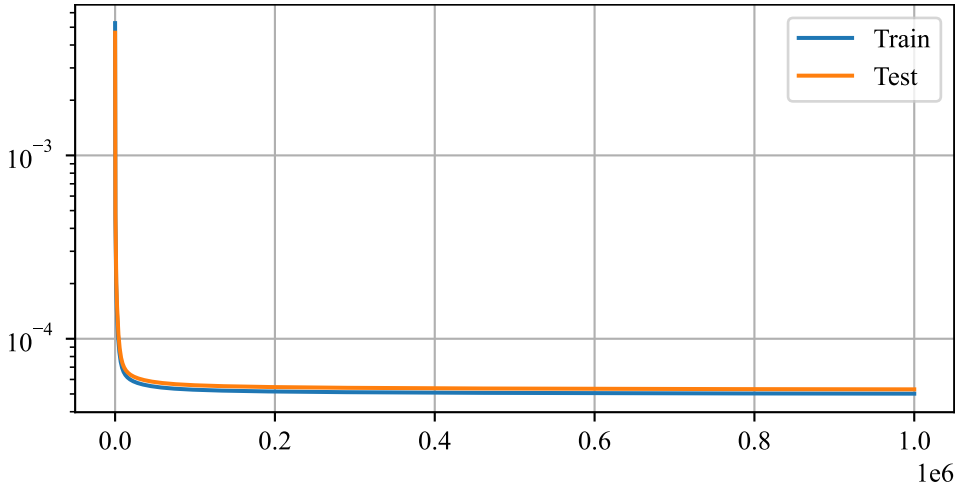
Figure 5.8: The architecture of the NN block in case 5 where the invariants are passed to the hidden layer separately and are not added in the exponent of the activation function.

should also be noted that in this work the *normalisation of energy and stress* are fulfilled *a priori* using the LINEXP-PANN while in [44] such an NN model was not presented. Additionaly, in Fig. 5.9e, case 5, the NN where the invariants are separated in shown. By doing so and adding the subfunctions as $\psi(I_1, I_2) = \psi(I_1) + \psi(I_2)$, the training loss is not impacted much and the best loss is still on par with case 1 at $5.56 \cdot 10^{-5}$. Finally, in Fig. 5.9f, case 6, the CANN is shown and it encountered difficulties during training where it was not guaranteed to start training properly on each run, instead starting to diverge. The best training run that was obtained is shown in Fig. 5.9f with the best loss at $2.77 \cdot 10^{-4}$, after which it started to diverge and NaN values appeared in the loss. It should be noted that these exact values are not reproducible since the weights $w^{[l]}_{i,j}$ are randomly initialised during each training run, but the orders of magnitudes of the errors are always the same.

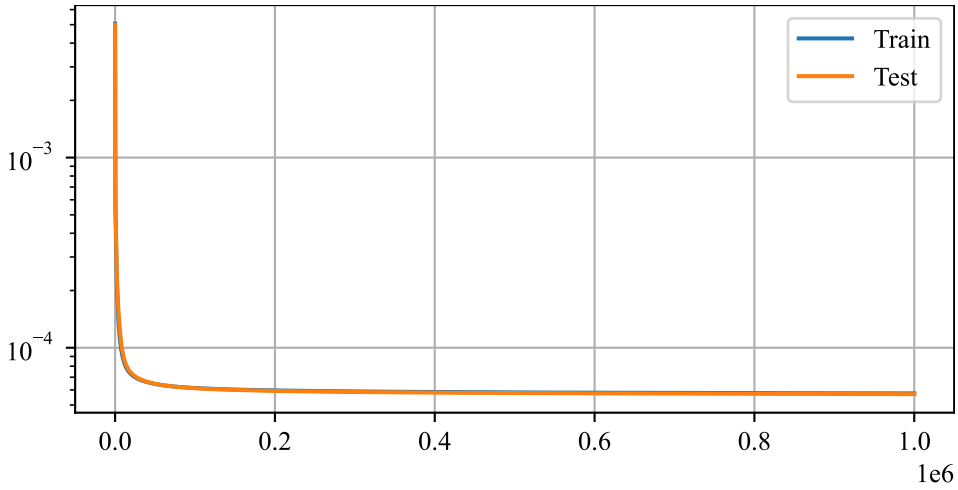
In regards to the *non-negativity of the energy* all the NNs were numerically tested on the invariant domain, see Fig. 3.12, to see if the NNs calculate negative values for the energy, and all NNs passed the test, i.e. none have predicted negative values. This is attributed to the *thermodynamic consistency* where the NNs are trained on stress data, i.e. the derivatives of energy, thus indirectly enforcing the *non-negativity of energy*.



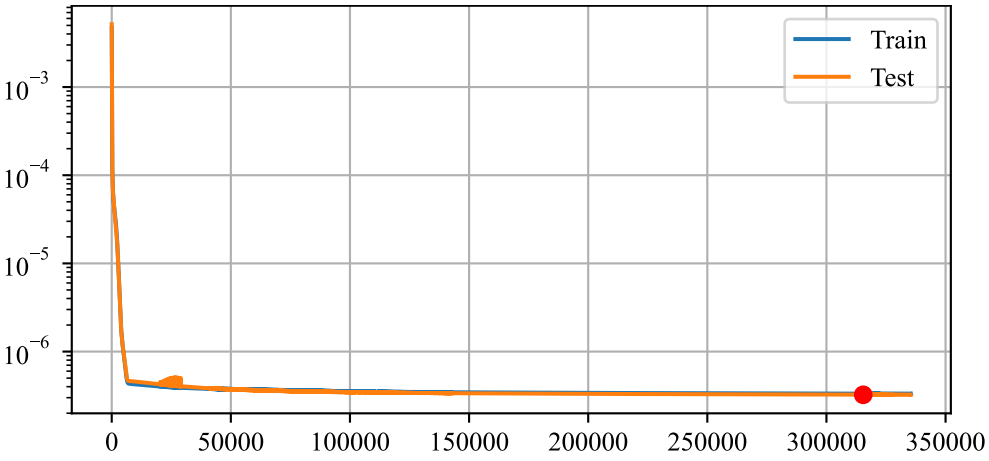
(a) Case 1.



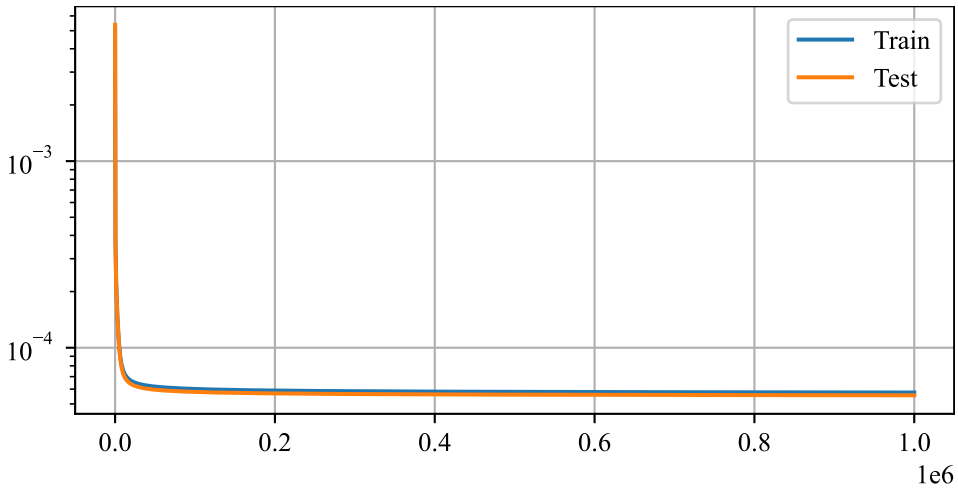
(b) Case 2.



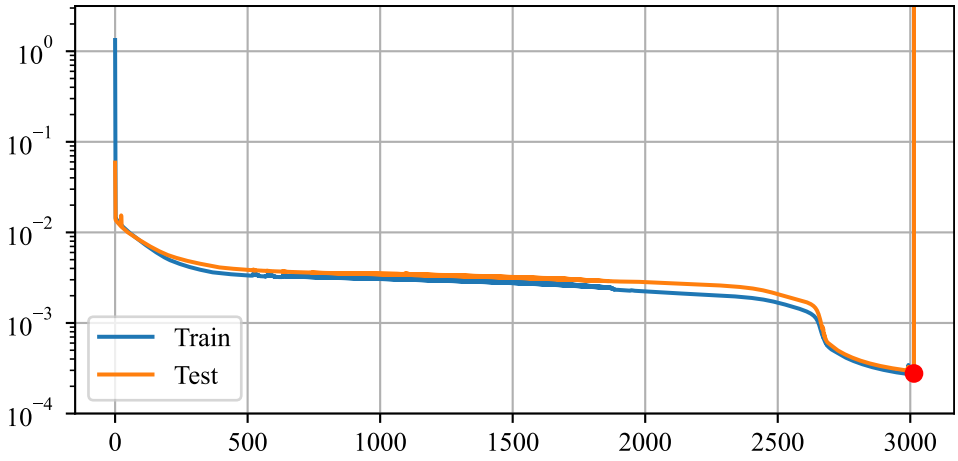
(c) Case 3.



(d) Case 4.



(e) Case 5.



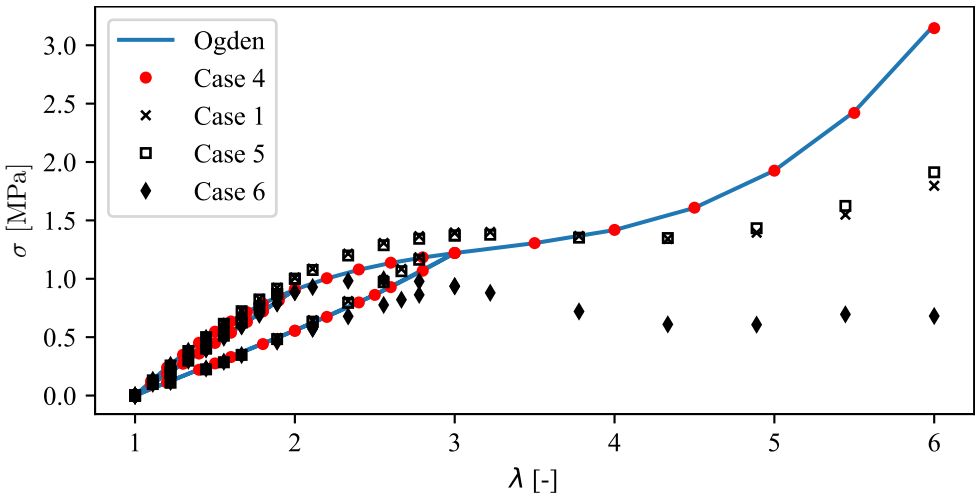
(f) Case 6.

Figure 5.9: Train and test losses during the training of the various proposed NN architectures. Case 4, LINEXP-PANN without constraints, shows lowest losses while case 6, the CANN model, shows highest losses during training. The red dots, where present, show the epoch where the lowest test loss was reached.

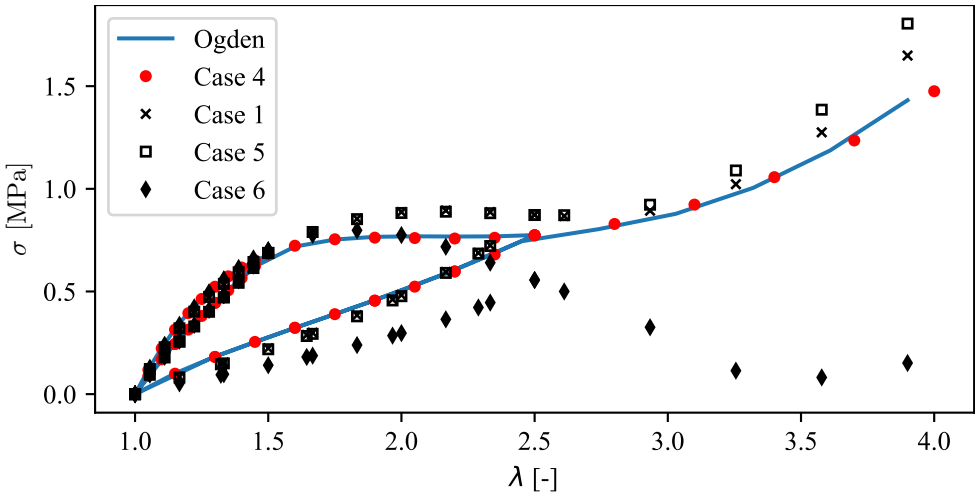
To evaluate the performance of the framework proposed in Fig. 5.4 for the different cases of NNs, a verification of the results via Abaqus is done on simple UT, ET and PT tests with different loadings than the training data. These simple verification tests are done with the following loadings:

- uniaxial $\lambda : 1 \xrightarrow{load} 2 \xrightarrow{unload} 1 \xrightarrow{load} 3 \xrightarrow{unload} 1 \xrightarrow{load} 6$
- equibiaxial $\lambda : 1 \xrightarrow{load} 1.5 \xrightarrow{unload} 1 \xrightarrow{load} 2.5 \xrightarrow{unload} 1 \xrightarrow{load} 4$
- planar tension $\lambda : 1 \xrightarrow{load} 1.75 \xrightarrow{unload} 1 \xrightarrow{load} 2.5 \xrightarrow{unload} 1 \xrightarrow{load} 5$

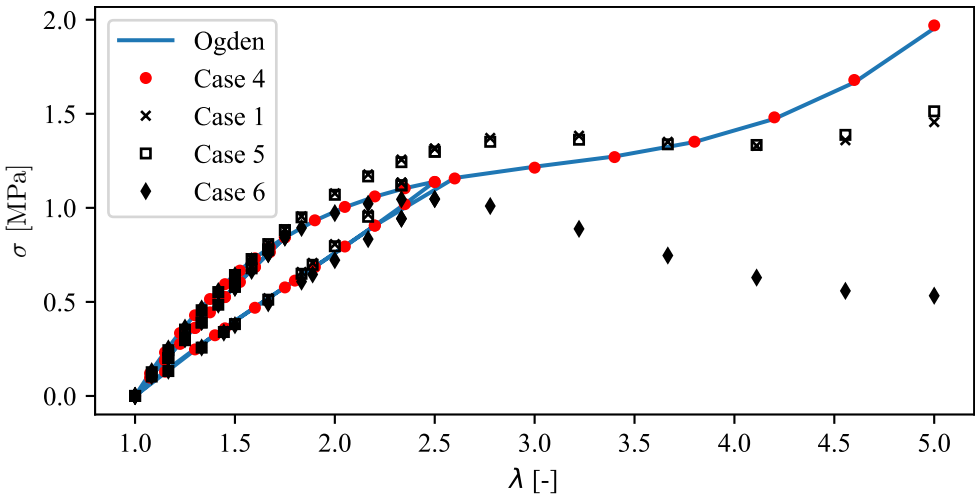
First, an overview of the NNs from case 1, 4, 5, and 6 is given on the aforementioned simple tests with the uniaxial tests shown in Fig. 5.10. To recall, these cases were taken because of their individual importance, case 1 is a polyconvex model with the LINEXP-PANN serving as the base, case 4 also has the LINEXP-PANN as its base but without any imposed restrictions, case 5 is a polyconvex model utilizing the linear exponential activation function but where the functions of the invariants are separated and added which is a more standard approach to modelling (e.g. the Mooney-Rivlin model, the CANN model), and case 6 uses a state of the art CANN model from the literature as a base model.



(a) Uniaxial.



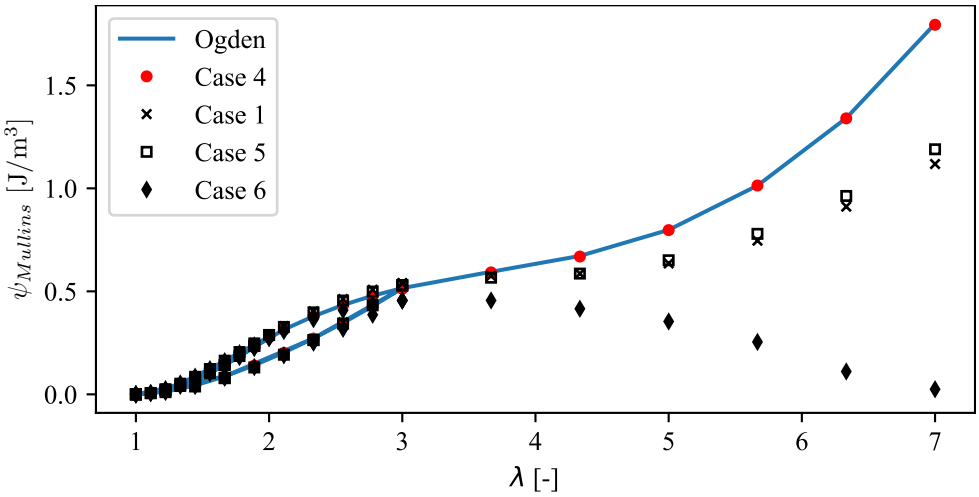
(b) Equibiaxial.



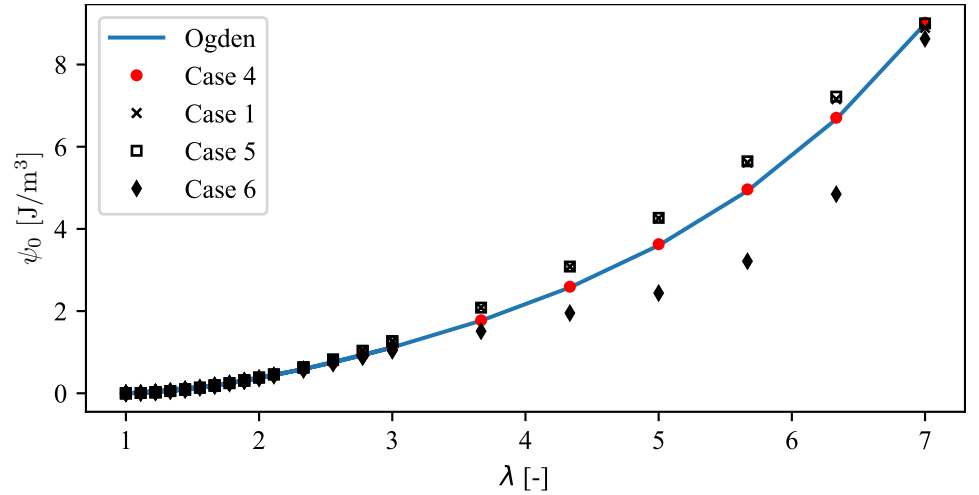
(c) Planar tension.

Figure 5.10: Comparison of the different cases with the base Ogden model with Mullins effect. Case 4, the unconstrained LINEXP-PANN, shows best behaviour in all scenarios. Separating the invariants, case 5, does not show any improvement over the basic polyconvex LINEXP-PANN, case 1. The CANN model shows the worst behaviour, which follows the trend from the training diagram in Fig. 5.9f.

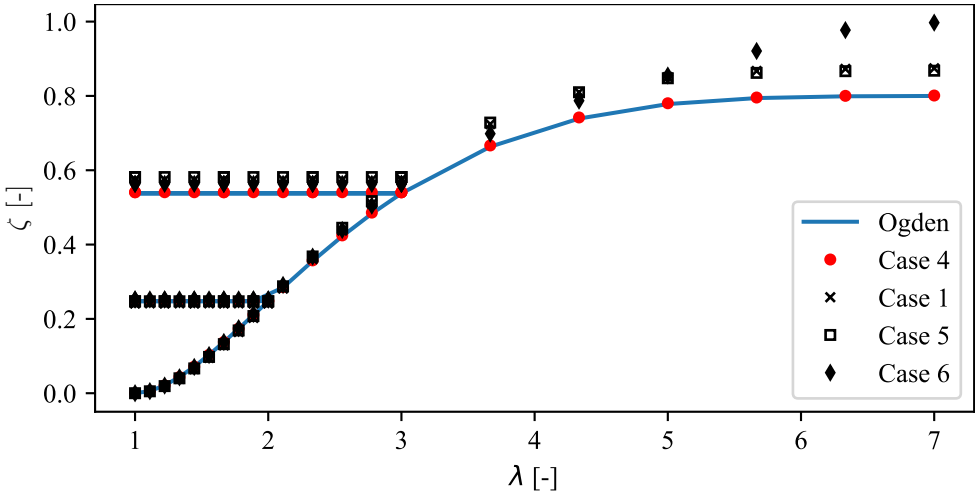
It is also prudent to check if the various models managed to capture the underlying strain energy behaviour even though it was trained on the stresses, and the evolution of the internal damage variable ζ. The ultimate stretch was increased to $\lambda = 7$ to show the evolution of ψ_0 in Fig. 5.11b where all the cases successfully captured the beginning and end of the energy curve, and that in general the deviations were much smaller than those that occur with the inclusion of the Mullins effect or with the stresses. Combining Figs. 5.11b and 5.11c it can be seen that the CANN model, case 6, underestimates the energy in Fig. 5.11a because it overestimates the damage parameter ζ. After examining the results in Figs. 5.10 and 5.11 it can be concluded that case 4, the unconstrained LINEXP-PANN model, is the best performing model.



(a) With Mullins effect.



(b) Undamaged energy.



(c) ζ evolution.

Figure 5.11: Evolution of the damage energy ψ, undamaged energy ψ_0 and damage parameter ζ.

To find out how well has the NN captured the energy evolution, a numerical test can be performed on the invariant domain from Fig. 3.12 since the underlying strain energy function is known. The results are presented in Fig. 5.12 where a highest error of 1.18% is reported at the area in the domain that is furthest from the training bounds noting that although the model was trained on the boundaries of the $I_1 - I_2$ domain it successfully captured the full behaviour of the underlying model. All further results related to the Mullins effect will be presented using the unconstrained LINEXP-PANN model given the accuracy and overall better performance when compared to the other models.

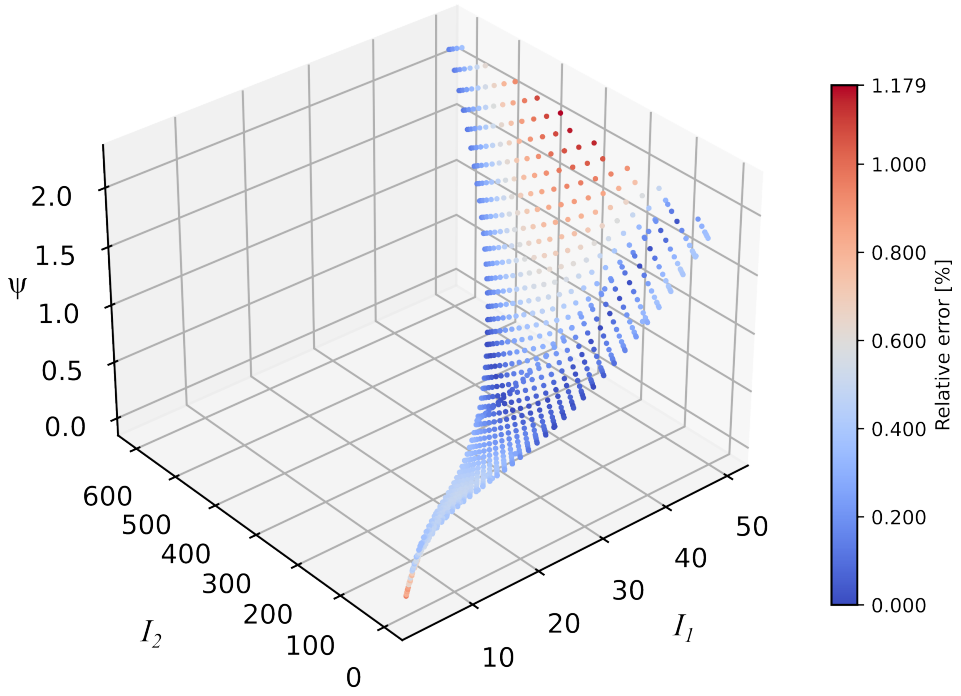


Figure 5.12: Relative error of the predicted Mullins energy, unconstrained LINEXP-PANN from case 4.

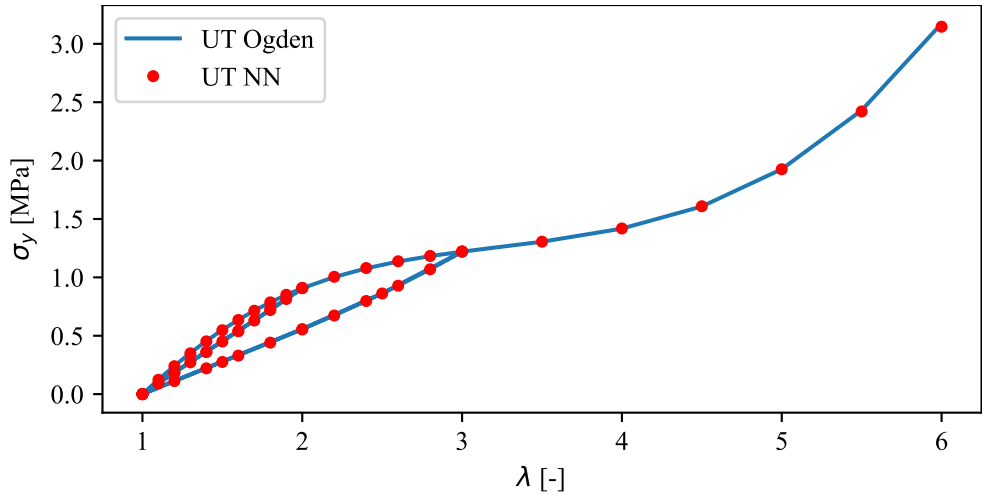
The NN is implemented into Abaqus via the UHYPER subroutine which requires the derivatives of the energy with respect to the invariants, these derivatives were already given in Eqs. (4.19) and (4.20). The outline of the UHYPER subroutine is shown in Algorithm 2. All the numerical examples are performed in 3D using mixed formulation brick elements of type C3D8H.

Algorithm 2 UHYPER procedure.

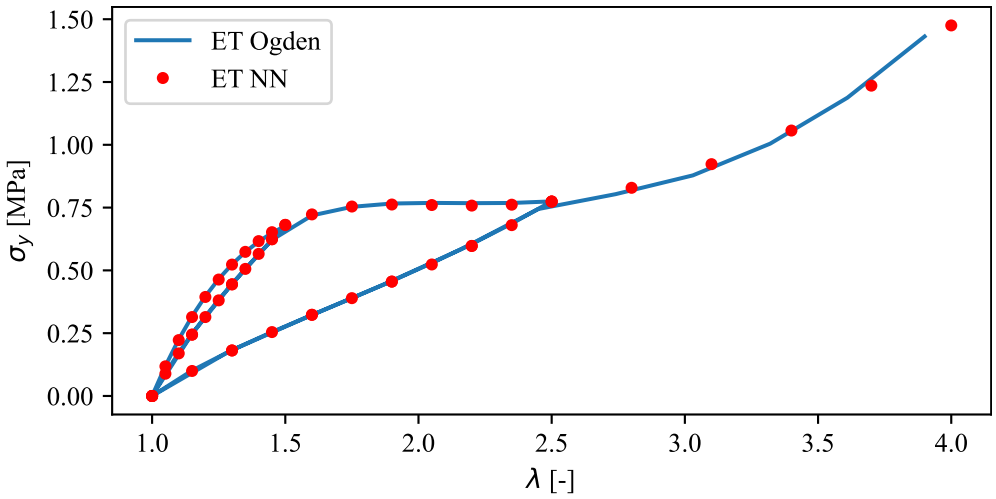
```
PSIM = 0
Calculating undamaged strain energies.
DO J = 1, NNEUR                                   ▷ NNEUR is the number of neurons.
    H = EXP(ALPHA(J)*(W11(J)*(I1MAX-3)+W21(J)*(I2MAX-3))) -1
    PSIM = PSIM + W3(J)*H                         ▷ Maximum energy.
    H = EXP(ALPHA(J)*(W11(J)*(BI1-3)+W21(J)*(BI2-3))) - 1
    DUDI(1) = DUDI(1) + W3(J)*H                   ▷ Current energy.
```

```
 7: END DO
    Check for new maximum energy.
 8: IF (DUDI(1).GT.PSIM) THEN
 9:     I1MAX = BI1
10:     I2MAX = BI2
11:     PSIM = DUDI(1)
12:     STATEV(1) = I1MAX                          ▷ User variable.
13:     STATEV(2) = I2MAX                          ▷ User variable.
14: END IF
    Calculate damage parameter.
15: A = 1D0-ZETA_MAX+ZETA_MAX*EXP(-PSIM/IOTA)
    Calculate the derivatives.
16: DO I = 1, NNEUR
17:     H = EXP(ALPHA(I)*(W11(I)*(BI1-3)+W21(I)*(BI2-3))) - 1
18:     CI = EXP(ALPHA(I)*(W11(I)*(BI1-3)+W21(I)*(BI2-3)))
19:     DUDI(2) = DUDI(2) + W3(I)*ALPHA(I)*W11(I)*CI
20:     DUDI(3) = DUDI(3) + W3(I)*ALPHA(I)*W21(I)*CI
21:     DUDI(4) = DUDI(4) + W3(I)*ALPHA(I)**2*W11(I)**2*CI
22:     DUDI(5) = DUDI(5) + W3(I)*ALPHA(I)**2*W21(I)**2*CI
23:     DUDI(6) = DUDI(6) + W3(I)*ALPHA(I)**2*W11(I)*W21(I)*CI
24: END DO
    Apply damage.
25: DO I = 1, 6
26:     DUDI(I) = A*DUDI(I)
27: END DO
```

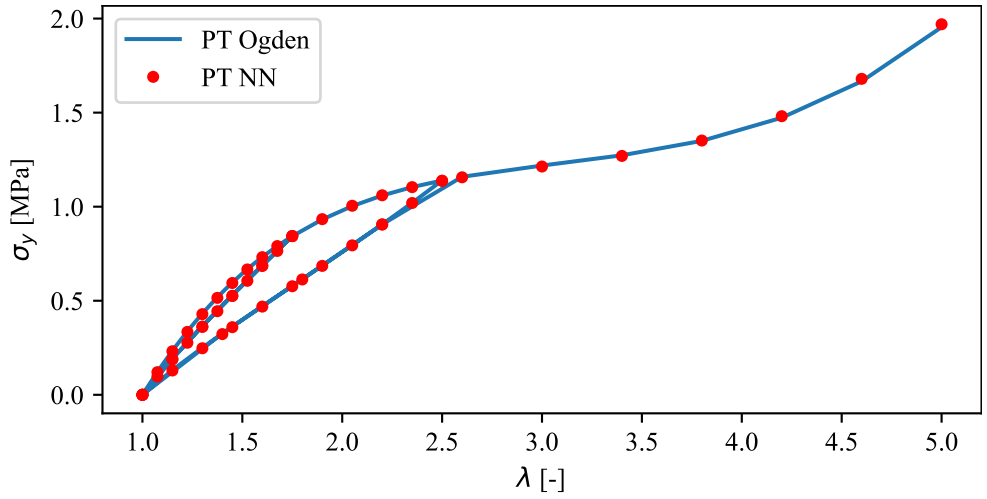
The results for the basic Abaqus tests should be the same as in Fig. 5.10, which would confirm that the application is sound. The results in Fig. 5.13 are shown for the UT, ET and PT cases.



(a) Uniaxial.



(b) Equibiaxial.



(c) Planar tension.

Figure 5.13: Simple test results in Abaqus to confirm the implementation validity.

Another simple test related to the Mullins effect is cyclic loading with an increasing amplitude, i.e. a detailed look into whether the NN model correctly captured the Mullins effect although it is only given 2 unloading curves to train on. The results using the unconstrained LINEXP-PANN are presented in Fig. 5.14 with 20 cycles and a maximum stretch reaching $\lambda = 5$. The results are in agreement with the base Ogden model thus once again proving the LINEXP-PANN has successfully captured the Mullins effect.

More complex loading scenarios would be more indicative of the generalisation capabilities of the approach. As such, 2 numerical examples with complex loading conditions are used to further test the LINEXP-PANN. These are the solid rubber disc example and the diabolo example.

5.1 Solid Rubber Disc Numerical Example

The 1^{st} example is taken from Abaqus' examples [1]. It is a solid rubber disc that is pressed against an analytical surface and then rotated by one full circle. The geometry is shown in Fig. 5.15 with the downward displacement $u = 3.84$ mm shown. The disc is constrained with a rigid tie between the reference point in the centre (S) and the inner circle.

The reaction forces of the LINEXP-PANN model compared to the original Ogden model are given in Fig. 5.16. They are taken at the reference point where the displacement and rotation are prescribed. The NN model shows excellent agreement with the median relative error for the reaction forces of 0.92% and for the reaction moments of 1.03%. For a more detailed insight into the predictive quality of the LIENXP-PANN model the von Mises stress plots are given in Fig. 5.17. The maximum relative stress error is 0.6%. In Fig. 5.17c the absolute stress differences are shown and the largest absolute error is $2.6 \cdot 10^{-3}$ MPa. Additionally the damage parameter values can also be plotted over the geometry and this is shown in Fig. 5.18 where the damage plots are nearly indistinguishable. The values of the damage differ slightly since the values of ζ_∞ and ι that the NN learned are not the identical to those in the material from which the data was generated, but as in Fig. 5.11c the NN successfully captured the behaviour.

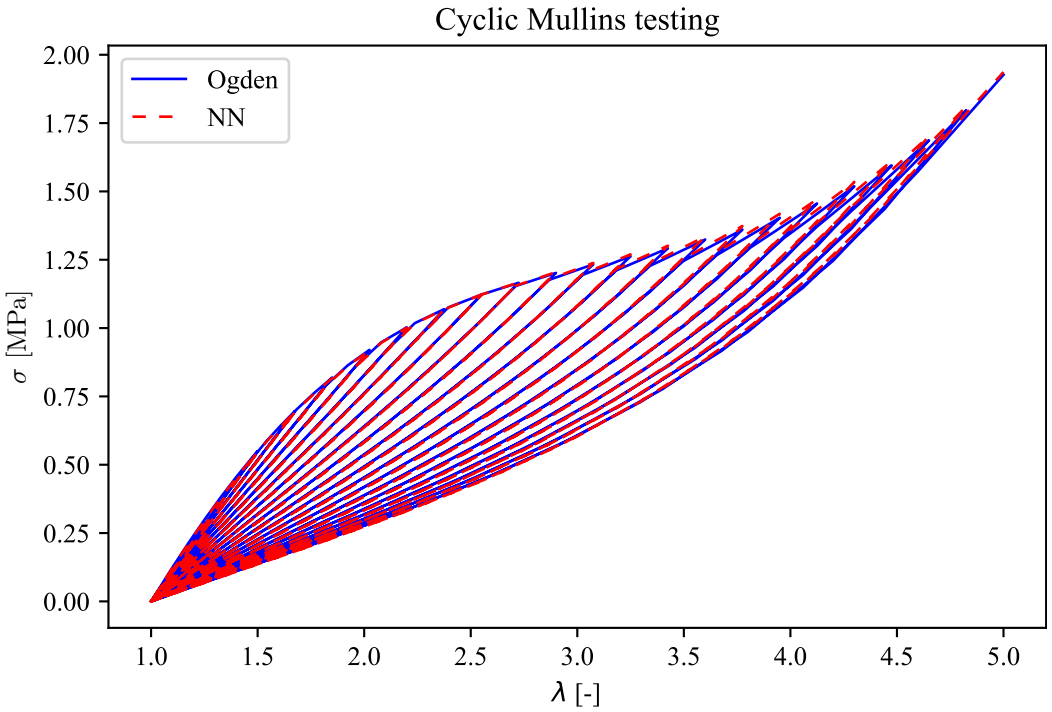


Figure 5.14: Cyclic test in Abaqus with an increasing amplitude, 20 cycles performed.

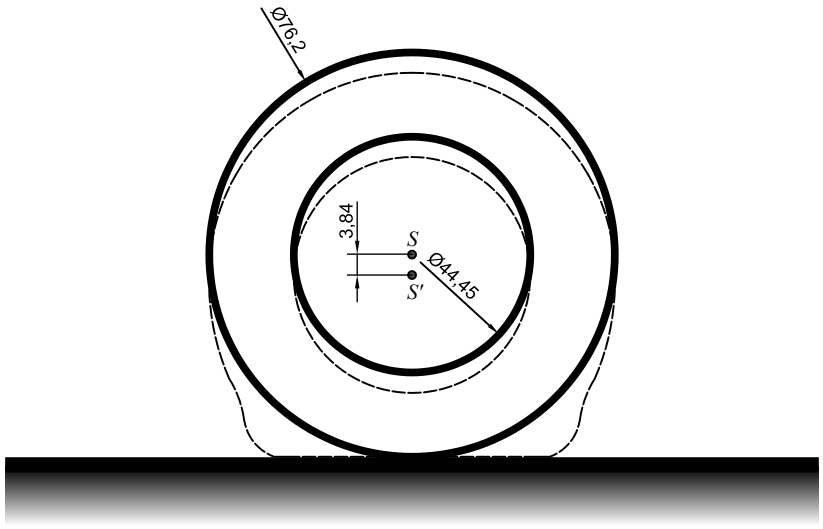


Figure 5.15: Geometry of the solid rubber disc with the deformed configuration shown in dashed lines with the undeformed configuration in full solid lines. Dimensions are given in millimetres. The thickness of the disc is 17.78 mm. A displacement of 3.84 mm followed by a rotation of one full circle are prescribed in the reference point S.

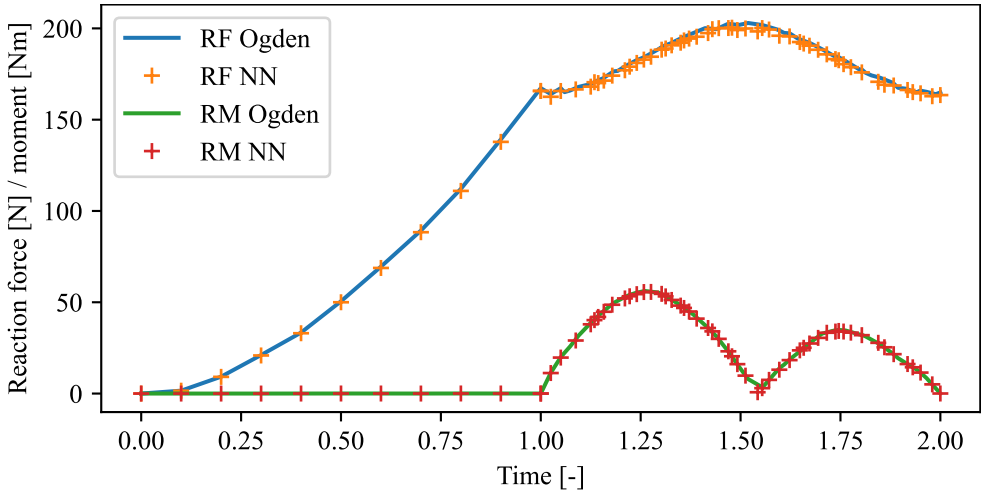
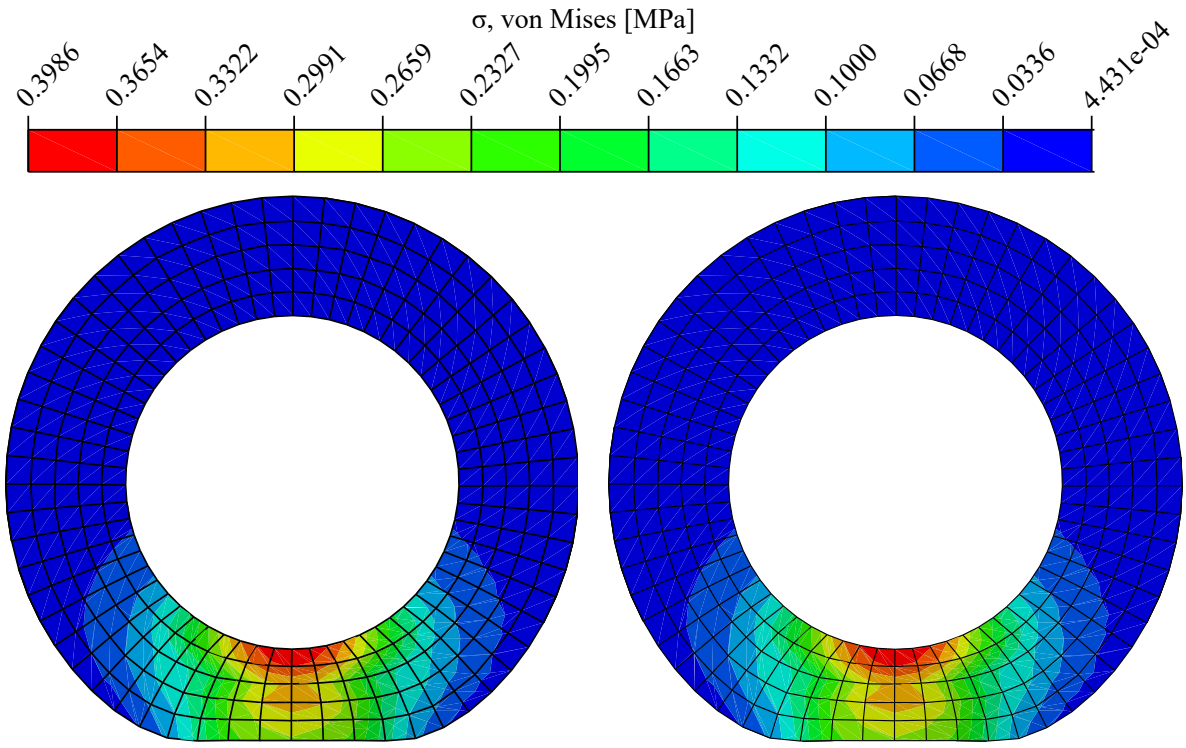
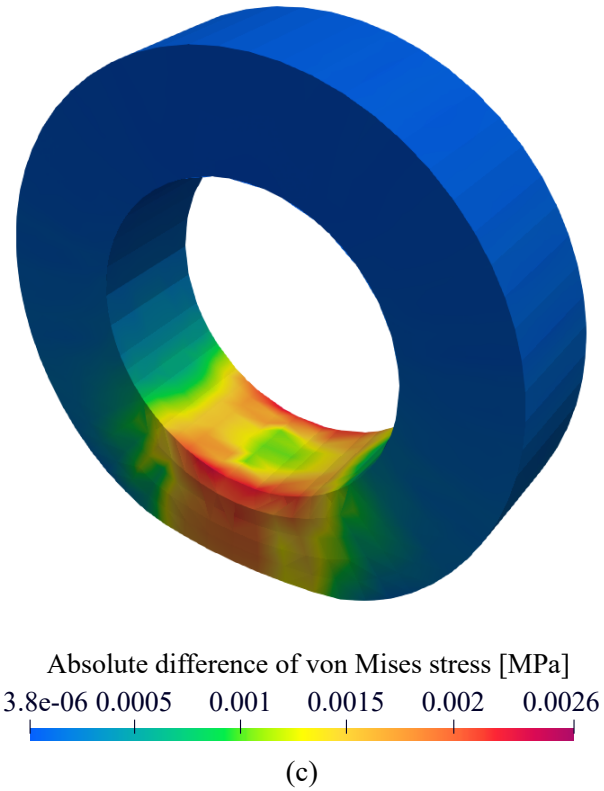


Figure 5.16: Reaction force and reaction moment at the reference point. Comparison between the original Ogden model with Mullins effect and the LINEXP-PANN model.



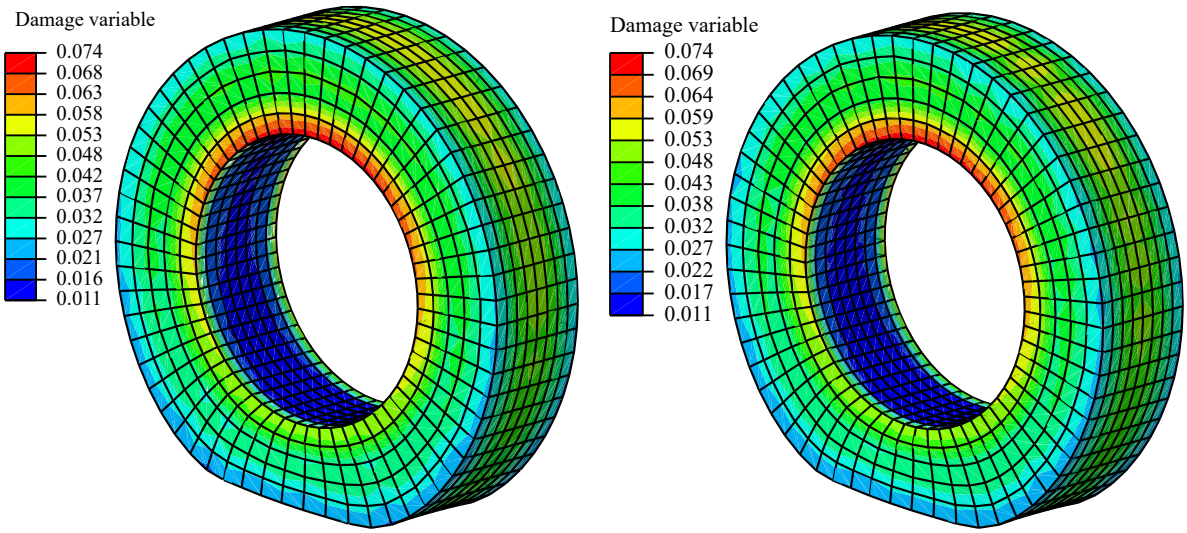
(a) NN solution, max. 0.3962 MPa, min. $4.431 \cdot 10^{-4}$ MPa.

(b) Ogden solution, max. 0.3986 MPa, min. $4.47 \cdot 10^{-4}$ MPa.



(c)

Figure 5.17: Von Mises stress plots of the NN and referent solution, the absolute difference of the solutions is given as well. The relative error of maximum stress is 0.6% and of the minimum 0.88%.



(a) NN model. (b) Ogden model.

Figure 5.18: Plots of damage variable ζ of the NN model and reference Ogden model.

5.2 Diabolo Numerical Example

The second numerical example with complex loading is the diabolo example that was adopted from [13]. It is a simple cylindrical object which is loaded uniaxially and then torsionally. The geometry is given in Fig. 5.19 as well as the position of the uniaxial displacement u and torsional load θ. Multiple loading combinations were presented in the original work where the example is introduced, but in this work only the first loading combination of $u = 30$ mm and $\theta = 5$ rad is taken. As shown in Fig. 5.19 the lower base is fixed and the upper base is rigidly tied to a reference point on which the loads are introduced, similarly to the solid rubber disc example.

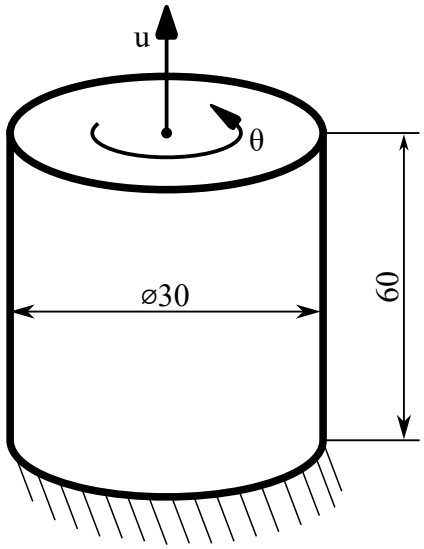


Figure 5.19: Geometry of the diabolo example, fixed boundary condition shown at the bottom and the prescribed displacement u and rotation θ shown at the top.

The reaction forces of the LINEXP-PANN model compared to the original Ogden model are given in Fig. 5.20. They are taken at the reference point where the displacement and rotation are prescribed. The NN model shows excellent agreement with the median relative error for the reaction forces of 0.42% and for the reaction moments of 0.33%.

The von Mises stress plots of the diabolo example are shown in Fig. 5.21 where the solutions with the LINEXP-PANN and Ogden models are shown side by side. The maximum stress values are 0.8864 MPa and 0.8882 MPa, and minimum values are 0.0766 MPa and 0.0769 MPa, respectively. The relative error of the maximum stresses is 0.01% and of the minimum stresses 0.48%.

Additionally, the evolution of the damage parameter ζ is taken at the centre node on the cylinder surface and shown in Fig. 5.22. The median relative error is 0.39%.

Plots of the damage parameter ζ over the entire domain at the end of the simulation are given in Fig. 5.23 and shows agreement between the LINEXP-PANN and Ogden results.

Lastly, plots of the strain energy over the entire domain are given in Fig. 5.24 with a relative median error of 1.05%. This shows that even for complex loading conditions the LNEXP-PANN

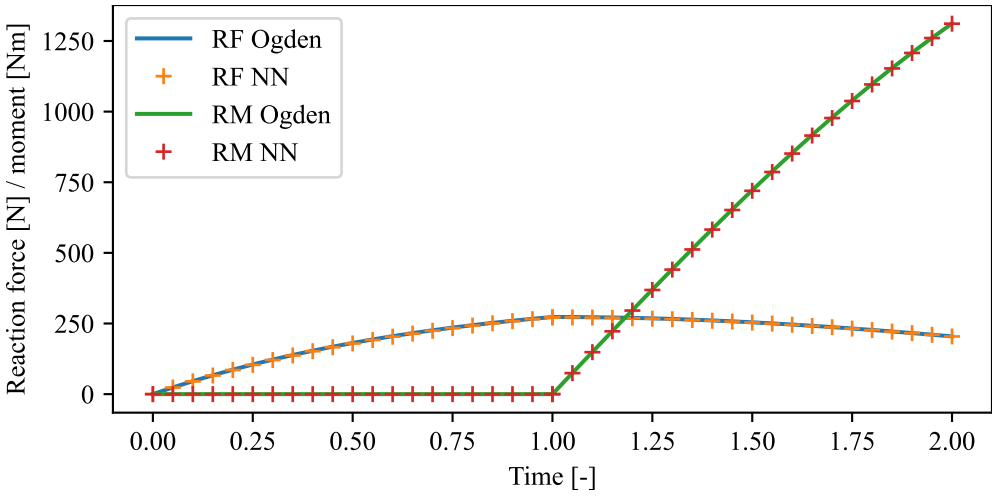
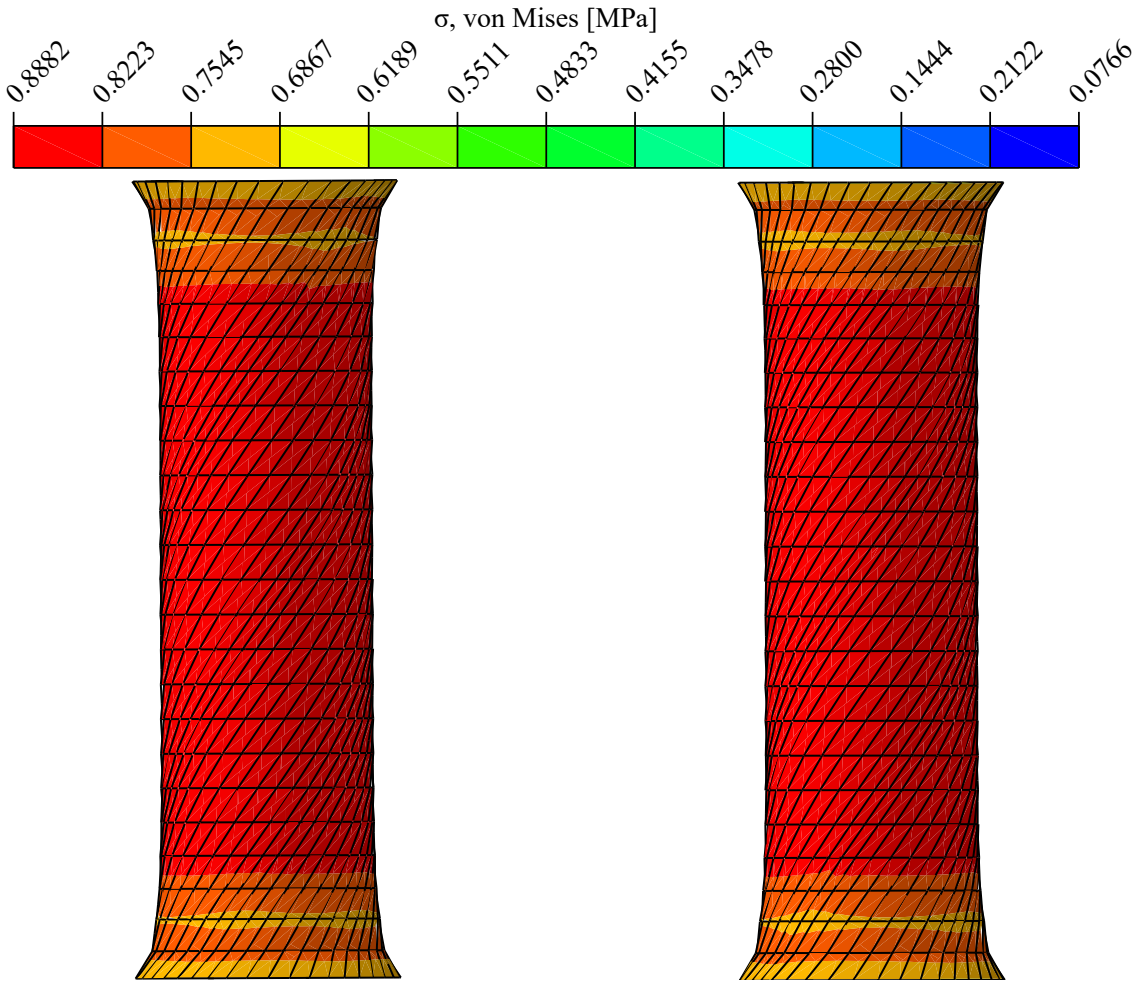


Figure 5.20: Diagram of the reaction force and moment for the diabolo numerical example. Results are taken at the node where displacement and rotation are prescribed.



(a) NN solution. (b) Ogden solution.

Figure 5.21: Von Mises stress plots of the diabolo example using the LINEXP-PANN and the Ogden model with Mullins effect.

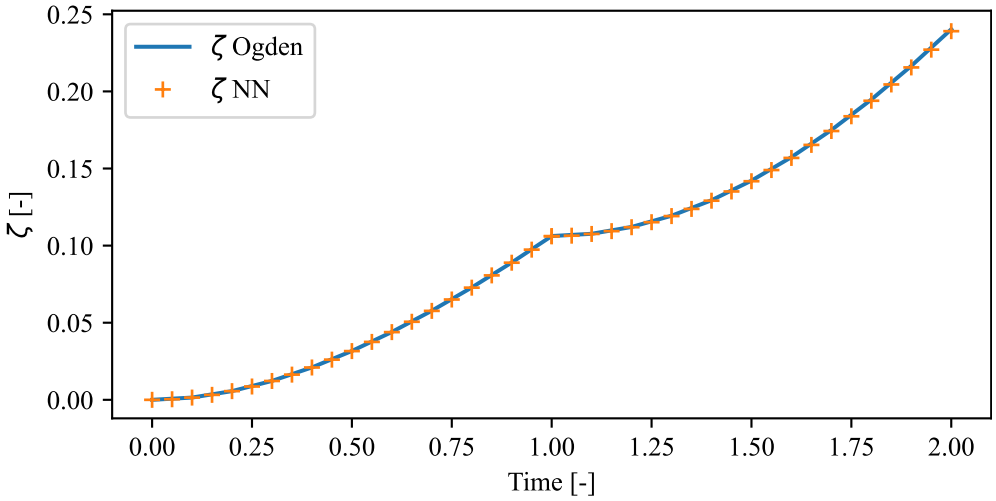
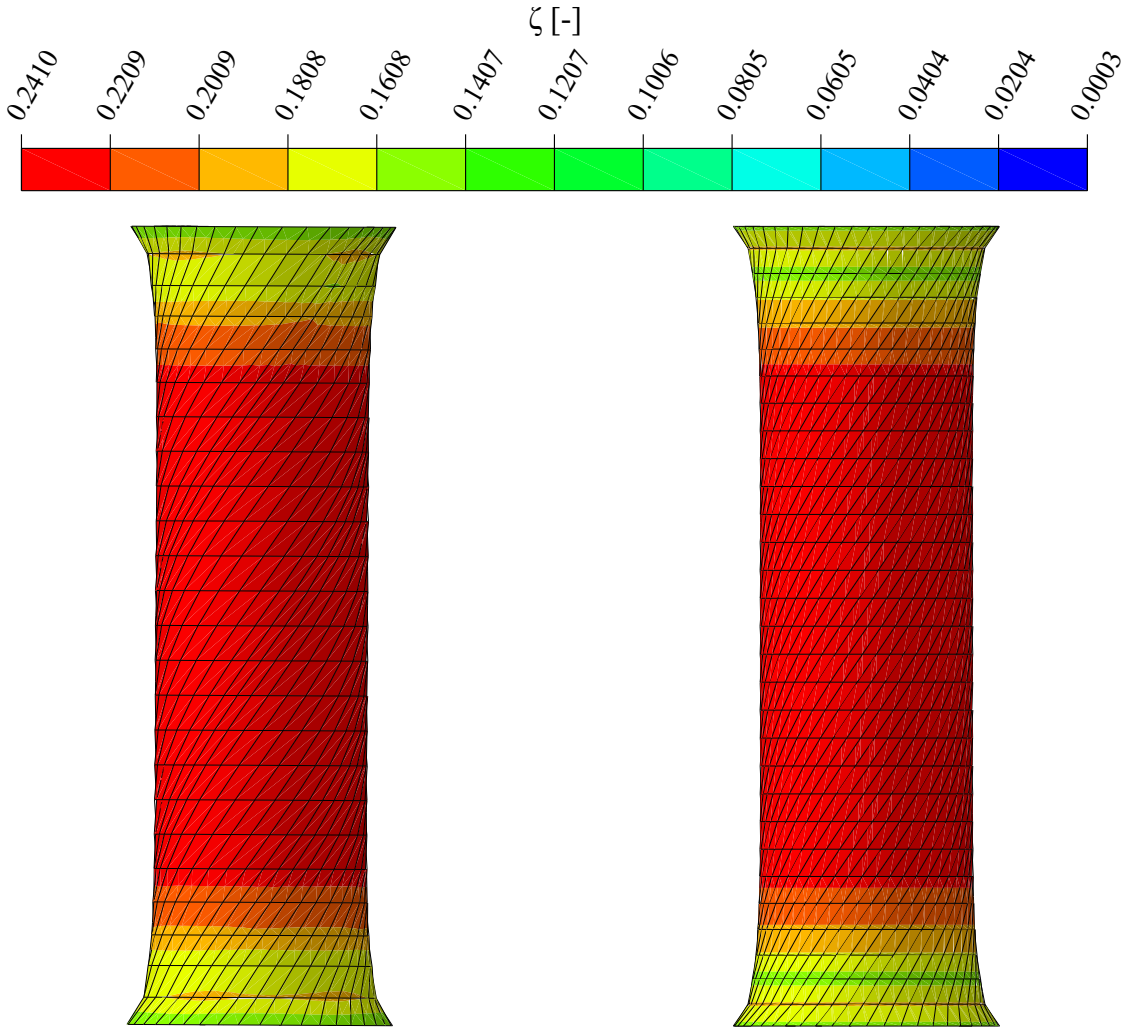


Figure 5.22: Evolution of the damage parameter ζ for the diabolo example. Comparison between the LINEXP-PANN and Ogden model with Mullins effect.

successfully captured the energy evolution even though it was trained only on simple UT, ET and PT tests in plane stress.



(a) NN solution. (b) Ogden solution.

Figure 5.23: Plots of the damage variable ζ on the full diabolo geometry at the end of the simulation.

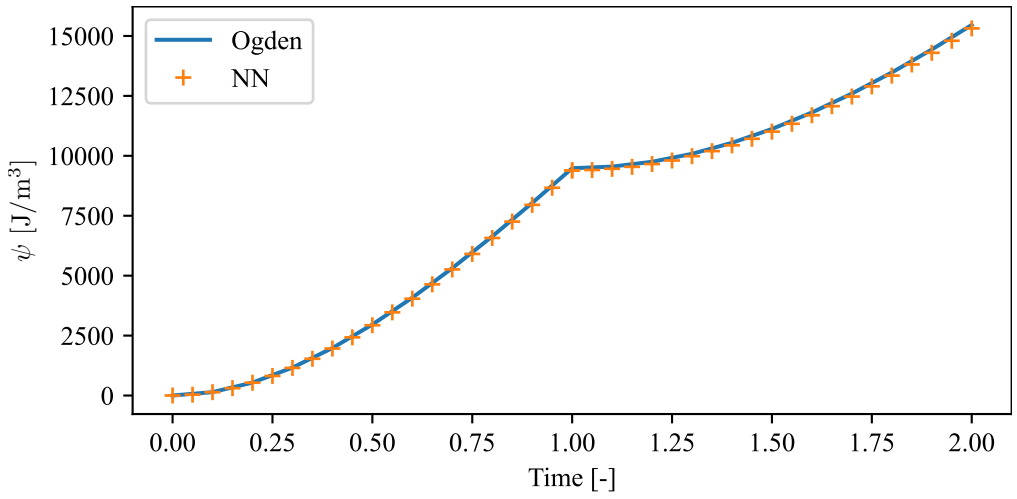


Figure 5.24: Evolution of the Mullins strain energy during simulation for the diabolo example.

5.3 General damage modelling - Mullins subnetworks

In the presented results it is clear that the LINEXP-PANN is a suitable candidate for the general modelling of hyperelastic behaviour. However, the formula for the damage parameter ζ, Eq. (5.8) is simply taken from the literature and implemented as a custom layer in a NN framework. This can be done for any of the candidate functions presented in [16] but this does not truly allow for a general model of the damage parameter ζ in the same sense as the strain energy ψ_0 is modelled by the LINEXP-PANN. In order to obtain a general model for the damage parameter ζ a separate FNN will be introduced that is from this point onwards referred to as a subnetwork.

The architecture of the subnetwork must be such that it satisfies the requirement $\zeta \in [0, 1]$ and $\zeta(\psi_{0,\text{max}}) = 0$. The sigmoid function might seem appropriate since it exactly fulfils the first requirement, but fails to fulfil the second one. A normalization term might be added, but this would violate the first one since its lower bound would not be 0. The hyperbolic tangent satisfies both conditions. It is centred around 0 for an input of 0, thus satisfying the second condition. It is also within the bounds $[-1, 1]$ thus satisfying the first condition, but only if it can be guaranteed that a non-negative argument is passed to it. Since the maximum undamaged strain energy $\psi_{0,\text{max}}$ would be the argument of the function then this is satisfied. However, this would still be an assumption that the damage evolution takes the form of a tanh function which might not be sufficient to capture it. To solve this, a hidden layer is added between the tanh function and $\psi_{0,\text{max}}$, see Fig. 5.25. The linear exponential from Eq. (4.8) is taken as the activation function since it can be restrained to give only positive values and also it has already proven to be capable in capturing non-linear behaviour. Also, using it as the activation function satisfies the requirement $\zeta(\psi_{0,\text{max}}) = 0$. In the subnetworks the activation functions are denoted using g_i to avoid confusion with the activation functions used in LINEXP-PANN in the NN block in Fig. 5.4. The number of neurons in the hidden layer is arbitrary, but in this work it was chosen to be 5. An additional parameter $\beta \in [-1, 1]$ is added as a multiplicator to the tanh in order to restrict the maximum value of damage in a similar manner to ζ_∞ in Eq. (5.2) and ζ_{max} in Eq. (5.8). The architecture is illustrated in Fig. 5.25.

The remainder of this section examines the approach to modelling the damage parameter using an NN approach that meets only the most basic physically meaningful requirements. The test previously done in Sec. 5 will be repeated. Also, two versions of the subnetwork are investigated. The first one is the one presented in Fig. 5.25 with β being a trainable parameter, and in the second version β is omitted (it is fixed to be equal to 1 and non-trainable) to allow the NN more freedom in capturing the damage evolution, since it already satisfies the conditions $\zeta \in [0, 1]$ and $\zeta(\psi_{0,\text{max}}) = 0$. For the basic NN block only the unrestricted LINEXP-PANN is taken for modelling ψ_0 as it is the best performing of the previously investigated NN architectures.

The training is performed on the same datasets as for the NNs presented in Fig. 5.9. The

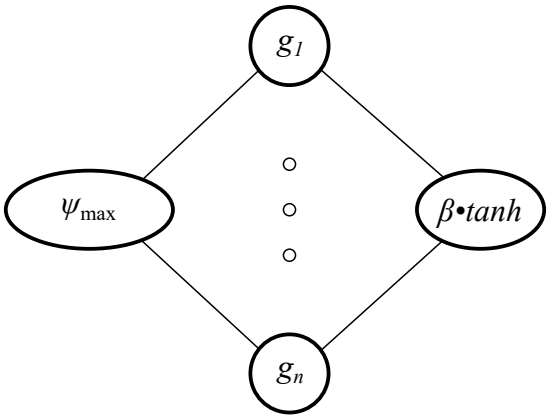
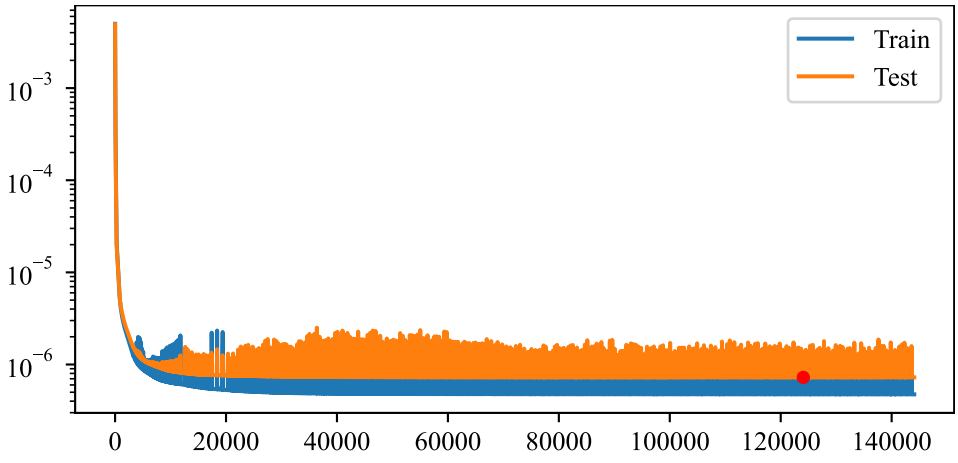
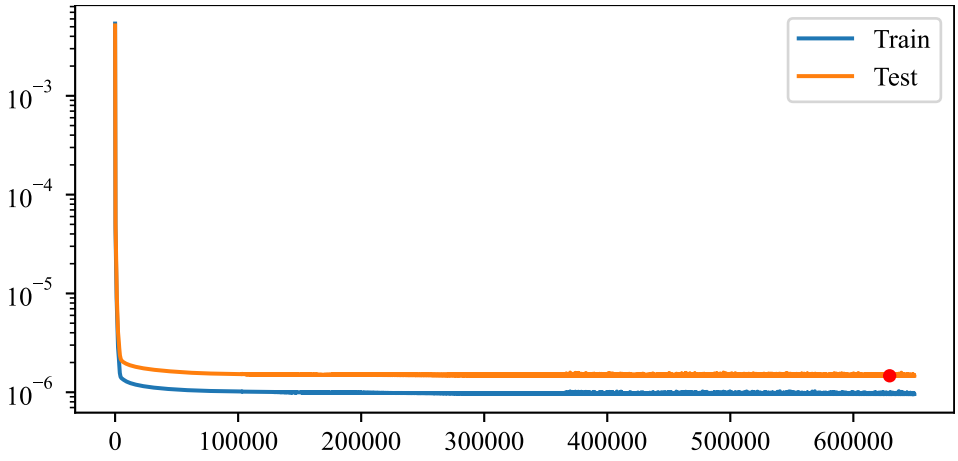


Figure 5.25: Architecture of the damage subnetwork.

training curves for the NNs are presented in Fig. 5.26 and they show that the training losses are about the same with the one in Fig. 5.26a being somewhat lower. Also, it is noted that when β is included the training becomes smoother, i.e. the loss does not oscillate much which is different than when β is omitted. After training, the parameter β was obtained as 0.86 which, although different from the base ζ_∞ of 0.9 is somewhat close.



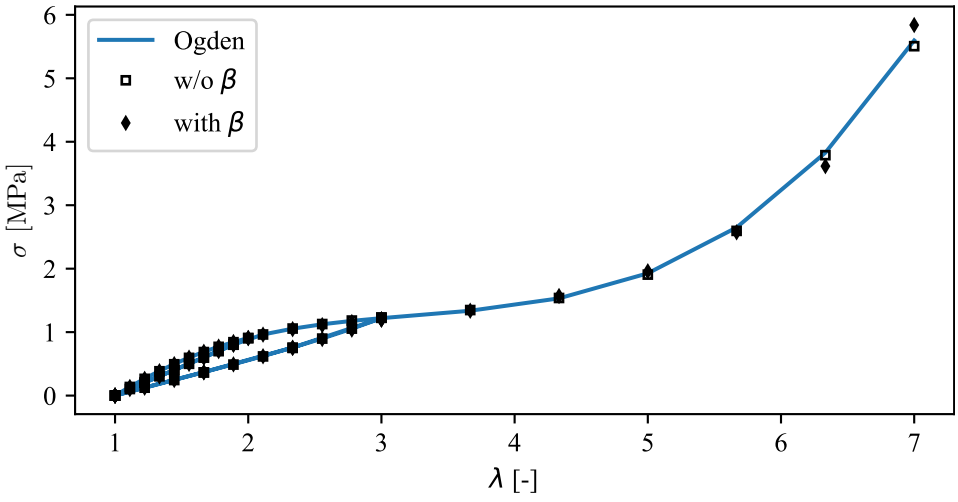
(a) β is omitted.



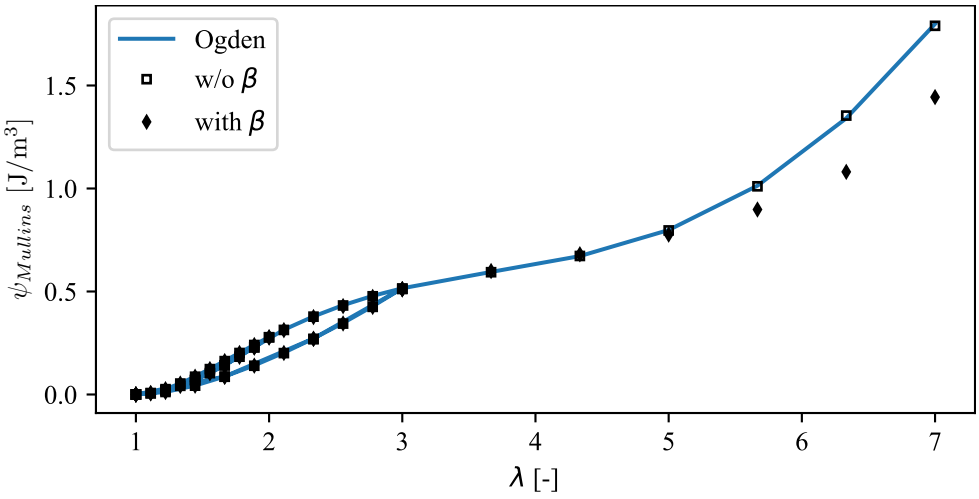
(b) β is included.

Figure 5.26: Training histories of the two versions of the Mullins subnetwork. In the bottom case at the end of training $\beta = 0.86$.

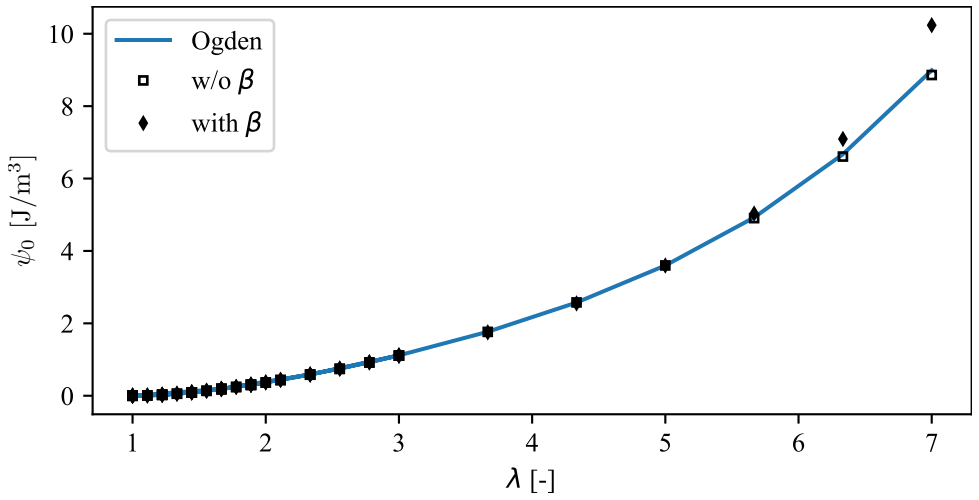
The approach is first presented on uniaxial tension, with the results presented in the same manner as before in Fig. 5.11. In all the results presented in Fig. 5.27 the NN where β is omitted has outperformed the one where it is included as a trainable parameter. It is interesting to compare ψ_{Mullins} and ψ_0 and the parameter β. The NN recovered a higher value of the maximum damage and thus it compensated by predicting a stronger material so ψ_0 is overestimated. In this way the stress is recovered but the learned ψ_{Mullins} is still underestimated at the largest stretch.



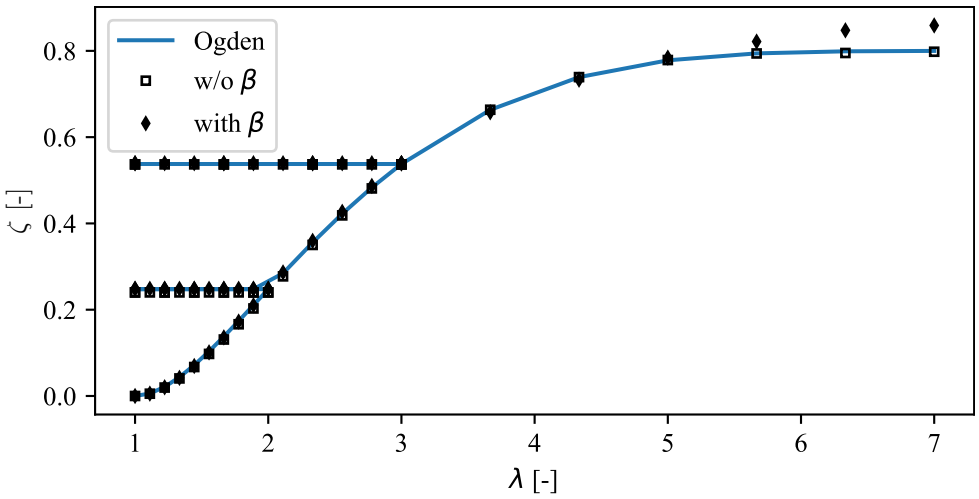
(a)



(b)



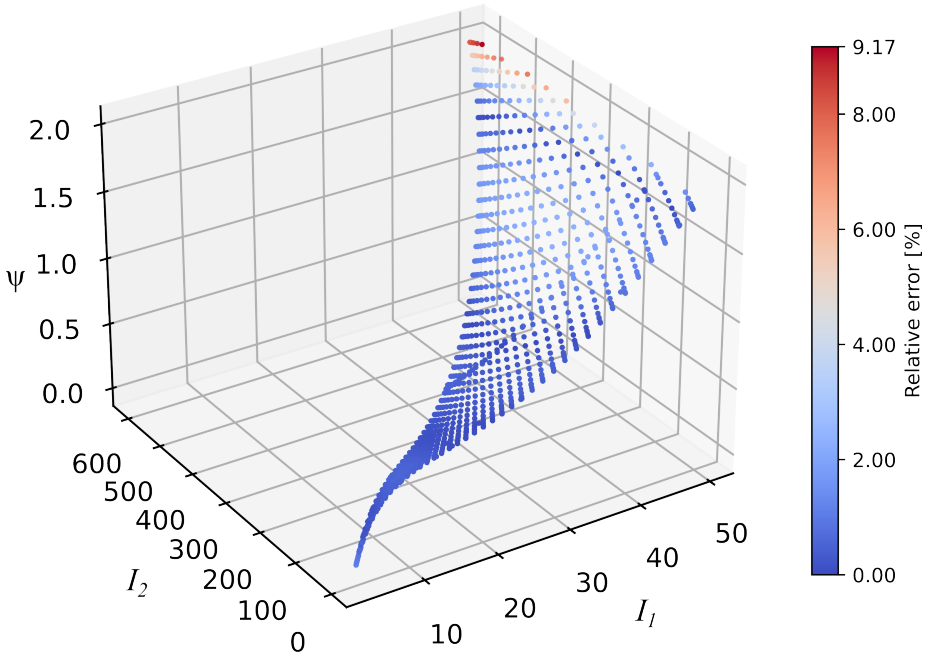
(c)



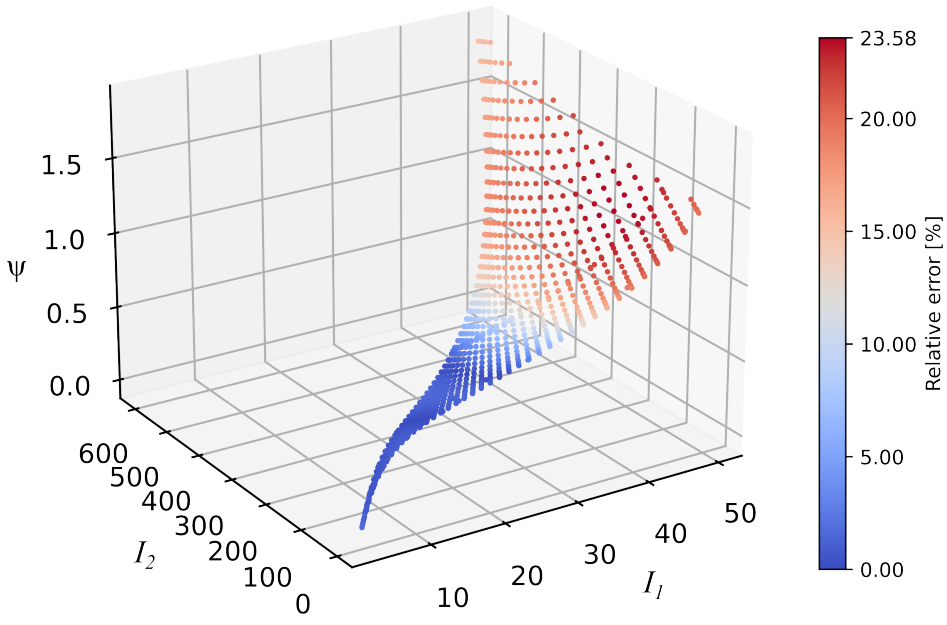
(d)

Figure 5.27: Results of the Cauchy stress (a), damaged (b) and undamaged (c) energy, and damage parameter (d) evolution for the LINEXP-PANN with a subnetwork for modelling the damage parameter ζ in place of the expression from Eq. (5.2). The value of β obtained by the NN where it is a trainable parameter is 0.86.

The errors of ψ_{Mullins} predicted by the NNs with a subnetwork are given in Fig. 5.28 over the invariant domain. Following the previous results from Fig. 5.27, when using the subnetwork with β omitted the results are much better. The largest error is larger from LINEXP-PANN with ζ defined by Eq. (5.8) which is expected, but it allows for the capture of more complicated behaviours and a general approach to modelling the damage evolution.



(a) β is omitted.

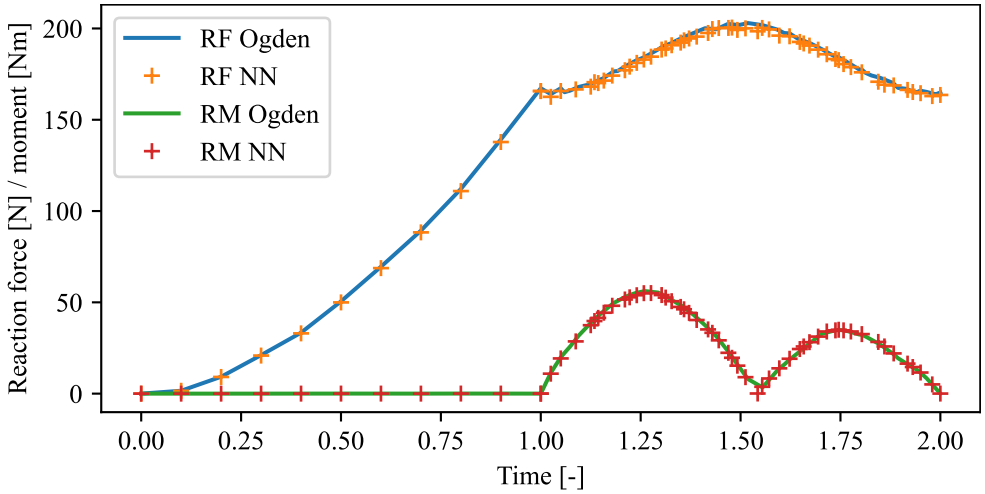


(b) $\beta = 0.86$.

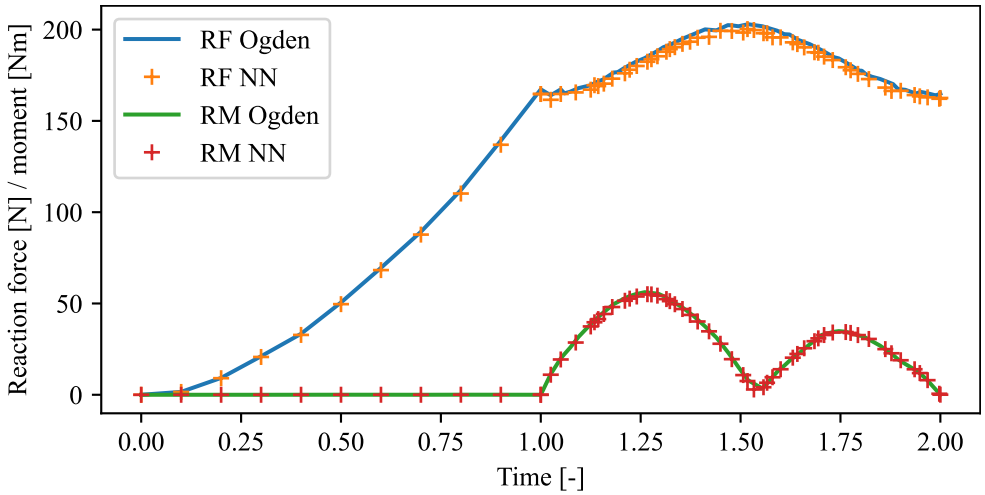
Figure 5.28: Relative errors of the predicted ψ_{Mullins} over the invariant domain from Fig. 3.12.

Following the results presented so far it is safe to assume that using a subnetwork with β omitted is the preferable option due to the lower errors and better prediction qualities. This

is further investigated using more complex numerical examples. Following the outline of the previous results from Sec. 5.1, the reaction forces and moments of the solid rubber disc example using the subnetworks are shown in Fig. 5.29. The reaction force and moment median errors for the subnetwork with β omitted are 0.9% and 1.8%, and for the subnetwork with $\beta = 0.86$ the errors are 1.5 % and 1.74%. Just by comparing the reaction force and moment curves it is difficult to distinguish between the results of the subnetworks showing that both can be used in this scenario.



(a) β is omitted.

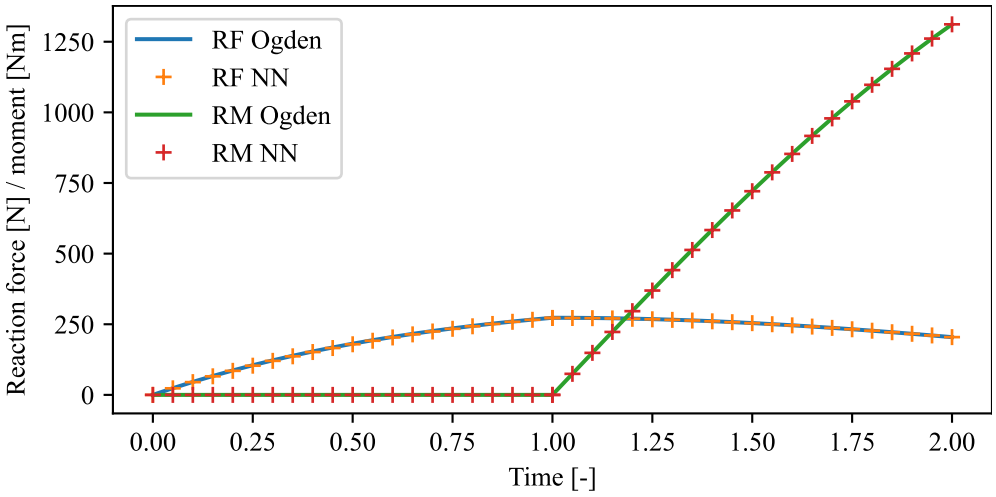


(b) $\beta = 0.86$.

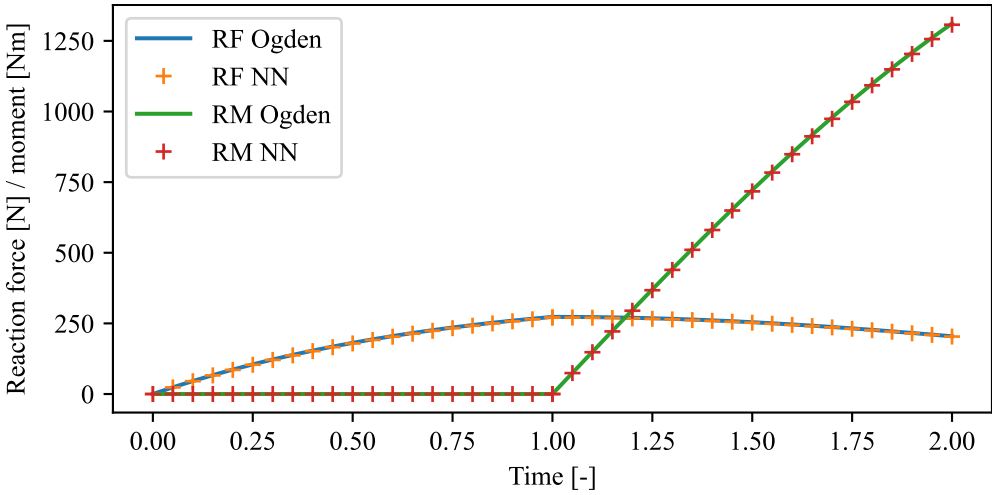
Figure 5.29: Reaction force and moment for the solid rubber disc example at the reference point where the displacement and rotation are prescribed. Results shown using the subnetworks for modelling the damage parameter ζ.

Moving on to the diabolo example taken from Sec. 5.2, the reaction force and moment are shown in Fig. 5.30 with the median errors of 0.24% and 0.15% for the subnetwork with β omitted, and 0.76% and 0.65% for the subnetwork with $\beta = 0.86$. As in the previous example, this shows excellent agreement regardless of the chosen modelling strategy. The evolution of the damage variable ζ is shown in Fig. 5.31 for both subnetworks with the subnetwork where β is

omitted showing better agreement with the reference model on par with what the specialised function from Eq. (5.8).

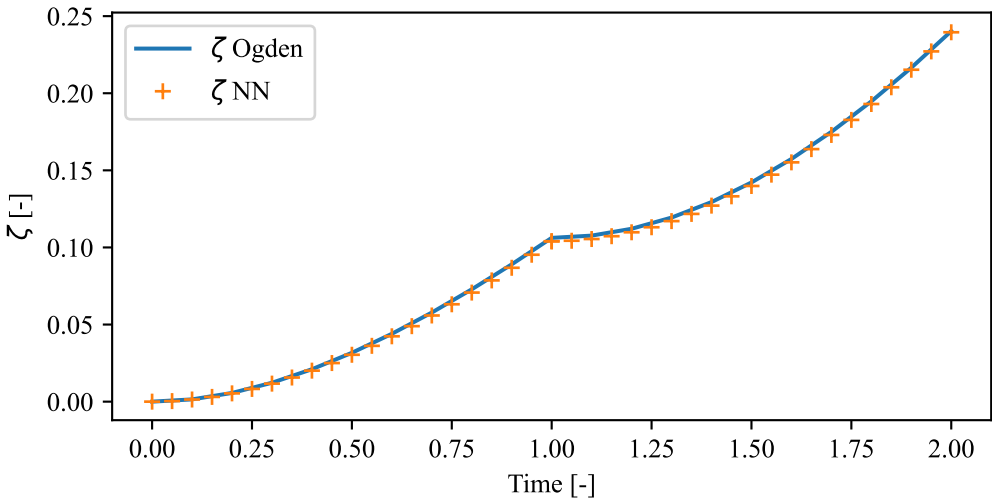


(a) β is omitted.

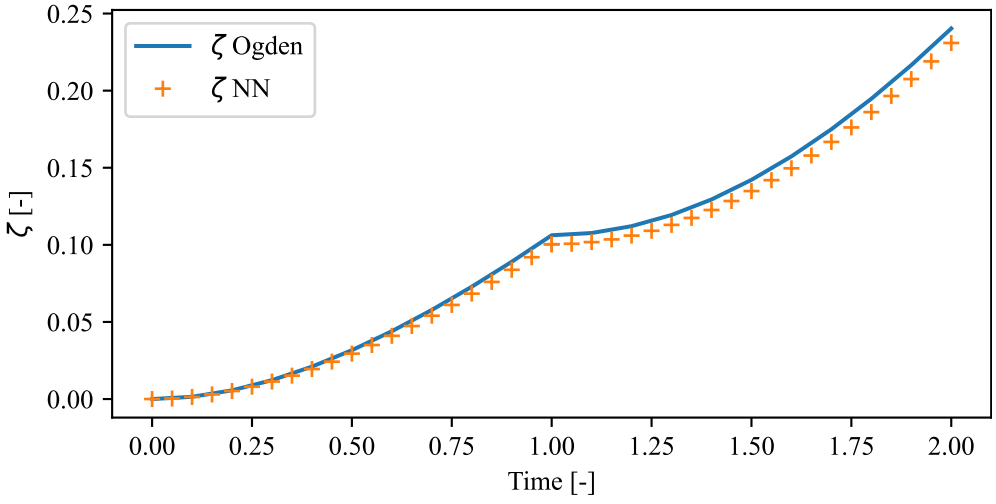


(b) $\beta = 0.86$.

Figure 5.30: Reaction force and moment for the diabolo example at the reference point where the displacement and rotation are prescribed. Results shown using the subnetworks for modelling the damage parameter ζ.



(a) β is omitted.



(b) $\beta = 0.86$.

Figure 5.31: Damage evolution for the diabolo example at the centre node of a side surface. Results shown using the subnetworks for modelling the damage parameter ζ.

Finally, it can be concluded that using the subnetworks to model the parameter ζ is a viable and potentially the preferable option since it allows for a more flexible approach without *a priori* assuming the damage evolution function.

5.3.1 On the Implementation of Subnetworks

When implementing a NN with a subnetwork that models the damage a slight modification is necessary. In place of the specialized function implemented in Algorithm 2 line 15, that line of code is replaced with the code snippet given in Algorithm 3. It is the code representation of the FNN presented in Fig. 5.25.

Remark 1 If the subnetwork without the trainable parameter β is implemented, then it is possible that for high values of $\psi_{0,\max}$ the damage ζ is equal to 1 and in Algorithm 3 in line 6 the value can be 0 thus rendering all the derivatives 0. This does not occur when simple tests like uniaxial, equibiaxial or planar tension are calculated but it does occur in complex loading scenarios such as the solid disc example. To avoid this the variable W6 should be set to nearly 1 (e.g. 0.99999).

Remark 2 If an NN with β is trained it is possible that it trains to mimick the situation from Remark 1 so that β is trained to be nearly 1 (i.e. 0.99999 or similar) and the NN behaves exactly like the one where β is omitted, i.e. it correctly predicts the Mullins energy, the undeformed energy and the damage variable, as in Fig. 5.27. This is however a niche situation and in most cases β is less than 1 like in Fig. 5.27, so it is not necessarily reproducible. However, training without β always yields almost the same result (a small variance exists due to the random initialization of the weights) and is therefore preferred.

Algorithm 3 Subnetwork snippet.

```
   Subnetwork is used to calculate energy from PSIM.
1: AUX = 0D0
2: DO I = 1, 5
3:     AUX = AUX + (EXP(ALPHA2(I)*W4(I)*PSIM)-1)*W5(I)
                          ▷ W4, ALPHA2, W5 are the subnetwork weights.
4: END DO
5: ZETA = W6*TANH(AUX)                    ▷ W6 is either β or 0.99999.
6: A = 1D0 - ZETA
```

6 Extension to Compressible Hyperelasticity and Comparison with Data-Driven Computational Mechanics

An ICNN based on the LINEXP-PANN architecture from Sec. 4 is used to model compressible hyperelasticity, similar to case 2 in Sec. 5. The inputs are expanded to include the third invariant I_3 since compressible behaviour will be modelled. Note that in the undeformed state $I_3 = 1$, so the input is actually taken as $(I_3 - 1)$. The final NN architecture is a modification of the NN proposed in Fig. 4.5 and is shown in Fig. 6.1. The convexity of the NN w.r.t. its inputs is guaranteed by constraining the weights in between the hidden and output layer $w_{i,j}^{[2]}$ to be non-negative. *Objectivity* is accounted for by using the invariants as inputs, *normalisation of the energy* is guaranteed in the same manner as introduced in Sec. 4, with the third invariant in the input as $(I_3 - 1)$. *Thermodynamic consistency* is accounted for by training the NN on its derivatives, the expression for the 2$^{\text{nd}}$ Piola-Kirchhoff stress in case of compressibility is taken as

$$\begin{aligned} \mathbf{S} = 2\frac{\partial \psi(I_1, I_2, I_3)}{\partial \mathbf{C}} = 2\left[\frac{\partial \psi}{\partial I_1}\frac{\partial I_1}{\partial \mathbf{C}} + \frac{\partial \psi}{\partial I_2}\frac{\partial I_2}{\partial \mathbf{C}} + \frac{\partial \psi}{\partial I_3}\frac{\partial I_3}{\partial \mathbf{C}}\right] = \\ = 2\left[\left(\frac{\partial \psi}{\partial I_1} + I_1\frac{\partial \psi}{\partial I_2}\right)\mathbf{I} - \frac{\partial \psi}{\partial I_2}\mathbf{C} + I_3\frac{\partial \psi}{\partial I_3}\mathbf{C}^{-1}\right]. \end{aligned} \tag{6.1}$$

The *normalisation of stress* in the undeformed configuration was not performed so the normalisation constant $\mathfrak{n}$ from Eq. (4.14) is not included in the energy as in Eq. (4.13). It should be noted that in [72] the behaviour of NNs trained directly on energy was also investigated, i.e. without fulfilling *thermodynamic consistency* which is similar to the invariant based NN model presented in Sec. 3.3, but it proved to be a significantly less accurate model with large variance between different NN models trained on the same dataset.

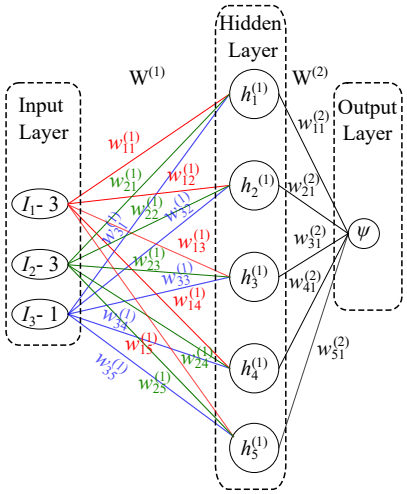


Figure 6.1: Illustration of the ICNN (Input Convex Neural Network) used to model compressible hyperelasticity. Convexity w.r.t. the inputs is guaranteed by choosing the linear exponential activation function and constraining the weights $w_{i,j}^{[2]}$ to be non-negative.

In this section a short comparison is shown with another emerging approach to data-driven modelling called Data-Driven Computational Mechanics (DDCM) which was first published in [36]. The approach was expanded to hyperelasticity in [57] and further refined in [72]. This section presents the results of [72], where a comparison was made between two data-based approaches, the NN model and data-driven model-free approaches.

The aim of DDCM is to minimise the distance between points in a (experimental) dataset. For the one dimensional linearly elastic case, as taken from [36]. The distance is defined as

$$F_e(\varepsilon_e, \sigma_e) = W_e(\varepsilon_e - \varepsilon'_e) + W^*_e(\sigma_e - \sigma'_e), \tag{6.2}$$

where ε_e is the small strain tensor of a bar finite element (in this particular case it is a scalar given that the problem is one-dimensional), σ_e is the Cauchy stress of a bar element, and ε'_e and σ'_e are values from a dataset (that could have been obtained experimentally). The metrics W_e and W^*_e are defined as

$$W_e = \frac{1}{2} C_e \varepsilon_e^2, \quad W^*_e = \frac{\sigma_e^2}{2C_e}, \tag{6.3}$$

which are usually defined as the strain energy and complementary strain energy densities and C_e is a material constant that could be identified as the Young's modulus in case of uniaxial linear elasticity, but in the context of DDCM it is simply a numerical constant since DDCM is driven by data alone.

The goal is to find values of strain ε_e and stress σ_e (a local state (ε_e,σ_e)) that minimise the distance F_e while at the same time satisfying the equilibrium constraint (balance of linear momentum). A goal function is expressed as

$$F = \sum_{e=1}^{m} w_e F_e(\varepsilon_e, \sigma_e), \tag{6.4}$$

with w_e being the volume of a bar element and m the total number of elements. The constrained minimisation problem can be posed as

$$\text{Minimize:} \quad \sum_{e=1}^{m} w_e F(\varepsilon_e, \sigma_e), \tag{6.5}$$

$$\text{subject to:} \quad \varepsilon_e = \sum_{i=1}^{n} B_{ei} u_i \quad \text{and} \sum_{e=1}^{m} w_e B_{ei} \sigma_e = f_i, \tag{6.6}$$

with B_{ei} being the derivatives of interpolation functions, u_i the displacement degrees of freedom with n being the number of degrees of freedom, f_i the vector of applied forces. The final minimization problem can be written as

$$\sum_{e=1}^{m} w_e F_e \left(\sum_{i=1}^{n} B_{ei}, u_i, \sigma_e \right) - \sum_{i=1}^{n} \left(\sum_{e=1}^{m} w_e B_{ei} \sigma_e - f_i \right) \eta_i = 0, \tag{6.7}$$

where η_i are the Lagrange multipliers for solving the constrained minimization problem. Minimization with respect to the displacements and stresses would then lead to two sets of equations that can then be solved with a detailed procedure outlined in [36].

The problem statement is altered in [74] to account for finite strain and the local state is defined by the 2^{nd} Piola-Kirchhoff stress tensor $\mathbf{S}$ and the Green-Lagrange strain tensor $\mathbf{E} = \frac{1}{2}(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I})$. The distance is expressed as

$$d((\mathbf{E},\mathbf{S}),(\hat{\mathbf{E}},\hat{\mathbf{S}})) = \sqrt{\int_{\Omega} ||(\mathbf{E},\mathbf{S}) - (\hat{\mathbf{E}},\hat{\mathbf{S}})||^2_{\text{loc}} d\Omega}, \tag{6.8}$$

with $||(\mathbf{A},\mathbf{B})|| = \sqrt{\mathbf{A} : \mathbb{C} : \mathbf{A} + \mathbf{B} : \mathbb{C}^{-1} : \mathbf{B}}$ where $\mathbf{A} = \mathbf{E} - \hat{\mathbf{E}}$ and $\mathbf{B} = \mathbf{S} - \hat{\mathbf{S}}$, with the hat $(\hat{\bullet})$ sign noting the local states from a dataset. The procedure is explained in detail in [74] and the basic implementation is referred to as DD (Data-Driven). Additional enhancements to the basic DD method includes enriching the dataset by further rotating the available data into discretised orbits similar to [57], this can be done if an isotropic material is assumed and this approach is referred to as DDiso (Data-Driven isotropic). This would make the approach objective, i.e. frame indifferent. Another enhancement called locally convex embedding that was proposed in [25] is implemented to augment the datasets, this is referred to as DDLC (Data-Driven Locally Convex). Combining the two enhancements renders the DDLCiso implementation (Data-Driven Locally Convex isotropic). The DDCM approach does not include physical considerations in the same manner as the NN approaches. It is built upon the idea to minimize a distance with the constraint being the balance of linear momentum, if objectivity is to be included then the dataset needs to be artificially enriched by rotating the existing available data further increasing the computational burden. For other details of the used DDCM approaches see [72]. The development of the DDCM solvers was done by Felipe Rocha and Laurent Stainier and their contribution is gratefully acknowledged.

6.1 Cook Membrane Numerical Example

The 1^{st} numerical example on which the performance of the NN and DDCM approach is compared on is the Cook membrane with the Ciarlet law, from which datasets were gathered to train the NNs and to run the data-driven solvers. This example was taken from [61]. The Ciarlet law is a compressible invariant based model with the energy expression

$$\psi_{\text{Ciarlet}}(I_1, J) = \frac{\mu}{2}(I_1 - 3) + \frac{\lambda}{4}(J^2 - 1) - \left(\frac{\lambda}{2} + \mu\right)\log J, \tag{6.9}$$

with $\mu = 185.185$ MPa and $\lambda = 432.099$ MPa. The geometry and finite element meshes are shown in Fig. 6.2. The dataset which was used by both the NNs and the DDCM implementations was obtained from solving the Cook membrane on the mesh shown in blue in Fig. 6.2. Afterwards the mesh is slightly altered to the one shown in white, otherwise the DDCM approach

would obtain 100% agreement with the reference solution.

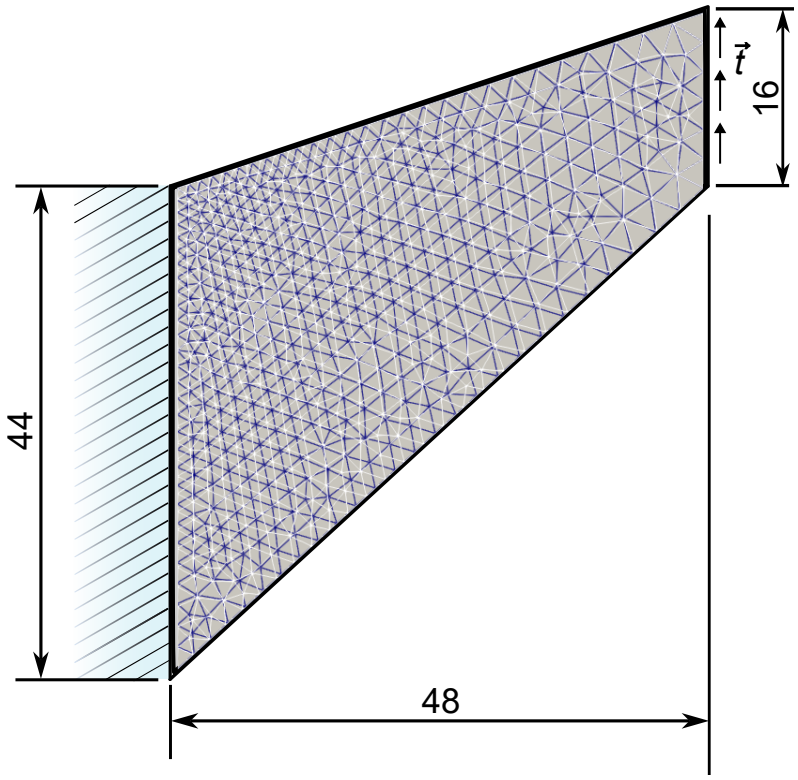
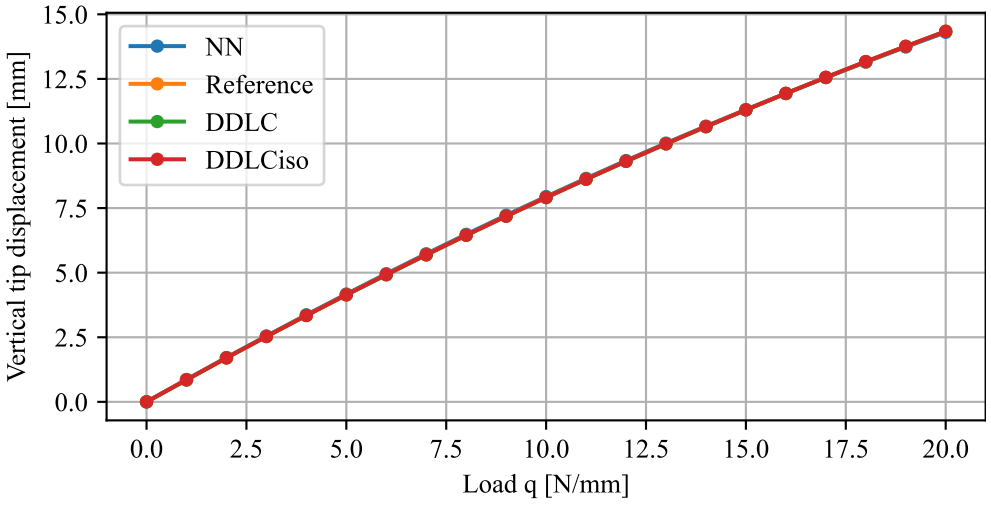
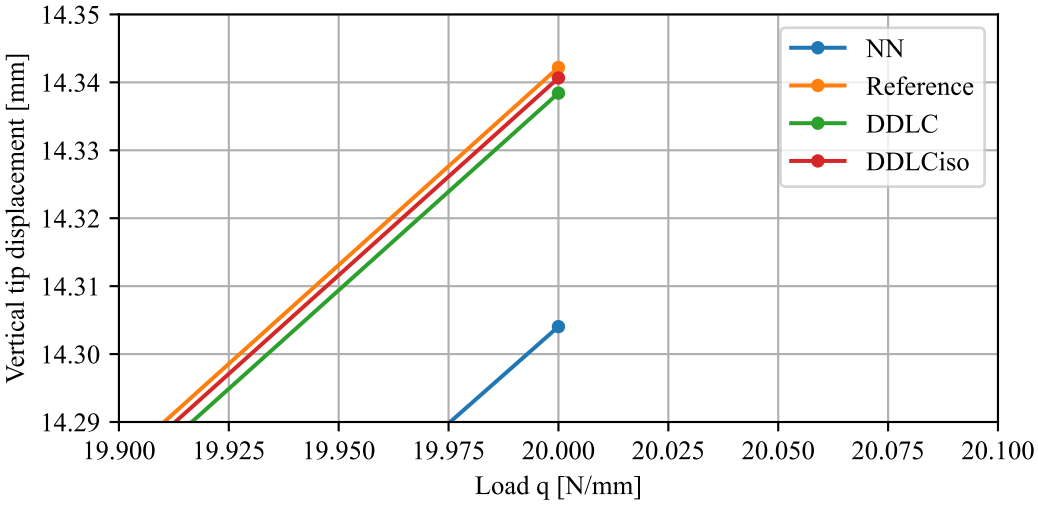


Figure 6.2: Illustration of the Cook membrane geometry and mesh used for comparing NNs and DDCM. Dimensions are given in millimetres. The source mesh from which data was gathered is shown in blue and the overlayed white mesh is the one on which the comparison was done.

The vertical displacement of the top right point (at the end of the applied traction $\boldsymbol{t}$) is shown in Fig. 6.3 for different solutions. Only the results of the DDLC and DDLCiso implementations are shown in the displacement diagrams as they are the most advanced techniques that are assumed to give the best results among the DDCM implementations. A close up is shown to better illustrate the performance between the various solutions, with the DDLCiso solution showing best agreement with an error of 0.01% while the NN solution had an error of 0.27%.



(a)



(b)

Figure 6.3: Diagram showing the displacement vs. traction load for the Cook problem. Solutions of various DDCM implementations along with the NN solution are shown.

Relative L_2 norm error values of the displacements, strain and stress over the entire domain are given in Fig. 6.4 for the DDLCiso solution and in Fig. 6.5 for the NN solution. Additional results are shown when a random normally distributed multiplicative noise is added to the underlying datasets. Also, the original dataset of 3944 data was sliced into smaller datasets to test the impact of the dataset size on DDCM and NNs. This means that for each dataset, i.e. combination of noise level and dataset size, a separate NN had to be trained, but to account for the random initialization of weights at the beginning of training 10 NNs were trained per dataset. The one with the lowest loss was chosen and the results were presented using it. The DDCM implementations simply use the available dataset without any special preparation. For nearly all the datasets except the reference dataset with no added noise, the neural networks outperformed the DDCM implementations. From Fig. 6.4 it can be seen that the approach is very sensitive to the quality of the data. More specifically when looking at the displacement

errors from Fig. 6.4a at the datasets without added noise the smallest dataset with 100 samples has an error of $1.32 \cdot 10^{-5}$ and the largest dataset an error of $1.9 \cdot 10^{-8}$, which is 3 orders of magnitude difference. Looking at the datasets with 1% noise the errors change, but remain in the same order of magnitude on the smallest dataset with 100 samples. For the largest dataset the error increases to $1.505 \cdot 10^{-3}$ which is 5 orders of magnitude worse than the original noise-free dataset and even worse than the noisy small dataset. This leads to an unexpected result where if lower quality data is available, i.e. more noise in the dataset, then smaller datasets seem to lead to better results. On the other hand the errors from Fig. 6.5 are much less varied and seem to be more or less the same in spite of the added noise with all the errors being within one order of magnitude falling in the range of $5 \cdot 10^{-5}$ to $1 \cdot 10^{-6}$ when looking at the displacement errors in Fig 6.5a. The ICNN approach thus lends models which are resilient to the worse quality of the underlying datasets with an observable trend in Fig. 6.5c where the errors for stress data are lesser with an increasing size of the dataset across all noise levels.

Relative errors comparing the accuracy of DDCM and NNs are compared in Fig. 6.6 where the values are calculated as

$$\text{error} = \frac{(\bullet)_{\text{DDCM}} - (\bullet)_{\text{NN}}}{(\bullet)_{\text{DDCM}}} \cdot 100\%, \tag{6.10}$$

so that the value describes how well the NNs perform relative to the DDCM implementation. A negative value would mean that the DDCM implementation outperforms the NNs. In nearly all the datasets and values in Fig. 6.6 the NNs outperform DDCM, the only situation where DDCM has an advantage is for larger datasets without added noise.

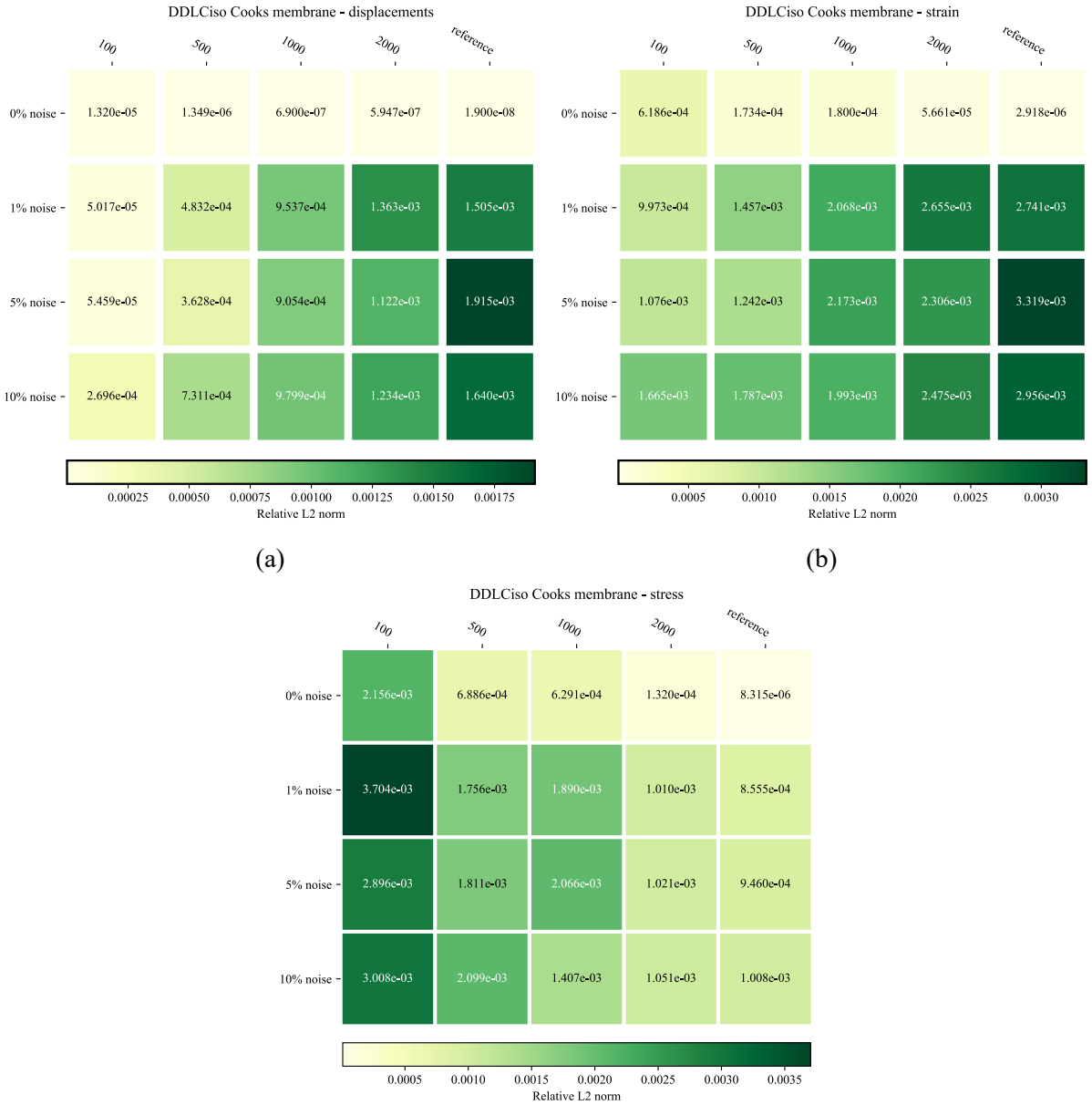


Figure 6.4: Relative L_2 norms of displacement, strain and stress errors for the DDLCiso implementation. Reference dataset contains 3944 data points.

Figure 6.5: Relative L_2 norms of displacement, strain and stress errors for the DDLCiso implementation. Reference dataset contains 3944 data points.

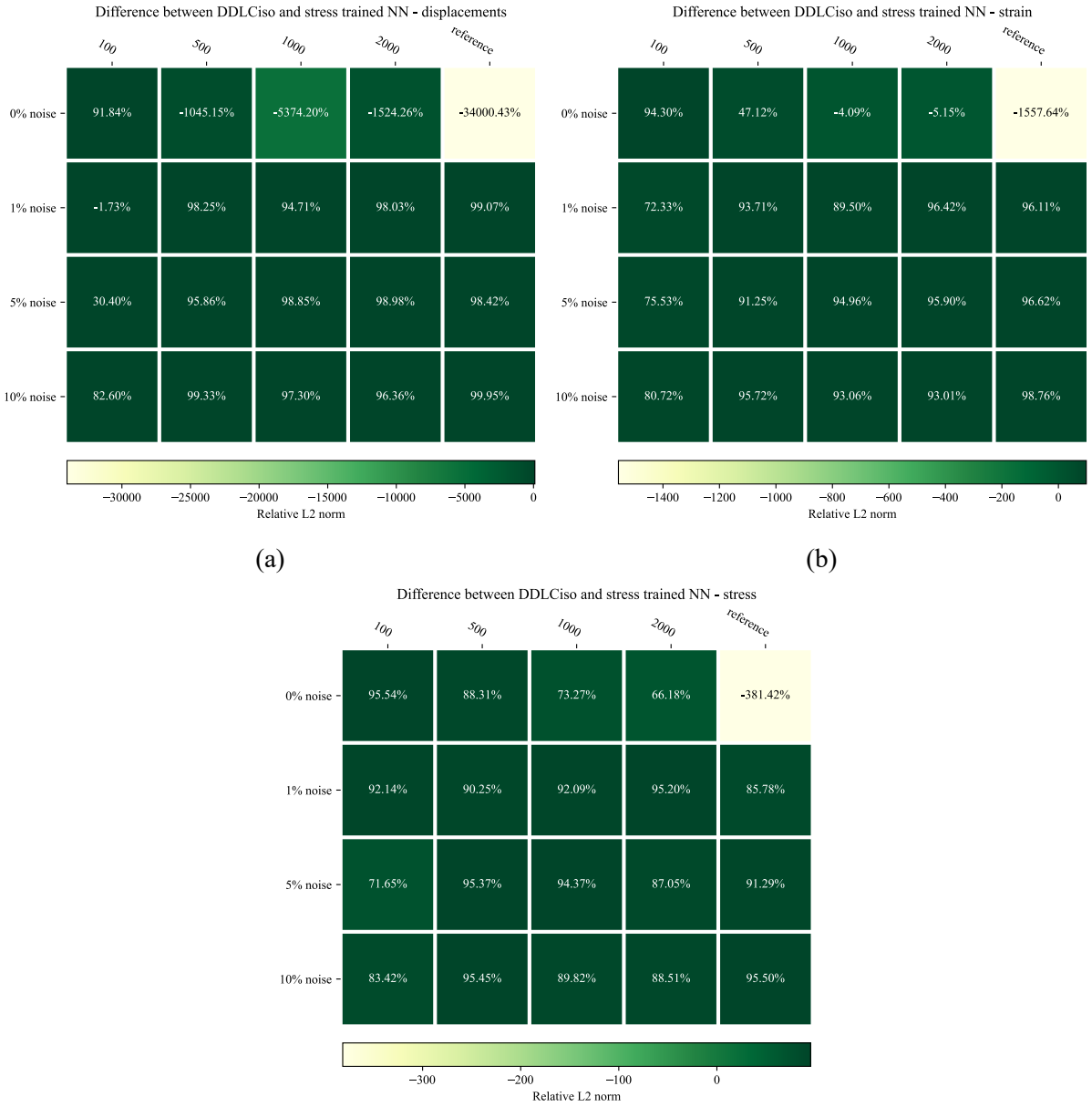


Figure 6.6: Relative errors of DDCM vs NNs for displacement, strain and stress. Reference dataset contains 3944 data points.

6.2 Punch Problem Numerical Example

The 2nd example is the punch problem that was already solved in Sec. 3.6.2. It is also taken from [72], the geometry, mesh and continuos load $q = 100$ N/mm are shown in Fig. 3.17. It is solved with the same Ciarlet model as the Cook problem before, using the same dataset gathered from the Cook model from Fig. 3.7. This is done in order to see generalisation capabilities of both the DDCM and NN approaches. The downward vertical displacement of the top left corner is shown in Fig. 6.8, and the relative errors (relative to the maximum value obtained for each respective plot) over the entire domain for the DDLCiso and NN approaches is shown in Fig. 6.9.

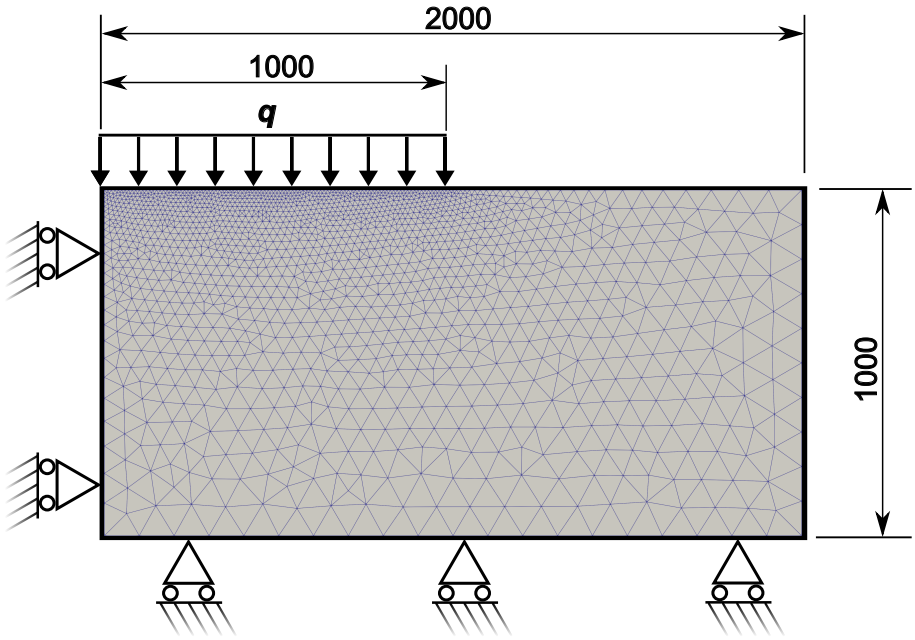
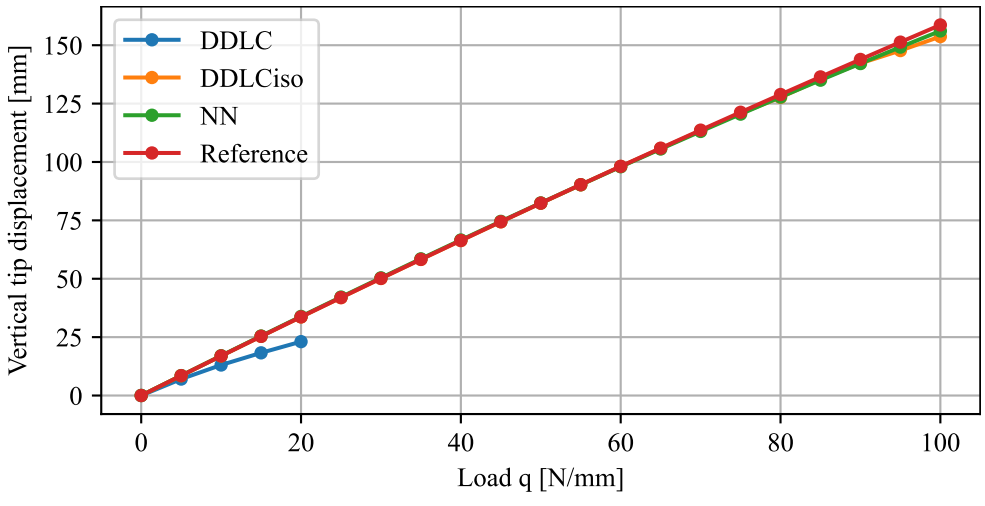
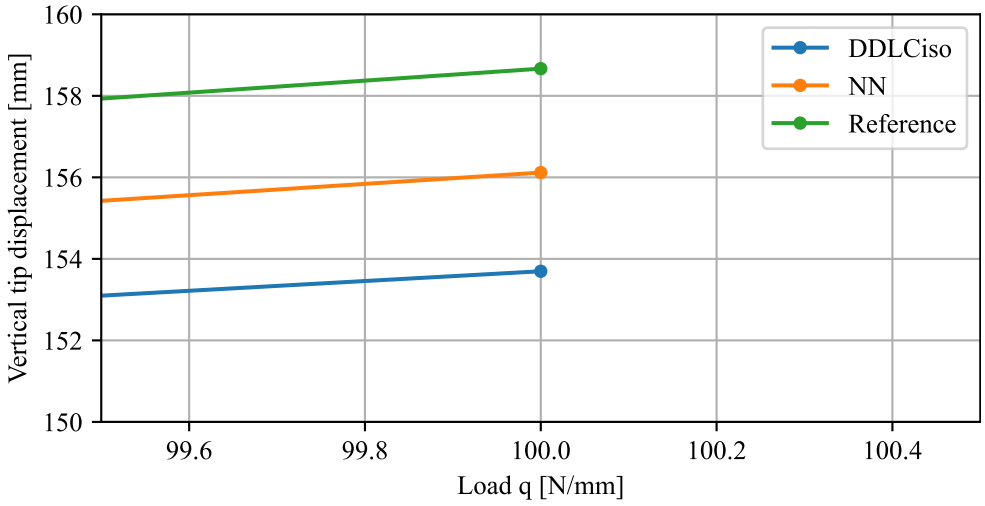


Figure 6.7: Geometry, mesh and load of the punch problem. Dimension are in millimetres.



(a)

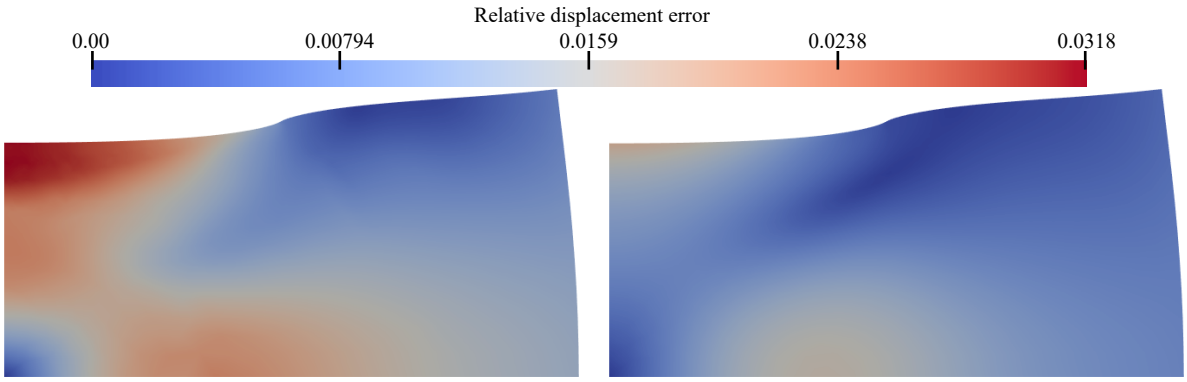


(b)

Figure 6.8: Downward vertical displacement of the top left corner of the punch problem.

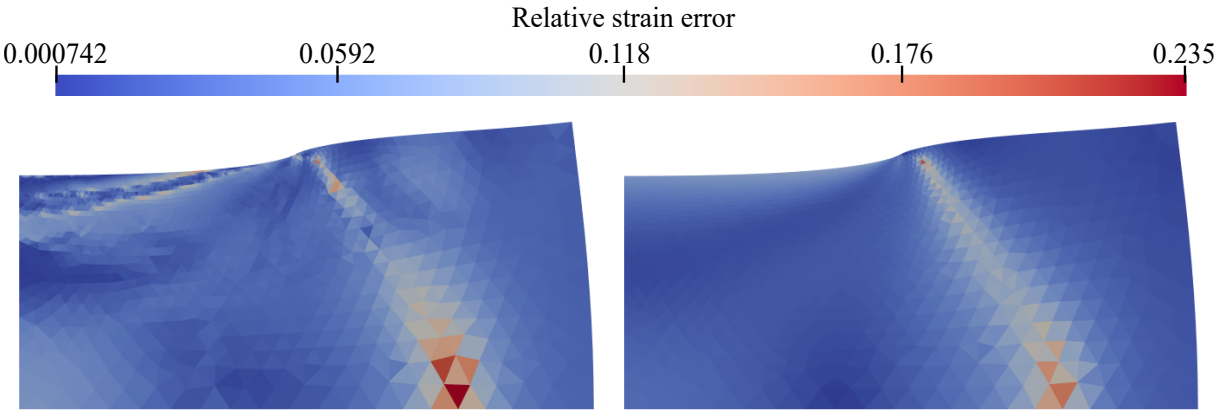
From the curve evolutions in Fig. 6.8a it is evident that the base data-driven solver without simulated isotropy is inadequate for general calculations, as shown by the divergence and failure of the DDLC implementation. The DDLCiso implementation is much more general in its application as it successfully completes the calculation with an error of 3.14%. The NN shows even better generalisation properties as it completed the calculation with an error of 1.61%. This points to the general properties and application of the NN and DDCM approaches, where the DDCM approaches are better applied to specialised cases where the behaviour is more or less known and the data-sets of the deformations that occur is rich and detailed. The application of NNs is on the other hand more general and are to be used in cases of sparse or noisy data, or when a general model is required. Another difference that further supports this notion is that the presented NN models have been trained on plane strain data but can be used in any type of analysis, whereas DDCM can only be successfully applied to the model analysis/deformation type from which the data was gathered.

Looking at the relative error plots in Fig. 6.9 the DDLCiso results are not smooth but are rather discontinuous, another property of the DDCM implementations which is indicative that the underlying dataset is insufficient for the given problem. This is reflected in the higher errors that occur in the DDLCiso solution for displacement, strain and stress.



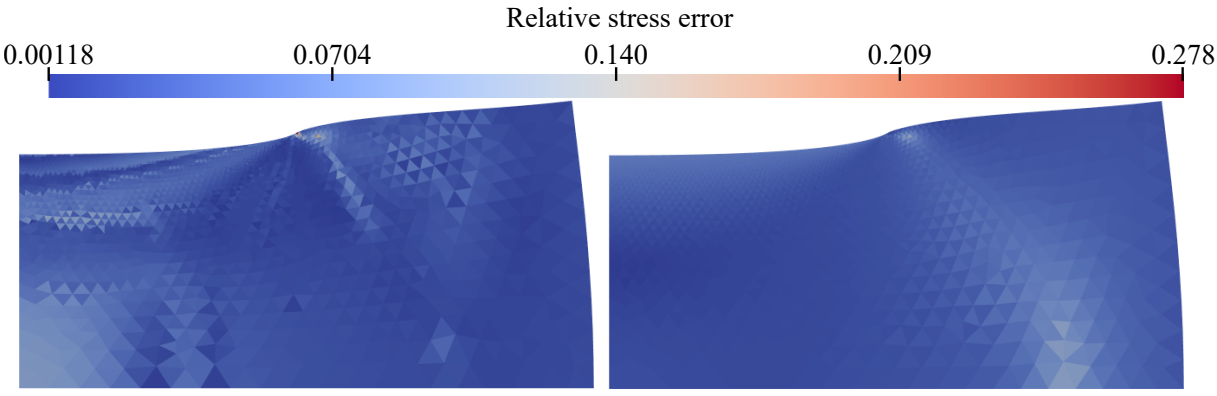
(a) Relative displacement error for DDLCiso, largest error is 3.18%.

(b) Relative displacement error for stress trained NN, largest error is 1.91%.



(c) Relative strain error for DDLCiso, largest error is 23.5%.

(d) Relative strain error for stress trained NN, largest error is 20.1%.



(e) Relative error stress for DDLCiso, largest error is 27.8%.

(f) Relative stress error for stress trained NN, largest error is 10.9%.

Figure 6.9: Relative stress errors for the DDLCiso and stress trained NN for the punch problem using the Ciarlet material model.

7 Summary and Conclusion

The thesis starts with the most general frameworks for machine learning and neural networks and carefully examines and compares the capabilities of different neural network architectures and training strategies. The first NN model was a strain-stress based NN which managed to capture the underlying isothermal hyperelastic and thermoelastic behaviour which were taken from Ogden's hyperelastic model and its thermally expanded version. However, this stress-strain based NN was limited to plane stress applications and required a large database of 1 000 000 samples to ensure reproducibility each time it was trained. A significant improvement to the model was made with the introduction of invariants of the right Cauchy-Green deformation tensor as inputs and predicting the strain energy/free energy instead of the stresses. This invariant based neural network was trained on energy and managed to capture the behaviours it was trained on with a much smaller dataset of about 30 000 samples, roughly 33 times smaller than for the stress-strain based NN. Utilising the fact that invariants are objective the invariant based NN offers the significant advantage that it can be applied to any other state other than plane stress. This was demonstrated by performing an analysis of a 3D cracked bar with mixed finite elements, while the data on which the NN was trained on was gathered from plane stress.

A further improvement was made by introducing the modified linear exponential function as an activation function. The linear exponential is used in hyperelastic material models, has convenient properties that are useful for satisfying *normalisation of the energy*, *non-negativity of the energy* and *convexity*, and by extension *polyconvexity* of the entire hyperelastic model. Additionally, the fact that using invariants of the right Cauchy-Green deformation tensor is a convenient way to calculate the stress tensor $\mathbf{S}$ (as well as $\boldsymbol{\sigma}$ or $\mathbf{P}$) the NNs can easily be trained on stresses instead of the energy directly, i.e. the NNs are trained on their derivatives. This last property enforces *thermodynamic consistency* meaning that the stresses are calculated from the energy. All of these improvements lead to the development of the LINEXP-PANN (LINear EXPonential Physics-Augmented Neural Network) model which was extensively discussed in Sec. 4, and which was successfully applied to several material models of different complexity, namely the Neo-Hookean, Mooney-Rivlin and Ogden models. The best performing model managed to capture the underlying isothermal Ogden model and needed just 15 samples obtained from three simple modes of deformation, more precisely uniaxial, equibiaxial and planar tension. Five samples were taken from each mode of deformation. Also, the LINEXP-PANN is a shallow NN since it contains only one hidden layer with five neurons for a total of 20 trainable parameters. Comparing with the first functional NN presented in this thesis, the stress-strain based NN, this presents a reduction in data size of roughly 66 000 times and a reduction in the number of trainable parameters of 1 116 times, while still providing excellent performance.

The LINEXP-PANN was further tested against its own variations and the *Constitutive Artificial Neural Network* (CANN) [46] in Sec. 5 for modelling the Mullins effect, a simple type of damage in hyperelastic materials. They were used as a model for the underlying undamaged

energy and the LINEXP-PANN which was found to be the best performing NN in Sec. 4 was also found to be the best for modelling the Mullins effect. Additionally, a novel type of training NNs for damage was presented in Sec. 5 where certain weights in the NN were reused to model both the undamaged energy and the damage evolution.

The application of NNs to hyperelasticity in this thesis was focused on incompressible behaviour, but for completeness in Sec. 6 the extension to compressibility is made using a input convex variation of the LINEXP-PANN. In the same section the performance of the NNs was compared to results obtained by other authors using an alternative approach to data-driven modelling called *Data-driven Computational Mechanics* (DDCM). The benefits of the NNs, such as the insensitivity to the quality of the underlying dataset, are shown and the overall advantage of using NNs in case of smaller or lesser quality datasets is demonstrated.

To conclude, both the hypotheses in this work were proven correct. The first hypothesis of this work has been proven correct since general neural network models were successfully applied as material models to thermoelasticity and the Mullins effect. In Sec. 3.3 a general neural network model was used to capture the thermoelastic behaviour. The same neural network model was used on both ordinary isothermal hyperelastic behaviour and on thermoelastic behaviour. Even though the differences in the two behaviours are small the same general neural network model described them and captured the small differences. In Sec. 5 the newly developed LINEXP-PANN model, which was introduced by modelling ordinary hyperelasticity, was also applied to model the Mullins effect. Further in the same section two different ways of modelling the damage evolution using the LINEXP-PANN as a basis were proposed and successfully implemented further showing the generality of neural networks as a hyperelastic material model. The second hypothesis was proven correct with the development of the LINEXP-PANN where the linear exponential custom activation function was introduced. The LINEXP-PANN model are very small shallow neural networks consisting of only 5 neurons in one hidden layer. It is able to capture ordinary hyperelastic behaviour with a considerably smaller number of parameters and training data when compared to more conventional neural networks which was commented on in Sec. 4.2.

Neural networks have proven to be flexible, accurate and practical to use. They enable engineers to streamline their workflows. Instead of spending time deciding which material model would be best, they can simply use a neural network and be confident that they will get reliable results. The material behaviours discussed in this thesis are a small sample of the various complex phenomena that can no longer be ignored in the development of modern systems that drive our industry. Neural network modelling will undoubtedly prove to be an irreplaceable tool in the modern engineer's toolbox.

References

[1] *Abaqus 6.14 Analysis User's Manual*. Dassault Systemes Simulia, Inc., 2014.

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] As'ad, F., Avery, P., and Farhat, C. A mechanics-informed artificial neural network approach in data-driven constitutive modeling. *International Journal for Numerical Methods in Engineering*, 123(12):2738–2759, Mar. 2022.

[4] Ball, J. M. Convexity conditions and existence theorems in nonlinear elasticity. *Archive for Rational Mechanics and Analysis*, 63(4):337–403, Dec. 1976.

[5] Bathe, K.-J. *Finite Element Procedures*. Klaus-Jürgen Bathe, 2014.

[6] Bulin, J., Hamaekers, J., Ariza, M., and Ortiz, M. Interatomic-potential-free, data-driven molecular dynamics. *Computer Methods in Applied Mechanics and Engineering*, 415:116224, Oct. 2023.

[7] Cam, J.-B. L. A Review of Volume Changes in Rubbers: The Effect of Stretching. *Rubber Chemistry and Technology*, 83(3):247–269, Sept. 2010.

[8] Čanađija, M. Deep learning framework for carbon nanotubes: Mechanical properties and modeling strategies. *Carbon*, 184:891–901, Oct. 2021.

[9] Čanađija, M. *Thermomechanics of Solids and Structures Physical Mechanisms, Continuum Mechanics, and Applications*. Elsevier, 2023.

[10] Capuano, G. and Rimoli, J. J. Smart finite elements: A novel machine learning application. *Computer Methods in Applied Mechanics and Engineering*, 345:363–381, Mar. 2019.

[11] Carrara, P., De Lorenzis, L., Stainier, L., and Ortiz, M. Data-driven fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 372:113390, Dec. 2020.

[12] Chadwick, P. Thermo-mechanics of rubberlike materials. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 276(1260):371–403, May 1974.

[13] Chagnon, G., Verron, E., Marckmann, G., and Gornet, L. Development of new constitutive equations for the Mullins effect in rubber using the network alteration theory. *International Journal of Solids and Structures*, 43(22–23):6817–6831, Nov. 2006.

[14] Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. Sobolev training for neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[15] de Borst, R. *Nonlinear finite element analysis of solids and structures*. Wiley, 2012.

[16] Diani, J., Fayolle, B., and Gilormini, P. A review on the Mullins effect. *European Polymer Journal*, 45(3):601–612, Mar. 2009.

[17] du Bos, M. L., Balabdaoui, F., and Heidenreich, J. N. Modeling stress-strain curves with neural networks: a scalable alternative to the return mapping algorithm. *Computational Materials Science*, 178:109629, June 2020.

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

[19] Freitag, S., Graf, W., and Kaliske, M. A material description based on recurrent neural networks for fuzzy data and its application within the finite element method. *Computers & Structures*, 124:29–37, Aug. 2013.

[20] Ghaboussi, J., Garrett, J. H., and Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of Engineering Mechanics*, 117(1):132–153, Jan. 1991.

[21] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research - Proceedings Track*, 9:249–256, Jan. 2010.

[22] Goodfellow, I., Bengio, Y., Courville, A., and Bach, F. *Deep Learning*. MIT Press, 2016.

[23] Gough, J. A description of a property of caoutchuc, or indian rubber; with some reflections on the cause of the elasticity of this substance. *Mem. Lit. Phil. Soc. Manchester*, vol. 1:288–295, 1805.

[24] Hartmann, S. and Neff, P. Polyconvexity of generalized polynomial-type hyperelastic strain energy functions for near-incompressibility. *International Journal of Solids and Structures*, 40(11):2767–2791, June 2003.

[25] He, Q. and Chen, J.-S. A physics-constrained data-driven approach based on locally convex reconstruction for noisy database. *Computer Methods in Applied Mechanics and Engineering*, 363:112791, May 2020.

[26] Holzapfel, G. and Simo, J. Entropy elasticity of isotropic rubber-like solids at finite strains. *Computer Methods in Applied Mechanics and Engineering*, 132(1-2):17–44, May 1996.

[27] Holzapfel, G. A. *Nonlinear Solid Mechanics*. Wiley, 2000.

[28] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, Jan. 1989.

[29] Huang, D., Fuhg, J. N., Weißenfels, C., and Wriggers, P. A machine learning based plasticity model using proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 365:113008, June 2020.

[30] Joshi, A., Thakolkaran, P., Zheng, Y., Escande, M., Flaschel, M., De Lorenzis, L., and Kumar, S. Bayesian-EUCLID: Discovering hyperelastic material laws with uncertainties. *Computer Methods in Applied Mechanics and Engineering*, 398:115225, Aug. 2022.

[31] Jr., C. B. M. Quasi-convexity and the lower semicontinuity of multiple integrals. *Pacific Journal of Mathematics*, 2(1):25 – 53, 1952.

[32] Jung, J., Yoon, K., and Lee, P.-S. Deep learned finite elements. *Computer Methods in Applied Mechanics and Engineering*, 372:113401, Dec. 2020.

[33] Kalina, K. A., Gebhart, P., Brummund, J., Linden, L., Sun, W., and Kästner, M. Neural network-based multiscale modeling of finite strain magneto-elasticity with relaxed convexity criteria. *Computer Methods in Applied Mechanics and Engineering*, 421:116739, Mar. 2024.

[34] Karapiperis, K., Stainier, L., Ortiz, M., and Andrade, J. Data-driven multiscale modeling in mechanics. *Journal of the Mechanics and Physics of Solids*, 147:104239, Feb. 2021.

[35] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[36] Kirchdoerfer, T. and Ortiz, M. Data-driven computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 304:81–101, June 2016.

[37] Kirchdoerfer, T. and Ortiz, M. Data driven computing with noisy material data sets. *Computer Methods in Applied Mechanics and Engineering*, 326:622–641, Nov. 2017.

[38] Klein, D. K., Fernández, M., Martin, R. J., Neff, P., and Weeger, O. Polyconvex anisotropic hyperelasticity with neural networks. *Journal of the Mechanics and Physics of Solids*, 159:104703, Feb. 2022.

[39] Klein, D. K., Roth, F. J., Valizadeh, I., and Weeger, O. Parametrized polyconvex hyperelasticity with physics-augmented neural networks. *Data-Centric Engineering*, 4, 2023.

[40] Klinkel, S., Gruttmann, F., and Wagner, W. A mixed shell formulation accounting for thickness strains and finite strain 3d material models. *International Journal for Numerical Methods in Engineering*, 74(6):945–970, Oct. 2007.

[41] Košmerl, V., Štajduhar, I., and Čanađija, M. Predicting stress–strain behavior of carbon nanotubes using neural networks. *Neural Computing and Applications*, June 2022.

[42] Lederer, J. Activation functions in artificial neural networks: A systematic overview. *ArXiv*, abs/2101.09957, 2021.

[43] Liang, L., Liu, M., Martin, C., and Sun, W. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. *Journal of The Royal Society Interface*, 15(138):20170844, Jan. 2018.

[44] Linden, L., Klein, D. K., Kalina, K. A., Brummund, J., Weeger, O., and Kästner, M. Neural networks meet hyperelasticity: A guide to enforcing physics. *Journal of the Mechanics and Physics of Solids*, 179:105363, Oct. 2023.

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

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

[47] McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, Dec. 1943.

[48] Mendizabal, A., Márquez-Neila, P., and Cotin, S. Simulation of hyperelastic materials in real-time using deep learning. *Medical Image Analysis*, 59:101569, Jan. 2020.

[49] Mullins, L. Effect of stretching on the properties of rubber. *Rubber Chemistry and Technology*, 21(2):281–300, June 1948.

[50] Ogden, R. Large deformation isotropic elasticity: on the correlation of theory and experiment for compressible rubberlike solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 328(1575):567–583, June 1972.

[51] Ogden, R. Large deformation isotropic elasticity – on the correlation of theory and experiment for incompressible rubberlike solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 326(1567):565–584, Feb. 1972.

[52] Ogden, R. Volume changes associated with the deformation of rubber-like solids. *Journal of the Mechanics and Physics of Solids*, 24(6):323–338, Dec. 1976.

[53] Ogden, R. W. On the thermoelastic modeling of rubberlike solids. *Journal of Thermal Stresses*, 15(4):533–557, Oct. 1992.

[54] Ogden, R. W. *Non-linear elastic deformations*. Dover Publications, 1997.

[55] Parisch, H. Efficient non-linear finite element shell formulation involving large strains. *Engineering Computations*, 3(2):121–128, Feb. 1986.

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

[57] Platzer, A., Leygue, A., Stainier, L., and Ortiz, M. Finite element solver for data-driven finite strain elasticity. *Computer Methods in Applied Mechanics and Engineering*, 379:113756, June 2021.

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

[59] Rosenkranz, M., Kalina, K. A., Brummund, J., Sun, W., and Kästner, M. Viscoelasticty with physics-augmented neural networks: model formulation and training methods without prescribed internal variables. *Computational Mechanics*, 74(6):1279–1301, May 2024.

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

[61] Schröder, J., Wick, T., Reese, S., Wriggers, P., Müller, R., Kollmannsberger, S., Kästner, M., Schwarz, A., Igelbüscher, M., Viebahn, N., Bayat, H. R., Wulfinghoff, S., Mang, K., Rank, E., Bog, T., D'Angella, D., Elhaddad, M., Hennig, P., Düster, A., Garhuom, W., Hubrich, S., Walloth, M., Wollner, W., Kuhn, C., and Heister, T. A selection of benchmark problems in solid mechanics and applied mathematics. *Archives of Computational Methods in Engineering*, 28(2):713–751, 2021.

[62] Shen, Y., Chandrashekhara, K., Breig, W. F., and Oliver, L. R. Neural network based constitutive model for rubber material. *Rubber Chemistry and Technology*, 77(2):257–277, May 2004.

[63] Steinmann, P., Hossain, M., and Possart, G. Hyperelastic models for rubber-like materials: consistent tangent operators and suitability for treloar's data. *Archive of Applied Mechanics*, 82(9):1183–1217, Feb. 2012.

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

[65] Treloar, L. R. G. Stress-strain data for vulcanised rubber under various types of deformation. *Transactions of the Faraday Society*, 40:59, 1944.

[66] Truesdell, Clifford; Noll, W. *The non-linear field theories of mechanics*. Springer, 2004.

[67] van Huyssteen, D. and Reddy, B. A virtual element method for isotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 367:113134, Aug. 2020.

[68] Vlassis, N. N. and Sun, W. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695, Apr. 2021.

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

[70] Wriggers, P. *Nonlinear finite element methods*. Springer, 2008.

[71] Yagawa, G. and Okuda, H. Neural networks in computational mechanics. *Archives of Computational Methods in Engineering*, 3(4):435–512, Dec. 1996.

[72] Zlatić, M., Rocha, F., Stainier, L., and Čanađija, M. Data-driven methods for computational mechanics: A fair comparison between neural networks based and model-free approaches. *Computer Methods in Applied Mechanics and Engineering*, 431:117289, Nov. 2024.

[73] Zlatić, M. and Čanađija, M. Incompressible rubber thermoelasticity: a neural network approach. *Computational Mechanics*, 71(5):895–916, Feb. 2023.

[74] Zlatić, M. and Čanađija, M. Recovering mullins damage hyperelastic behaviour with physics augmented neural networks. *Journal of the Mechanics and Physics of Solids*, 193:105839, Dec. 2024.

[75] Zopf, C. and Kaliske, M. Numerical characterisation of uncured elastomers by a neural network based approach. *Computers & Structures*, 182:504–525, Apr. 2017.

List of Figures

List of Tables

Curriculum Vitae

Martin Zlatić was born on March 18th 1996 in Rijeka. He completed his elementary and secondary education in Rijeka, and then enrolled in the undergraduate mechanical engineering study at the Faculty of Engineering at the University of Rijeka in 2014 and subsequently the graduate mechanical engineering study in 2017 with a focus on computational mechanics. He obtained his masters degree in 2019. During his studies he worked at various naval architect bureaus as a student intern, most notably at XIMAR and Royal IHC. Since 2019 he is employed at the Faculty of Engineering's Department of Engineering Mechanics as an assistant where he holds auditory exercises in the subjects of Statics and Thermomechanics. The same year he enrolled in the postgraduate doctoral study in fundamental technical sciences at the Faculty of Engineering. His scientific research focuses on the usage of neural networks as material models with application to rubber-like materials. During his postgraduate study he spent 3 months in 2023 at the École Centrale de Nantes in Nantes, France working with professor Laurent Stainier and postdoctoral fellow Felipe Rocha (now associate professor at Université Paris-Est Créteil - UPEC). He also attended courses at the International Centre for Mechanical Sciences (CISM) in Udine, Italy regarding data-driven methods and machine learning and reduced order methods in solid mechanics. He is fluent in Croatian and English, with intermediate proficiency in French and conversional proficiency in Italian.

List of Publications

- Zlatić, M. and Čanađija, M. Incompressible rubber thermoelasticity: a neural network approach. *Computational Mechanics*, 71(5):895–916, Feb. 2023.
- Čanađija, M., Košmerl, V., Zlatić, M., Vrtovšnik, D. and Munjas, N. A computational framework for nanotrusses: Input convex neural networks approach. *European Journal of Mechanics - A/Solids*, 103:105195, Jan. 2024.
- Zlatić, M., Rocha, F., Stainier, L., and Čanađija, M. Data-driven methods for computational mechanics: A fair comparison between neural networks based and model-free approaches. *Computer Methods in Applied Mechanics and Engineering*, 431:117289, Nov. 2024.
- Zlatić, M. and Čanađija, M. Recovering Mullins damage hyperelastic behaviour with physics augmented neural networks. *Journal of the Mechanics and Physics of Solids*, 193:105839, Dec. 2024.