

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Constitutive Modelling in Hyperelasticity with Neural Networks

Eduardo da Silva Carvalho

DISSERTATION



Master in Mechanical Engineering

Supervisor: João Pedro Sousa Ferreira

Second Supervisor: Marco Paulo Lages Parente

July 14, 2023

Constitutive Modelling in Hyperelasticity with Neural Networks

Eduardo da Silva Carvalho

Master in Mechanical Engineering

July 14, 2023

Abstract

When subjected to large deformations, hyperelastic materials, such as soft tissues and other biological materials, present a highly nonlinear behaviour, which often makes their constitutive modelling complex and computationally expensive. Surrogate models can replace these traditional and costly models and overcome some computational limitations by learning directly from acquired data.

To develop the surrogates, Artificial Neural Networks (ANNs) were trained from a large dataset with the deformations (inputs) and the corresponding stress and elasticity tensors (outputs), where two material models were considered: the Neo-Hookean model for an isotropic material and the Holzapfel-Gasser-Ogden model for transversely isotropic materials. Then, the weights and biases of the trained models were used to write the forward pass equations in a Fortran subroutine. By adopting this approach, it is possible to define a material of interest in a finite element (FE) software without directly expressing its constitutive equations.

The proposed method was used to create user-defined materials (UMATs) in the finite element software Abaqus and the results obtained with the proposed approach for some benchmark problems were compared with the ones obtained with conventionally defined UMATs. The results obtained in the numerical examples highlight the possibility of using a data-driven approach to describe the constitutive behaviour of hyperelastic materials instead of using a classical approach and show that creating a material in a FE software with the proposed approach can be a viable alternative, where there is no need to directly express the material's constitutive equations.

Resumo

Quando submetidos a grandes deformações, materiais hiperelásticos, como tecidos moles e outros materiais biológicos, apresentam um comportamento altamente não linear, o que muitas vezes torna a descrição do seu comportamento constitutivo numa tarefa complexa e computacionalmente cara. *Surrogate models* podem substituir os modelos tradicionais e caros e superar algumas limitações computacionais aprendendo diretamente com dados adquiridos.

Para desenvolver os *surrogate models*, redes neurais foram treinadas a partir de um grande conjunto de dados com as deformações (*inputs*) e o correspondente tensor de tensão e de elasticidade (*outputs*), onde foram considerados dois modelos de materiais: o modelo Neo-Hook para um material isotrópico e um modelo para materiais transversalmente isotrópicos. Em seguida, os pesos e vieses dos modelos treinados foram usados para escrever as equações do *forward pass* numa sub-rotina Fortran. Adotando esta abordagem, é possível definir um material de interesse num software de elementos finitos, sem a necessidade de expressar diretamente as equações constitutivas.

O método proposto foi usado para criar materiais (UMATs) no software de elementos finitos Abaqus e os resultados obtidos com a abordagem proposta para alguns problemas de referência foram comparados com os obtidos com UMATs definidas convencionalmente. Os resultados obtidos nos exemplos numéricos destacam a possibilidade de usar uma abordagem baseada em dados para descrever o comportamento constitutivo de materiais hiperelásticos em vez de usar uma abordagem clássica e mostram que a criação de um material num software FE com a abordagem proposta pode ser uma alternativa viável, onde não há necessidade de expressar diretamente as equações constitutivas do material.

Acknowledgements

Ao meu orientador, Prof. Doutor João Pedro Sousa Ferreira, agradeço todo o apoio, sugestões, compreensão e paciência demonstrada ao longo das várias etapas deste trabalho. Apesar da distância, esteve sempre disponível para ajudar e para providenciar os recursos necessários para conseguir finalizar esta tese.

Agradeço também ao meu coorientador, Prof. Doutor Marco Paulo Lages Parente, pela ajuda e suporte providenciado ao longo de todos os momentos desta dissertação. Demonstro sincera gratidão por me ter "pescado" para o mundo da Biomecânica e por ser a pessoa, que, de modo alegre, cativante e com enorme disponibilidade, me ensinou, desafiou e proporcionou oportunidades, mesmo antes do início dos trabalhos da tese.

Aos meus amigos e ao basket, por me proporcionarem excelentes momentos que me motivaram e fizeram abstrair deste trabalho e dos momentos mais complicados.

À Luísa que durante estes 6 meses esteve sempre ao meu lado para me apoiar. Pela paciência e positividade que sempre teve, dando-me forças para continuar empenhado. Obrigado por todos os momentos passados juntos que me faziam esquecer o trabalho e me permitiam voltar mais focado e motivado.

Aos meus tios e prima por sempre me apoiarem e incentivarem, e à minha avó por tudo o que fez por mim e por me deixar sempre radiante ao ver o orgulho e felicidade que ela tinha pelo neto estar a concluir o mestrado.

Ao meu irmão, pelo apoio dado, pelas conversas nas fases mais difíceis e por me tirar do trabalho para desanuviar a cabeça quando era necessário.

Aos meus pais, por todo o carinho e compreensão que demonstraram, pela paciência e ajuda nas fases mais trabalhosas e complicadas. Por me ouvirem e darem sempre os melhores conselhos, por sempre me incentivarem a continuar a trabalhar e por sempre me terem proporcionado tudo o que precisei para terminar este curso e esta dissertação de mestrado. Um obrigado não chega para agradecer tudo o que representam para mim.

Eduardo da Silva Carvalho

*“Great things come from hard work and perseverance.
No excuses.”*

Kobe Bryant

Contents

1	Introduction	1
1.1	Context	1
1.2	Motivation and Goals	5
1.3	Dissertation Outline	6
2	Machine Learning	7
2.1	Artificial Neural Networks	8
2.1.1	Activation Functions	10
2.1.2	Backward Propagation	12
2.1.3	Learning Rate	14
2.2	Hyperparameters	14
3	Continuum Mechanics	17
3.1	Kinematics	17
3.1.1	Motion	17
3.1.2	Deformation Gradient	18
3.1.3	Polar Decomposition	19
3.1.4	Stretch	20
3.1.5	Strain Measures	21
3.1.6	Directional Derivative and Lie Time Derivatives	22
3.1.7	Material and Spatial Time Derivatives	23
3.2	Stress Measures	25
3.3	Balance Principles	27
3.3.1	Conservation of Mass	27
3.3.2	Balance of Momentum	28
3.3.3	Thermodynamic Laws	29
3.4	Constitutive Equations	30
3.4.1	Objectivity Rates	31
3.4.2	Hyperelasticity	32
3.4.3	Strain Energy Functions	41
3.4.4	Elasticity Tensors	46
3.5	Finite Element Method	48
3.5.1	Principle of Virtual Work	48
3.5.2	Linearization of the Principle of Virtual Work	49
4	Hyperelastic Surrogate Modelling	52
4.1	Development Workflow	54
4.1.1	Dataset Generation	54

4.1.2	Loading the Data	59
4.1.3	Training with PyTorch Lightning	61
4.2	Neo-Hookean Model	64
4.2.1	Homogeneous Deformations	64
4.2.2	Arbitrary Deformation	66
4.3	HGO Model	69
5	Finite Element Method Integration	72
5.1	Converting Forward Pass to Fortran	72
5.2	Neo-Hook Numerical Examples	73
5.2.1	One Element Cube	73
5.2.2	Block with Pressure	91
5.2.3	Cook's Membrane	100
5.2.4	Rectangular Block - Uniaxial, Shear and Torsion	105
5.2.5	Convergency	120
5.3	HGO Numerical Examples	122
5.4	Concluding Remarks	129
6	Conclusion	130
6.1	Final Remarks	130
6.2	Future Work	131

List of Figures

1.1	Abaqus flowchart	4
2.1	Scheme of a perceptron	9
2.2	Scheme of a multi-layer perceptron	9
2.3	Activation functions.	11
2.4	Comparisons between loss functions used in regression models	13
2.5	Influence of the learning rate in the value of the loss function	14
3.1	Comparison between strain measures for a uniaxial stretch in the x direction	22
3.2	Comparison between stress measures for a uniaxial stretch in the x direction	27
4.1	ANN architecture	53
4.2	Example of random deformation gradient	57
4.3	Results for each NN trained	65
4.4	Evolution of training and validation losses - Stress Neo-Hook	67
4.5	Evolution of training and validation losses - Elasticity tensor Neo-Hook	68
4.6	Stratification	70
4.7	Evolution of training and validation losses - Stress HGO	71
5.1	Maximum principal stress - Cube with uniaxial load	74
5.2	Middle principal stress - Cube with uniaxial load	75
5.3	Minimum principal stress - Cube with uniaxial load	76
5.4	Displacement magnitude - Cube with uniaxial load	77
5.5	Error - Cube with uniaxial load	77
5.6	Maximum principal stress - Cube with biaxial load	78
5.7	Middle principal stress - Cube with biaxial load	79
5.8	Minimum principal stress - Cube with biaxial load	80
5.9	Displacement magnitude - Cube with biaxial load	81
5.10	Error - Cube with biaxial load	81
5.11	Maximum principal stress - Cube with simple shear load	82
5.12	Middle principal stress - Cube with simple shear load	83
5.13	Minimum principal stress - Cube with simple shear load	84
5.14	Displacement magnitude - Cube with simple shear load	85
5.15	Error - Cube with simple shear load	85
5.16	Maximum principal stress - Cube with random deformation	86
5.17	Middle principal stress - Cube with random deformation	87
5.18	Minimum principal stress - Cube with random deformation	88
5.19	Displacement magnitude - Cube with random deformation	89
5.20	Error - Cube with random deformation	89

5.21	Geometry and boundary conditions - Block under partial load	91
5.22	Maximum principal stress - Block under suction	92
5.23	Middle principal stress - Block under suction	93
5.24	Minimum principal stress - Block under suction	94
5.25	Displacement magnitude - Block under suction	95
5.26	Errors - Block under suction	95
5.27	Maximum principal stress - Block under pressure	96
5.28	Middle principal stress - Block under pressure	97
5.29	Minimum principal stress - Block under pressure	98
5.30	Displacement magnitude - Block under pressure	99
5.31	Errors - Block under pressure	99
5.32	Geometry and boundary conditions - Cook's membrane with 5 of thickness .	100
5.33	Mesh - Cook's membrane	101
5.34	Maximum principal stress - Cook's membrane	101
5.35	Middle principal stress - Cook's membrane	102
5.36	Minimum principal stress - Cook's membrane	103
5.37	Displacement magnitude - Cook's membrane	104
5.38	Error - Cook's membrane	104
5.39	Geometry and boundary conditions - Rectangular block with 1 of thickness	106
5.40	Maximum principal stress - Rectangular block under uniaxial load	107
5.41	Middle principal stress - Rectangular block under uniaxial load	108
5.42	Minimum principal stress - Rectangular block under uniaxial load	109
5.43	Displacement magnitude - Rectangular block under uniaxial load	110
5.44	Errors - Rectangular block under uniaxial load	110
5.45	Maximum principal stress - Rectangular block under shear load	111
5.46	Middle principal stress - Rectangular block under shear load	112
5.47	Minimum principal stress - Rectangular block under shear load	113
5.48	Displacement magnitude - Rectangular block under shear load	114
5.49	Errors - Rectangular block under shear load	114
5.50	Maximum principal stress - Rectangular block under torsional load	115
5.51	Middle principal stress - Rectangular block under torsional load	116
5.52	Minimum principal stress - Rectangular block under torsional load	117
5.53	Displacement magnitude - Rectangular block under torsional load	118
5.54	Errors - Rectangular block under torsional load	119
5.55	Maximum principal stress - Cube with uniaxial load and fibres in x-direction	122
5.56	Middle principal stress - Cube with uniaxial load and fibres in x-direction .	122
5.57	Minimum principal stress - Cube with uniaxial load and fibres in x-direction	123
5.58	Displacement magnitude - Cube with uniaxial load and fibres in x-direction	123
5.59	Errors - Cube with uniaxial load and fibres in x-direction	123
5.60	Maximum principal stress - Cube with uniaxial load and fibres in y-direction	124
5.61	Middle principal stress - Cube with uniaxial load and fibres in y-direction .	124
5.62	Minimum principal stress - Cube with uniaxial load and fibres in y-direction	124
5.63	Displacement magnitude - Cube with uniaxial load and fibres in y-direction	124
5.64	Errors - Cube with uniaxial load and fibres in y-direction	125
5.65	Maximum principal stress - Cube with uniaxial load and fibres in z-direction	125
5.66	Middle principal stress - Cube with uniaxial load and fibres in z-direction .	125
5.67	Minimum principal stress - Cube with uniaxial load and fibres in z-direction	126
5.68	Displacement magnitude - Cube with uniaxial load and fibres in z-direction	126

5.69	Errors - Cube with uniaxial load and fibres in z-direction	126
5.70	Maximum principal stress - Cube with compressive load and fibres in x-direction	127
5.71	Middle principal stress - Cube with compressive load and fibres in x-direction	127
5.72	Minimum principal stress - Cube with compressive load and fibres in x-direction	127
5.73	Displacement magnitude - Cube with compressive load and fibres in x-direction	128
5.74	Errors - Cube with compressive load and fibres in x-direction	128

List of Tables

4.1	Combinations to obtain the deformation gradients	55
4.2	Dataset split	59
4.3	NN architecture	64
4.4	Hyperparameters - Possible values	66
4.5	Hyperparameter tuning results - Stress Neo-Hook	67
4.6	Losses in last epoch - Stress Neo-Hook	67
4.7	Hyperparameter tuning results - Elasticity tensor Neo-Hook	68
4.8	Losses in last epoch - Elasticity tensor Neo-Hook	68
4.9	Material parameters HGO model	69
4.10	NN architecture - HGO	69
4.11	Losses in last epoch - Stress HGO	71
5.1	Comparison - Total number of iterations and required CPU time	121

List of Abbreviations

FE	Finite Element
SEF	Strain Energy Function
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
UMAT	User Material
ANN	Artificial Neural Network
AI	Artificial Intelligence
ReLU	Rectified Linear Function
MSE	Mean Squared Error
MAE	Mean Absolute Error
ADAM	Adaptive Moment Estimation
API	Application Programming Interface
HGO	Holzappel-Gasser-Ogden
CPU	Central Processing Unit

List of Symbols

$\Delta\epsilon$	Strain increment
σ	Cauchy stress tensor
\mathbf{K}	Material jacobian matrix
\mathbf{R}_f	Residual force
\mathbf{u}	Displacement field
\mathbf{F}	Deformation gradient
y_k	Output of perceptron k
x_i	Input i of perceptron k
w_i	Weight of input i of perceptron k
f	Activation function
b_k	Bias of perceptron k
L	Loss function
\hat{y}_i	Predicted value for loss calculation
y_i	True value for loss calculation
δ	Huber loss parameter
η	Learning rate
t	Time
Ω_0	Reference configuration
Ω	Deformed configuration
\mathbf{X}	Material coordinates
\mathbf{x}	Spatial coordinates
ϕ	Deformation mapping function
\mathbf{I}	Second order identity tensor

J	Jacobian
v	Deformed configuration volume
V	Reference configuration volume
a	Deformed configuration area
A	Reference configuration area
\mathbf{R}	Rotation tensor
\mathbf{U}	Right-Stretch tensor
\mathbf{V}	Left-Stretch tensor
\mathbf{C}	Right Cauchy-Green deformation tensor
\mathbf{B}	Left Cauchy-Green deformation tensor
\mathbf{a}_0	Unit vector in the reference configuration
$\lambda_{\mathbf{a}_0}$	Stretch vector
λ	Stretch ratio
$\{\hat{\mathbf{N}}_{1,2,3}\}$	Principal stretch directions in the reference configuration
$\{\lambda_{i=1,2,3}\}$	Principal stretches
$\{\hat{\mathbf{n}}_{1,2,3}\}$	Principal stretch directions in the deformed configuration
\mathbf{E}	Green-Lagrange strain tensor
\mathbf{e}	Eulerian-Almansi strain tensor
ϵ	Infinitesimal strain tensor
\mathbf{v}	Velocity field
\mathbf{a}	Acceleration field
\mathbf{l}	Spatial velocity gradient tensor
$\dot{\mathbf{F}}$	Time derivative of the deformation gradient
\mathbf{d}	Rate of deformation or stretch tensor
\mathbf{w}	Rate of rotation or spin tensor
$\dot{\mathbf{E}}$	Material strain rate tensor
$\dot{\mathbf{e}}$	Spatial strain rate tensor
$\dot{\mathbf{C}}$	Time derivative of the right Cauchy-Green deformation tensor
$\dot{\mathbf{B}}$	Time derivative of the left Cauchy-Green deformation tensor

\dot{J}	Time derivative of the Jacobian
\mathbf{t}	Cauchy or surface traction vector
\mathbf{f}	Force on the current surface
\mathbf{n}	Vector normal to current surface
$\boldsymbol{\tau}$	Kirchhoff stress tensor
\mathbf{T}	First Piola-Kirchhoff traction vector
\mathbf{N}	Vector normal to reference surface
\mathbf{P}	First Piola-Kirchhoff stress tensor
\mathbf{S}	Second Piola-Kirchhoff stress tensor
ρ	Mass density in the deformed configuration
ρ_0	Mass density in the reference configuration
\mathbf{L}	Linear momentum
$\dot{\mathbf{L}}$	Rate of change of the linear momentum
\mathbf{F}_r	Resultant force
\mathbf{b}	Current body force vector
\mathbf{B}_f	Reference body force vector
\mathcal{P}_{int}	Rate of internal mechanical work
Q	Rate of thermal work
\mathcal{E}	Rate of internal energy
e	Internal energy
r	Heat source per unit time and per unit of the current volume
\mathbf{q}	Cauchy heat flux per deformed surface area
s	Entropy
T	Temperature
Ψ	Helmholtz free energy
\mathbf{Q}	Rigid body rotation tensor
$\boldsymbol{\sigma}^{\text{Trues}}$	Truesdell stress rate
$\boldsymbol{\sigma}^{\text{Oldr}}$	Oldroyd stress rate
$\boldsymbol{\sigma}^{\text{GN}}$	Green-Naghdi stress rate

$\boldsymbol{\sigma}^{Jau}$	Jaumann stress rate
$I_{1,2,3}$	Invariants of the Cauchy-Green deformation tensors
p	Lagrange multiplier
\mathbf{F}_{iso}	Modified deformation gradient
\mathbf{F}_v	Volumetric part of the deformation gradient
\mathbf{C}_{iso}	Modified right Cauchy-Green deformation tensor
\mathbf{C}_v	Volumetric part of the right Cauchy-Green deformation tensor
Ψ_{iso}	Isochoric part of the strain-energy function
Ψ_v	Volumetric part of the strain-energy function
\mathbb{P}	Fourth-order projection tensor in the reference configuration
\mathbb{I}	Fourth-order identity tensor
\mathbf{S}_{iso}	Isochoric part of the second Piola-Kirchhoff stress tensor
\mathbf{S}_v	Volumetric part of the second Piola-Kirchhoff stress tensor
$\bar{\mathbf{S}}$	Fictitious second Piola-Kirchhoff stress tensor
$\bar{\boldsymbol{\sigma}}_{iso}$	Isochoric part of the Cauchy stress tensor
$\bar{\boldsymbol{\sigma}}_v$	Volumetric part of the Cauchy stress tensor
\mathbf{B}_{iso}	Isochoric part of the left Cauchy-Green deformation tensor
\mathbb{P}	Fourth-order projection tensor in the current configuration
$\bar{\boldsymbol{\sigma}}$	Fictitious Cauchy stress tensor
$\bar{I}_{1,2,3}$	Invariants of the isochoric part of the Cauchy-Green deformation tensors
\mathbf{a}	Unit vector in the deformed configuration
$I_{4,5}$	Pseudo-invariants from anisotropy
q	Lagrange multiplier
\mathbf{a}_0'	Unit vector of the second family of fibres in the undeformed configuration
$I_{6,7}$	Pseudo-invariants from anisotropy of the second family of fibres
$I_{8,9}$	Invariants to couple the two family of fibres
D_1	Penalty parameter to model incompressibility
C_{10}, C_{01}	Material parameters - Mooney Model
μ_p, α_p	Material parameters - Ogden Model

μ	Shear modulus
n	Chain density per unit of volume
k	Boltzmann constant
C_{10}	Material parameters - Neo-Hookean Model
Ψ_{fib}	Anisotropic contribution of the fibres to the SEF
Ψ_{mat}	Isotropic contribution of the ground matrix to the SEF
c, b, A, a	Material parameters - Humphrey and Yin Model
Ψ_{fib}^{PE}	Passive component of the fibre contribution to the SEF
Ψ_{fib}^{SE}	Active component of the fibre contribution to the SEF
f_{PE}	Passive muscle function
f_{SE}	Active muscle function
$\bar{\lambda}_f$	Isochoric fibre stretch ratio in the direction of the undeformed fibre
ζ^{CE}	Parameter related to the strain of the contractile element of the fibres
T_0^M	Muscle peak stress
F_0^M	Peak isometric muscle force
A_0	Physiological cross-section area of the fibres
λ^M	Muscle stretch
k_1, k_2	Material parameters - HGO model
\mathbf{H}	Symmetric generalized structure tensor
\mathbf{M}	Arbitrary unit vector located in the three dimensional eulerian space
Θ, ψ	Eulerian angles to define the fibre dispersion
ρ_f	Orientation distribution density function
κ	Radial dispersion parameter
\bar{E}_i	Invariant of \mathbf{C} and \mathbf{H}
\mathbb{C}	Lagrangian or material elasticity tensor
\mathbb{c}	Eulerian or spatial elasticity tensor
\mathbb{C}_v	Volumetric part of the material elasticity tensor
\mathbb{C}_{iso}	Isochoric part of the material elasticity tensor
\tilde{p}	Scalar function

$\bar{\mathbb{C}}$	Fictitious material elasticity tensor
$\tilde{\mathbb{P}}$	Modified fourth order projection tensor
$\bar{\mathbf{u}}$	Prescribed displacement
$\partial\Omega_u, \partial\Omega_t$	Surfaces of the body in the current configuration
$\bar{\mathbf{t}}$	Prescribed Cauchy traction vector
$\delta\mathbf{u}$	Virtual displacement field
δW_{ext}	Virtual external work
δW_{int}	Virtual internal work
$\bar{\mathbf{T}}$	Prescribed first Piola-Kirchhoff traction vector

List of Operators

∇	Gradient
\det	Determinant
\cdot	Scalar product
\otimes	Dyadic or tensor product
$D_{\mathbf{v}}\Phi$	Directional derivative of Φ in the direction of \mathbf{v}
\mathcal{L}	Lie time derivative
$:$	Double dot product or contraction
div	Divergence
tr	Trace

Chapter 1

Introduction

1.1 Context

Bioengineering applies concepts and knowledge from multiple engineering and medical sciences to solve healthcare-related problems, improving patient care, diagnosis, and treatment. It is also responsible for the design, development, and improvement of medical devices, equipment, and technologies. There are many subfields within biomedical engineering such as medical imaging, medical instrumentation, biomaterials, prosthetics, tissue engineering and biomechanics [1]. From these subfields, biomechanics is of particular relevance and has attracted more interest over the past years.

Biomechanics plays a crucial role in many different areas, such as rehabilitation, ergonomics, orthopaedics, prosthetics and sports performance. To better define it, it is first important to look at the definition of biology and mechanics. Biology studies living things, and mechanics studies motions and the applied loads that cause them. Thus, Biomechanics can be defined as the study of the motions experienced by living things in response to applied loads. More precisely, it can be viewed as the development, extension, and application of mechanics to better understand how different loading conditions influence the structure, properties, and function of living things [2].

Biomechanical phenomena are analysed with engineering and biophysical principles. Such analyses have applications in several areas (pathophysiology [3–5], dentistry [6–8], orthopaedics [9–11]) and usually require simulations with computational approaches, such as the finite element (FE) method [12]. With these computational capabilities, it is possible to model and simulate biological components and analyse the interaction between them and the effects of loading conditions on stresses and strains. The simulations can be handled numerically, but they are complex and involve nonlinear multi-physics equations, making them computationally expensive [13].

To solve these analyses, it is necessary to have the domain of interest properly defined (geometry), to get the constitutive equations that characterize how the material behaves under certain conditions and to define the applied loads and boundary conditions. From

these three stages required, identifying a proper and robust constitutive model is probably the most challenging one. The materials involved in biomechanics simulations are usually hyperelastic materials, such as soft tissues and other biological materials and they are extremely nonlinear and subjected to large deformations, making their constitutive modelling expensive in terms of time and computational resources [12, 14]. Standardly, to obtain such a constitutive model, the following procedure is used [2]:

- Delineate general characteristic behaviours;
- Establish an appropriate theoretical framework;
- Identify specific functional forms of the constitutive relation;
- Calculate the values of the material parameters;
- Evaluate the predictive capability of the final constitutive relation;

This approach requires a high level of expertise, particularly in the first three steps, and was the golden standard for many years. Hyperelasticity theory was used to model the mechanical behaviour of soft biological tissues, and many constitutive models (some of them are introduced in Chapter 3) were proposed in the literature and have been widely used in many applications. These models consider a strain energy function (SEF) of a specific functional form constructed a priori and their material parameters are adjusted to describe the constitutive behaviours of different subjects, without the need to derive new constitutive equations [14].

These SEFs can be derived from phenomenological or micro-mechanically inspired mathematical expressions. Phenomenological models try to capture the appropriate functional forms of stress-strain curves observed in experiments, whereas micro-mechanically motivated models are based on physical and geometrical phenomena at the microstructural level of the material. Additionally, SEFs can also be classified based on their arguments. The simplest ones are based on the principal invariants, such as the Neo-Hookean model proposed by Treloar [15].

Recently, Machine Learning (ML) techniques are emerging as a promising solution and alternative for many problems and leading to a breakthrough in many fields [16–24], including a lot of works in biomechanics [25–29]. In addition, Deep learning (DL), which is a subset of ML, has also been used to estimate material properties [30–32] and to establish data-driven constitutive modelling as an alternative to the conventional approach.

Data-driven constitutive modelling might overcome some challenges posed by the classical approaches by providing a flexible form that can be determined directly from experimental data. One of the first works that employed data-driven constitutive modelling was the ones from Ghaboussi et al. [33] and Wu et al. [34], where a neural network (NN) was trained with experimental data to obtain stresses from strains. The material model used in

the FE method updates the stresses based on the current stress–strain state and strain increment and calculates the material stiffness matrix for the constitutive relation. The work from [Ghaboussi et al.](#) did not accomplish the second function, which led Hashash et al. [35] to propose an explicit formulation of the material stiffness matrix for NN constitutive material models. Afterwards, many authors proposed different data-driven constitutive models for hyperelastic materials [36–40] and even extended them to anisotropic hyperelastic materials [41–43].

The emerging field of data-driven constitutive modelling can be divided into two major branches. One uses geometric information about the microstructure of a material and creates representative volume elements that exhibit a behaviour similar to the macroscopic behaviour of the material. This branch of data-driven constitutive modelling can describe the mechanical behaviour of a wide range of materials, but it is computationally expensive [36]. The other one is focused on the relationship between stress and strain and often uses ML tools to build surrogates, which are fast and straightforward models that approximate the original model input-output relation. They use data to learn and replace expensive numerical models, reducing the computational effort. They usually come with a trade-off between accuracy and computation effort, but they still are becoming an interesting option in many engineering fields, particularly when used in combination with FE models [44]. In the field of biomechanics, complex FE models have already been replaced by ML-based surrogate models to predict stress distribution [28, 45] and to predict the biomechanical behaviour of organs [46, 47].

This work falls into the second branch, and a hybrid approach between purely data-driven and expert modelling is followed. The models already developed by experts obey physical-based principles relevant to soft tissues and include knowledge about mechanical behaviour and underlying microstructure properties. Stress-strain data is generated assuming different material models proposed in the literature and is then used to train a ML model that outputs stresses and the material stiffness matrix from a given deformation gradient. More specifically, fully connected NNs are trained for two material models: the Neo-Hookean model and a model for transversely isotropic materials. Afterwards, the surrogate model is used in Abaqus, which provides capabilities to model the problem, manage, monitor and visualize the results obtained, and the FE method is employed. Figure 1.1 shows a flow of data and actions from the start of the Abaqus analysis to the end of the step, with detail in the way the element stiffness is calculated. The convention used was that for decision points or specific states, a rounded rectangular box was used, whereas, for actions taken during the analysis, a rectangular box was used.

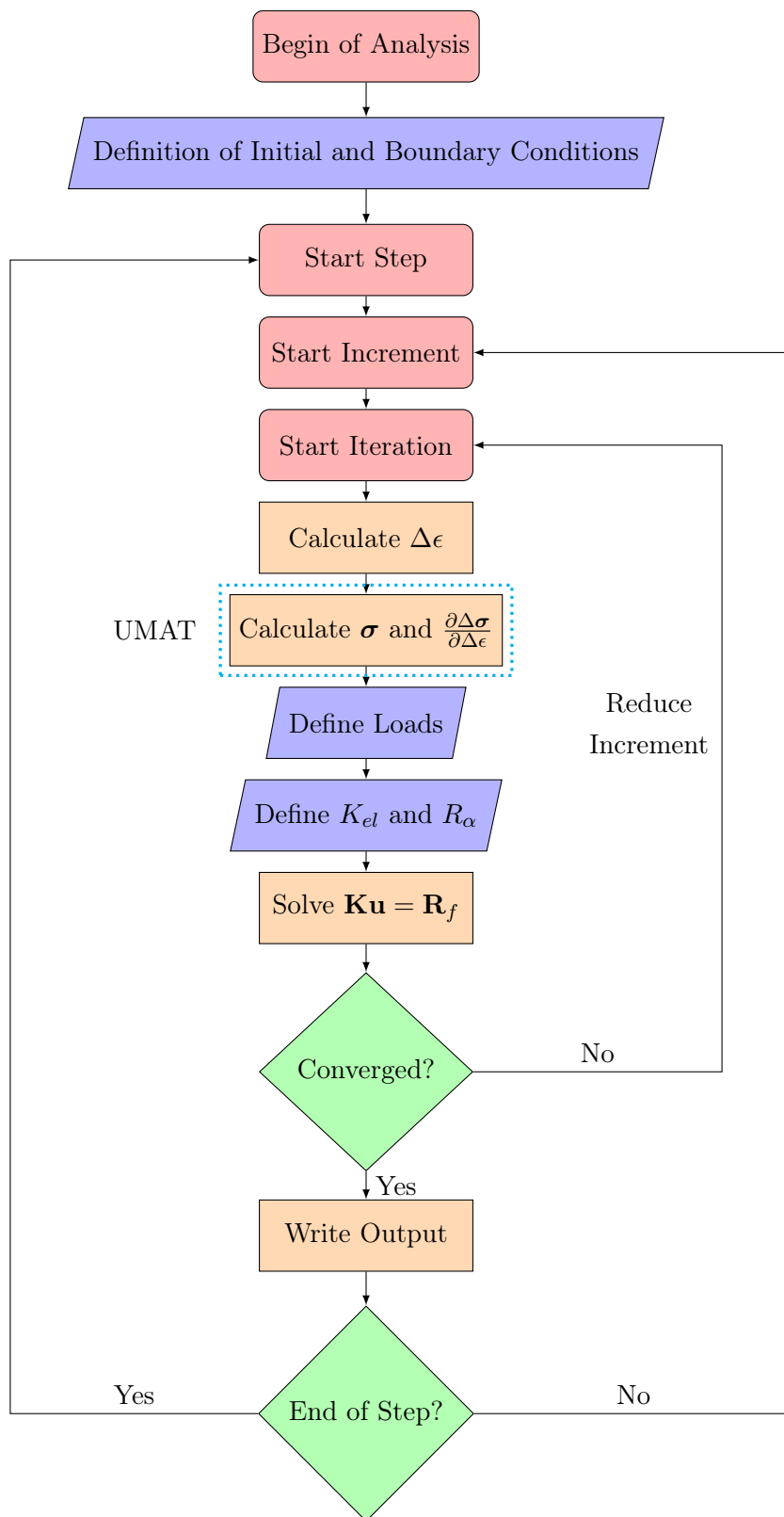


Figure 1.1: Abaqus flowchart

The stresses $\boldsymbol{\sigma}$ and the element stiffness matrix \mathbf{K} are calculated in a user-defined material (UMAT) subroutine, where the constitutive behaviour of the material is defined. Abaqus has some of the most common materials implemented, but for complex problems and in the field of bioengineering, it is often necessary to develop a UMAT with the constitutive equations that model the material in analysis. The UMAT subroutine is called at each integration and it takes as input the deformation gradient matrix \mathbf{F} and the user must compute the Cauchy stress tensor $\boldsymbol{\sigma}$ (named STRESS in the software) and a consistent tangent matrix \mathbf{K} (named DDSDDDE).

Usually, building a UMAT subroutine is a complicated task that requires the formulation of mathematical equations that model the desired material, but with a surrogate model, the task is easier. A trained ML model can be embedded in UMAT subroutine to calculate the isochoric parts of the stress and the material stiffness matrix, which eliminates the need to express analytically the constitutive equations that describe the behaviour of the material in analysis.

1.2 Motivation and Goals

This work is intended to explore the potential advantages of data-driven constitutive modelling of hyperelastic materials as an alternative to the conventional way to model their mechanical behaviour, based on phenomenological or physically based models proposed in the literature.

Expressing the constitutive equations in a Fortran subroutine is required to define a UMAT in Abaqus, which can be a complicated task because it is necessary to calculate many partial derivatives and Fortran does not have a built-in function for differentiation. This was the difficulty that motivated this dissertation, where the main goal is to develop surrogate models based on ML algorithms to describe the mechanical behaviour of hyperelastic materials and to use them in Abaqus to employ the FE method. To accomplish this fundamental goal, the following smaller objectives were defined:

1. Design and train ML models for a Neo-Hookean material to predict the stresses and the material stiffness matrix;
2. Hyperparameter tuning to improve the accuracy of the prediction;
3. Use a more complex model for the SEF that can deal with transversely isotropic materials;
4. Integrate the trained surrogate models in a UMAT to employ the FE method in Abaqus;
5. Validate this framework with the comparison of the results obtained with this approach with the results obtained with a conventional UMAT for some numerical examples.

1.3 Dissertation Outline

Chapter 2 of this dissertation gives an overview of ML. The different categories of ML are explained, with a focus on DL, mainly Artificial Neural Networks (ANNs). To do so, the principles behind such algorithms are explained, with special detail in the activations functions, the loss functions and the optimization processes used in backpropagation.

Chapter 3 gives an overview of the principles of Continuum Mechanics and of the FE method, which are fundamental for computational simulations in the bioengineering field. Firstly, the kinematics, motion and deformation are introduced. Then, the most important stress tensors are exposed, followed by the conservation principles and the constitutive modelling of hyperelastic materials. Finally, the FE method is briefly addressed.

Chapter 4 explains the process and the reasoning applied to obtain the surrogate models. The first stage was to generate the data required to feed the ML algorithms, and then the ML models were designed and trained after a hyperparameter tuning.

Chapter 5 explains the integration of the trained surrogate models in Abaqus and presents the results obtained for some numerical examples, where the results obtained with a ML-based UMAT and a conventional UMAT are compared. Chapter 6 includes the final remarks and suggestions for future works.

Chapter 2

Machine Learning

Artificial Intelligence (AI) is the ability of computer systems to perform tasks commonly associated with human intelligence and it is divided into two main categories depending on the "level of intelligence": weak or narrow AI and strong or general AI. Weak AI is designed to perform specific tasks and solve individual problems with a satisfactory level of accuracy. ML is a subfield of AI that falls into this category, which relies on advanced algorithms and models to improve performance automatically and autonomously in certain tasks, without the need for explicit programming. It requires data and iterative processing to get experience and to identify and learn patterns that enable the algorithm to make decisions and predictions [48, 49].

The first ML algorithms date back to the 1950s, when Alan Turing proposed a test for machine intelligence. This was the foundation for most of the ideas and techniques behind modern ML. Afterwards, Arthur Samuel developed a program to play checkers, often considered the first ML algorithm. Since then, the interest in ML has grown rapidly, with a substantial increase in the publications about the topic [48], and the recent advances in AI and computing power have enabled the application of ML to a variety of fields [49].

ML approaches can be divided into several classes and categories, depending on the learning method and the purpose of the algorithm [48, 49]:

Supervised learning The goal of these algorithms is to learn a function that maps an input to an output based on labelled training data. It is a task-driven approach often used for classification and regression problems. A good performance can be achieved, but it requires a large amount of labelled data, with quality and representative of the diversity present in the operational environment.

Unsupervised learning These algorithms receive unlabelled data and try to find its structure and patterns. This means that in this category, the algorithms are trained without a specific output in mind, which makes them an interesting choice for clustering.

Semi-supervised learning It uses a combination of labelled and unlabelled data. It is a mixture of the two previous categories and it is useful for real-world scenarios, where labelled data may be rare and unlabelled data is widely available.

Reinforcement learning It has an environment-driven approach. They evaluate the optimal behaviour in a particular environment to improve efficiency and are trained to make decisions based on positive and negative feedback, given in the form of rewards for good decisions and penalties for poor ones.

Deep learning These algorithms are inspired by the information-processing patterns found in the human brain. They learn hidden properties and relationships in data and reduce the need for feature extraction and preliminary data processing. However, these advantages only appear if there are large amounts of training data and the NN architecture is correctly chosen. In this category, there are three main types: feed-forward NNs, recurrent NNs and convolutional NNs. This category is going to be the focus of this chapter since the models trained in this work belong to this category.

2.1 Artificial Neural Networks

To understand the logic behind ANNs, it is important to first introduce the concept of perceptrons. They were developed in the 1950s by Frank Rosenblatt and consist of a node that receives input values. These inputs are multiplied by a corresponding weight value, which is a number between zero and one that expresses the importance of the respective input [50]. Afterwards, the weighted inputs are summed, giving the output value of the node, which is passed as input to the next layer. Thus, the output y_k of a perceptron k , as the one of Figure 2.1, with n inputs can be expressed as

$$y_k = \sum_{i=1}^n x_i w_i \quad (2.1)$$

where x_i is the i th input and w_i is the corresponding weight.

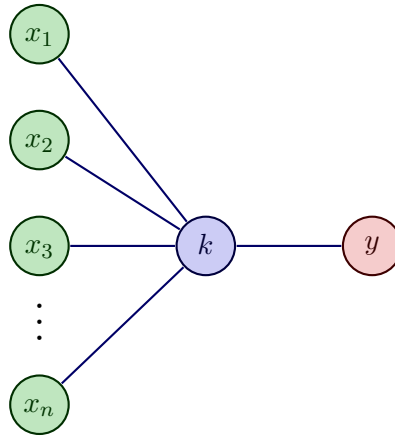


Figure 2.1: Scheme of a perceptron

A single perceptron can only solve linear problems, but if multiple layers of many perceptrons are used, more complex problems can be solved. The perceptrons, also known as neurons, process and transmit information and are usually grouped and assembled into layers called hidden layers. The simplest ANN has only one hidden layer, connecting the input layer to the output layer. If there are two or more hidden layers, the algorithm falls into the category of DL. The information from the inputs is passed to the first hidden layer after a series of mathematical computations. This process continues until the last hidden layer produces the output of the model.

In multilayer perceptron, the inputs are fed into all neurons of the first hidden layer and the corresponding outputs from this layer serve as input of the following hidden layer. This process continues until the last hidden layer is reached, where its output corresponds to the outputs of the model. A scheme of a general NN can be seen in Figure 2.2.

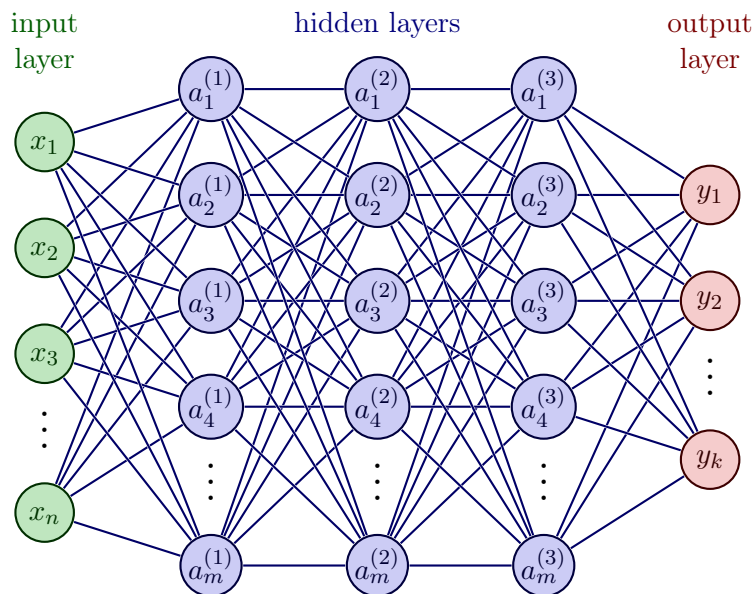


Figure 2.2: Scheme of a multi-layer perceptron

2.1.1 Activation Functions

Another way to solve more complex problems with non-linear relationships between inputs and outputs is to use activation functions. They introduce non-linearity into the network, and many of them can be found in the literature. Some of the most used ones are [50]:

Sigmoid This function converts input values into a value between 0 and 1, where most of the outputs from this activation function are going to be close to the extremes. A plot with the graphical representation of the sigmoid activation function can be found in Figure 2.3a, where it is notorious that it is S-shaped.

Hyperbolic Tangent This function is given by the hyperbolic tangent (\tanh), defined by the ratio between the hyperbolic sine and the hyperbolic cosine, and it maps the input to a value between -1 and 1. In comparison with the sigmoid function, this activation function has a steeper gradient around zero and it can give negative values as shown in Figure 2.3b.

Rectified Linear Function (ReLU) It has a linear behaviour if the input is greater than zero. However, the inputs with negative values are mapped to zero. This activation function is one of the most widely used in many applications, particularly for the hidden layers. Its graphical representation can be found in Figure 2.3c.

Leaky ReLU This function is similar to the ReLU but prevents one of its main problems, which is the so-called "dying ReLU". The previous activation function has a null derivative when the input is below 0, meaning that in backpropagation, which is going to be explained afterwards, numerous nodes can have a negative input and, consequently, do not contribute to the final output. To tackle this problem, the Leaky ReLU adds a slope to the ReLU function in the negative domain, as it is possible to see in Figure 2.3d.

Pass-Through This is a linear activation function since it keeps the input of the neuron unaffected and they can be used, for example, in the output layer of regression NNs.

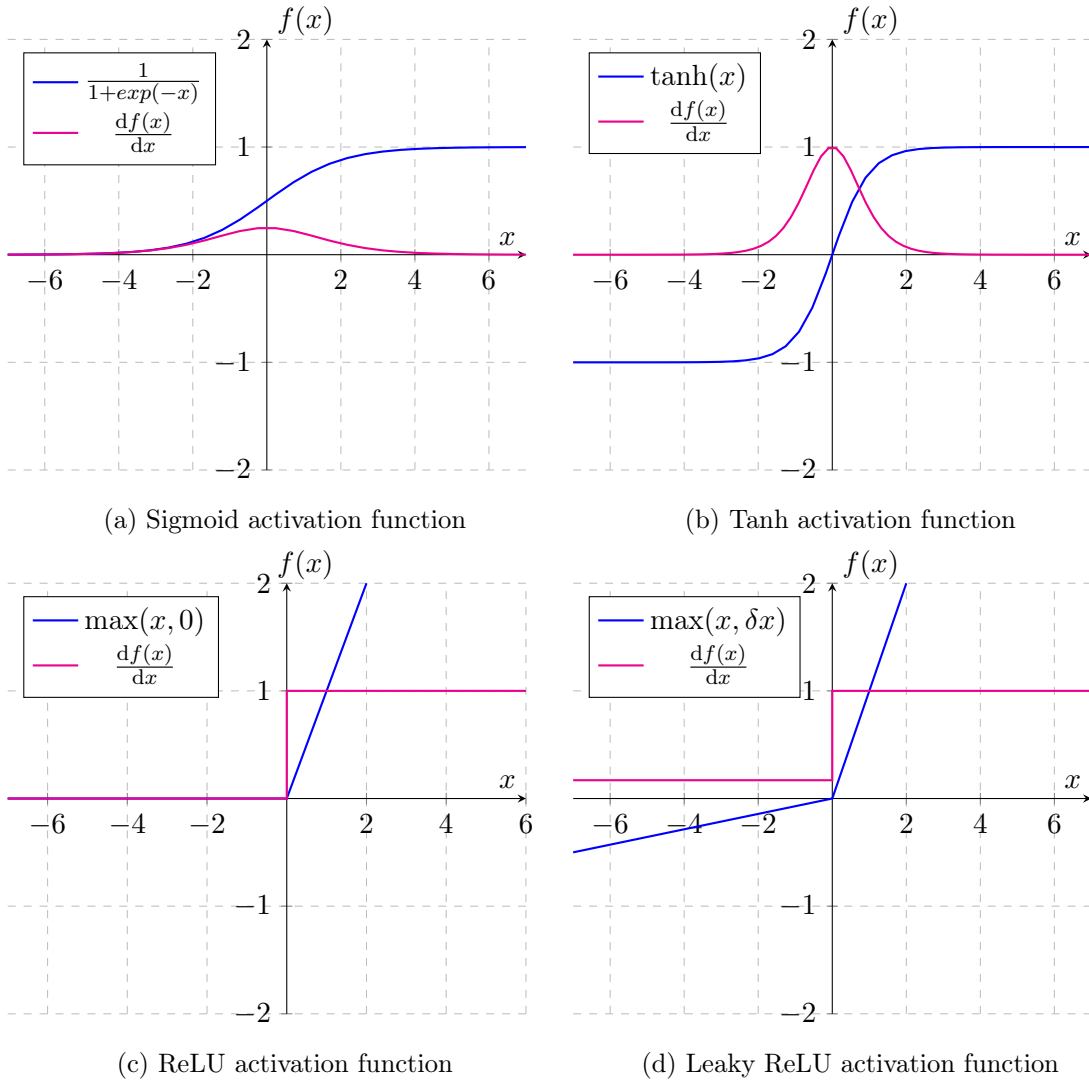


Figure 2.3: Activation functions

The training of a NN can be divided into two processes: forward propagation and backward propagation. The former is the process that goes through the architecture of the network from left to right to reach a certain output, and the latter goes from right to left and is essential for DL algorithms. Some specifications about forward propagation were already presented, but there is still an important concept to be introduced. To facilitate the training process, reduce the impact of noisy data and prevent problems such as overfitting, it is common to use biases, which are numerical values that shift the activation function of each perception [50, 51]. Thus, the fundamental equation for the forward propagation of a perceptron k is given by

$$y_k = f \left(\sum_{i=1}^m x_i w_i + b_k \right) \quad (2.2)$$

where b is the bias and f is an activation function.

2.1.2 Backward Propagation

If there was only forward propagation, the prediction obtained would be far from the desired output because the values of the weights and biases were arbitrarily chosen. This is why there is backward propagation, which is based on the calculation of a loss function L that estimates the error between the prediction of the model and the true/desirable value. As is with activation functions, there are also many loss functions proposed in the literature. Some of the most relevant ones are listed below.

Mean Squared Error (MSE) It measures the average squared difference between the predicted output value from the model and the true or desired one. This loss function is often used in regression models, is sensitive to outliers and can be calculated as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.3)$$

where N is the number of samples, \hat{y}_i is the predicted value and y_i the true value.

Mean Absolute Error (MAE) It is the average absolute difference between the predicted and desired values and contrarily to the MSE, MAE is not sensitive to outliers, which means that this loss function can be useful when dealing with unclean data since it does not give extra importance to large errors. The formulation is very similar to the one from the MSE and it is given by

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|. \quad (2.4)$$

Huber Loss This loss function is in between the previous ones, combines the benefits of each one and is commonly applied in regression problems. It uses the MAE if the absolute error is higher than a certain threshold and it uses MSE otherwise. As a consequence, a balanced evaluation criterion is obtained, where the outliers are not disregarded, but they also do not disturb the model completely. This loss is given by

$$L_i = \frac{1}{N} \sum_{i=1}^N \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta, \\ \delta (|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{if } |y_i - \hat{y}_i| > \delta \end{cases} \quad (2.5)$$

where δ is a parameter that can be adjusted according to the application's need.

In Figure 2.4 it is shown a comparison of the different loss functions introduced.

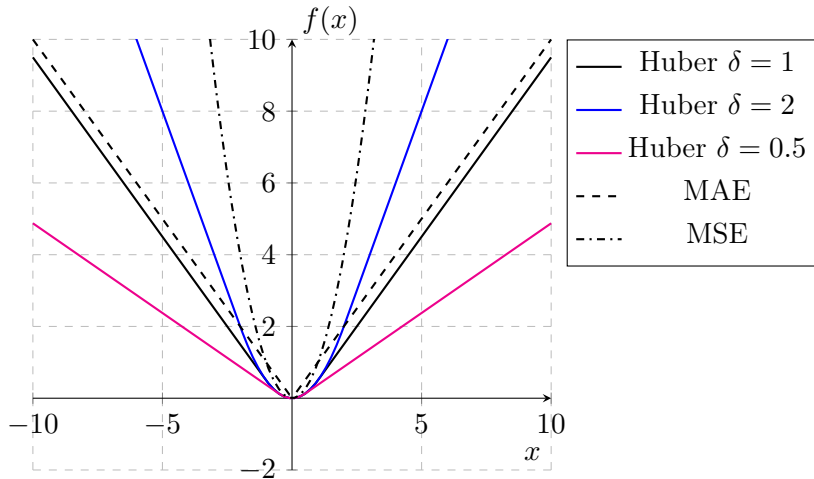


Figure 2.4: Comparisons between loss functions used in regression models

The loss function chosen is always dependent on the weights and biases of each neuron of every layer, which means that the loss function can be viewed as a surface in a multidimensional space where the axes are those weights and biases. The main goal in backpropagation is to reach a global minimum, where the loss function has the lowest possible value. However, the loss function is not a smooth surface most of the time, which introduces the risk of getting a local minimum during the process of minimization, often called optimization.

Different algorithms can be used for the optimization process, with some of the most widely used being the ones listed below [51].

Gradient Descent This algorithm is an iterative process that calculates the partial derivatives of the loss function with respect to the weight and biases. After the calculation of the gradients of each neuron, they are multiplied by a learning rate and the weights and biases are updated in the opposite direction of the gradient. The parameters of the NNs are updated as

$$w_i := w_i - \eta \nabla_w L \quad (2.6)$$

where $\nabla_w L$ is the gradient of the loss function with respect to the weight of the neuron and η is the learning rate.

Stochastic Gradient Descent It is also an algorithm based on the gradients of the loss function. In gradient descent, the gradient is computed for the whole training dataset, which can be computationally costly. Alternatively, in this algorithm, the gradient is calculated for random batches of the training data, which makes this algorithm better for large datasets and can be a way to avoid local minimums. The size of the batches is another hyperparameter of ML algorithms that should be carefully chosen to reach the

better results possible. Choosing a small value for the batch size can generally give better and faster results, but also leads to stability problems.

Adaptive Moment Estimation (ADAM) It is a combination between stochastic gradient descent and two other optimization methods proposed in the literature, the adaptive gradient algorithm and the root-mean-square propagation. The main difference to the previous optimization algorithm is that ADAM uses different learning rates for each weight. It computes individual adaptive learning rates from estimates of the first and second moments of the gradient for each weight and bias [52].

2.1.3 Learning Rate

The learning rate is an important hyperparameter in ML algorithms because it controls the size of each step taken in the optimization process and it prevents possible convergence problems due to over or undershooting. If the learning rate is too low, it is likely that the training is going to take too much time to converge and the desired minimum loss value is not reached, as it is shown in a simple example shown in Figure 2.5a. On the opposite side, if the learning rate has a higher value, there is a risk of not reaching the minimum value for the loss function, or it can even diverge (Figure 2.5b). If the learning rate is chosen properly, the minimum loss function can be reached, as it is shown in Figure 2.5c.

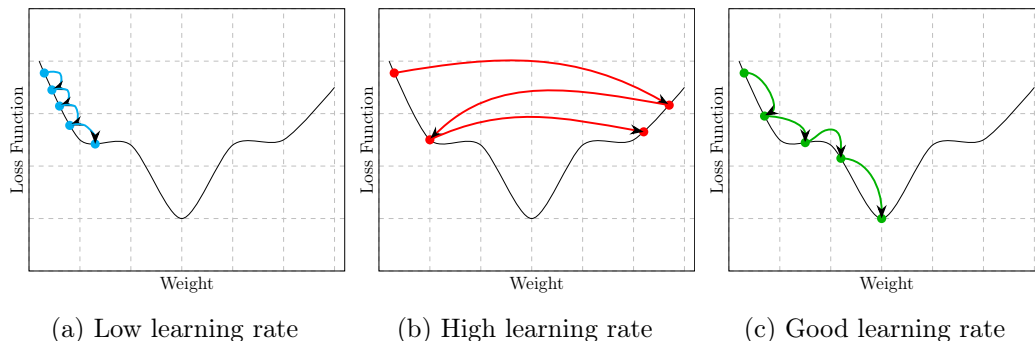


Figure 2.5: Influence of the learning rate in the value of the loss function

2.2 Hyperparameters

In this section, some hyperparameters of ML algorithms were introduced, such as the learning rate and the batch size. Contrarily to the weights and biases, the hyperparameters can not be learned directly by the model and they need to be set at the beginning of the training stages. The number of layers and the number of neurons of each hidden layer are also hyperparameters that have to be chosen before the training process. For complex models, it is important to find the best hyperparameters to archive the best performance on the data in a reasonable amount of time. This process is called hyperparameter optimization

or tuning. Unfortunately, the relationship between the hyperparameters and the performance of ML algorithms is not clear and it is necessary to train the model with different combinations of hyperparameters and then compare the performance to choose the best one [53].

There are two main types of hyperparameter optimization methods: manual search and automatic search. The former requires experience and relies on intuition. In addition, the number of possible hyperparameters can be quite large, which makes the process of manual search hard to manage and time inefficient [53]. The automatic search methods overcome these drawbacks and many different algorithms have been proposed, such as the ones listed below:

Grid Search This method is based on an exhaustive search between a set of values defined for the hyperparameters. It trains the model with each combination possible from the values given for the hyperparameters and evaluates the model based on performance metrics. It is simple and effective if the hyperparameter space is small. Although, its efficiency decreases dramatically as the number of hyperparameters and its possible range increase [53].

Random Search This algorithm tries a random combination of the possible values for the hyperparameters and is particularly efficient when only a few hyperparameters have significant importance. Thus, it improves efficiency by reducing the search for hyperparameters of lower significance and can be a good alternative to grid search in a high-dimensional space. However, for complex models, the random search can be unreliable [53].

Bayesian Optimization Bayesian optimization uses a probabilistic surrogate model to relate the hyperparameters and the unknown objective function. This model is then iteratively refined based on new observations, balancing exploration and exploitation, and the process ends when convergence is reached or a stopping criterion is met. In this way, there is more focus on promising regions of the hyperparameter space and, consequently, more efficiency. Usually, it is assumed that the optimization function obeys a Gaussian distribution [53, 54].

In this work, an optimization software called Optuna [55] was used. It uses a variant of the Bayesian optimization called Tree-structured Parzen which operates with the ratio of Parzen estimators for good and bad observations instead of the standard Gaussian process. Thus, after defining the hyperparameter space, the objective function is evaluated with a chosen set of hyperparameters and afterwards, two probability distributions are calculated, one for good configurations and another for bad ones. Finally, a new set of hyperparameters is sampled from the good and the bad configuration and the selection is done according to the highest ratio between the good and the bad distribution. The process is then repeated until the stopping criterium is reached [55, 56]. Optuna has three main design features:

Define-by-run style Application Programming Interface This allows the users to define the parameter search space without the need to explicitly define everything about the optimization strategy in advance. This is possible because Optuna formulates the hyperparameter optimization as a process of minimization/maximization of an objective function that takes the hyperparameters as input. Each optimization process is a study, and each evaluation of the objective function is a trial. An objective function is defined and the hyperparameters for each trial can be dynamically generated with the `suggest` Application Programming Interface (API) and then the `optimize` API can be invoked with the objective function as input [55].

Efficient pruning and sampling mechanism The efficiency of the searching and performance estimation strategies is key for a cost-effective optimization method. Regarding the searching strategy, there are generally two types of sampling methods: relational sampling, which uses the correlation between parameters, and independent sampling, which samples each parameter independently. Optuna is able to use both of them and handle various sampling methods. Considering the performance estimation strategies, Optuna has a strategy to stop unpromising trials, which is often referred to as pruning, which is different from the early stopping mechanism also used in ML to prevent overfitting, as addressed before. The pruning algorithm is extremely important to reduce the cost of the tuning process and it can be divided into two phases: one that checks periodically the intermediate objective and another that terminate the trials with unpromising results [55].

Easy to set up Optuna has a versatile architecture that can handle a wide range of problems. Optuna provides visualization and analysis of studies in real-time and by default, it uses its built-in in-memory data structure as the storage back-end [55].

Chapter 3

Continuum Mechanics

This chapter covers the fundamental aspects and concepts related to the mechanics of continuum bodies in motion. Continuum mechanics allow for the explanation of many physical phenomena without the need for a deep knowledge of the internal microstructure. It studies the kinematics (motion and deformation), the stress state in a continuum body and the balance principles. These three topics are the ones that are going to be addressed in this chapter, as well as the FE method. The chapter summarizes information from [57] and [58], where a more detailed explanation can be found.

3.1 Kinematics

Kinematics describes the motion and the deformation of a body in space, considering the time dependency and without consideration of the forces that cause them. It defines many important quantities and constitutive relations that are essential to characterize motion and deformation.

3.1.1 Motion

With a continuum approach and assuming the continuum theory, bodies have a continuous distribution of matter in space and time and can be considered composed of an infinite number of particles. Each region that the body occupies in space is called a configuration, and there are two with significant relevance: the reference and the current configurations. The former is usually the configuration occupied by the material at time $t = 0$, in which case is called initial configuration denoted by Ω_0 and the latter is the configuration of the body at a given time $t > 0$ and denoted by Ω .

A set of vectors can identify each configuration and define a particle of the body. The components of each one of these vectors are known as the material coordinates (X_1, X_2, X_3) if they define a particle in the reference configuration, whereas the ones that define a particle in the current configuration are called spacial coordinates (x_1, x_2, x_3) .

The motion of a continuum can be described by

$$\mathbf{x} = \phi(\mathbf{X}, t) \quad (3.1)$$

where ϕ is a vector field that carries each point of the reference configuration to the current configuration.

If the motion is invertible, the reference configuration can be obtained as

$$\mathbf{X} = \phi^{-1}(\mathbf{x}, t). \quad (3.2)$$

The motion can change some properties of the body, such as temperature, and these changes can be described in two ways: with the Lagrangian (or material) description or with the Eulerian (or spatial) description. The former characterizes the motion concerning the material coordinates and the focus is on a particle, while the latter describes the motion in the spatial coordinates and it focuses on a particular point in space.

The displacement field of a particle can be defined as

$$\mathbf{u}(\mathbf{X}, t) = \phi(\mathbf{X}, t) - \mathbf{X}. \quad (3.3)$$

3.1.2 Deformation Gradient

The deformation gradient \mathbf{F} is a second-order tensor and is the fundamental measure of deformation. Deformation can be defined as the change in size, shape or orientation of a body when moved from the reference (undeformed) configuration to a current (deformed) configuration. The deformation gradient characterizes the deformations, which means that it represents the spatial changes of each material point during motion with respect to its reference configuration. Thus, it relates \mathbf{X} with \mathbf{x} as

$$\mathbf{F}(\mathbf{X}, t) = \frac{\partial \phi(\mathbf{X}, t)}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \nabla_X \phi(\mathbf{X}, t) \quad (3.4)$$

where ∇_X is called the material gradient operator.

The deformation gradient can also be defined in the deformed configuration by

$$\mathbf{F}(\mathbf{x}, t) = [\nabla_x \phi^{-1}(\mathbf{x}, t)]^{-1} \quad (3.5)$$

where ∇_x is the spatial gradient operator.

If the displacement field \mathbf{u} is used, the deformation gradient can also be expressed as

$$\mathbf{F}(\mathbf{x}, t) = \nabla_X \mathbf{u} + \mathbf{I} \quad (3.6)$$

where \mathbf{I} is the second-order identity tensor.

The deformation gradient is a two-point tensor and it is homogeneous if it is independent of the material coordinates. The determinant of the deformation gradient is called

the Jacobian of the deformation, represents the local rate of change of the deformed configuration volume v with respect to the reference configuration volume V and is given by

$$J = \det(\mathbf{F}) = \frac{dv}{dV}. \quad (3.7)$$

Its value has to be greater than zero to adequately describe physical deformations and if $J = 1$, the deformation is isochoric.

With the definition presented above, it is possible to obtain relations between the reference and the current configuration

$$\begin{aligned} d\mathbf{x} &= \mathbf{F}d\mathbf{X} \\ d\mathbf{a} &= J\mathbf{F}^{-T}d\mathbf{A} \\ dv &= JdV. \end{aligned} \quad (3.8)$$

The deformation gradient is not a precise measure of deformations because it is not a symmetric tensor and in the absence of deformation (rigid body translations), this tensor is not equal to a null matrix. In fact, it is equal to the identity matrix, which is not optimal. In addition, in rigid body rotations, this tensor is different from the identity matrix, suggesting wrongly that there was a deformation.

3.1.3 Polar Decomposition

The deformation gradient can be decomposed into the product of two tensors and defined by

$$\mathbf{F} = \mathbf{R}\mathbf{U} = \mathbf{V}\mathbf{R} \quad (3.9)$$

where \mathbf{U} and \mathbf{V} are stretch tensors and \mathbf{R} is a rotation tensor that has no contribution to strain.

The rotation tensor \mathbf{R} is orthogonal ($\mathbf{R}^T = \mathbf{R}^{-1}$), unique, unimodular ($\det(\mathbf{R}) = 1$) and measures the change of local orientation.

The right-stretch tensor \mathbf{U} and the left-stretch tensor \mathbf{V} are unique, positive definite and symmetric and they measure the change of local shape. They can be written as

$$\begin{aligned} \mathbf{U} &= \sqrt{\mathbf{C}} \\ \mathbf{V} &= \sqrt{\mathbf{B}} \end{aligned} \quad (3.10)$$

where \mathbf{C} and \mathbf{B} are the right and left Cauchy-Green tensors, respectively.

With the properties of these tensors and with eq. (3.9), it is possible to get the following relations

$$\mathbf{C} = \mathbf{U}^2 = \mathbf{F}^T\mathbf{F} \quad (3.11)$$

$$\mathbf{B} = \mathbf{V}^2 = \mathbf{F}\mathbf{F}^T. \quad (3.12)$$

Unlike the deformation gradient, the right and left Cauchy-Green tensors are insensitive to pure body rotations and both of them are symmetric and positive definite.

3.1.4 Stretch

Considering a line element in the reference configuration $d\mathbf{X}$ with its direction described by the unit vector \mathbf{a}_0 , it is possible to define the stretch vector ($\boldsymbol{\lambda}_{\mathbf{a}_0}$) as

$$\boldsymbol{\lambda}_{\mathbf{a}_0}(\mathbf{X}, t) = \mathbf{F}\mathbf{a}_0. \quad (3.13)$$

The length of this vector is called the stretch ratio (λ) and it measures the changes in length that occurred during the motion. It is defined as the ratio between the length of a deformed line element and the length of the corresponding undeformed line element and is defined as

$$\lambda = |\boldsymbol{\lambda}_{\mathbf{a}_0}| = \frac{|d\mathbf{x}|}{|d\mathbf{X}|}. \quad (3.14)$$

If $\lambda < 1$ the element was compressed, if $\lambda = 1$, the element kept the same length, and if $\lambda > 1$, the element was extended.

Alternatively, it is possible to express a relation between the stretch and the right Cauchy-Green deformation tensor as

$$\lambda = \sqrt{\mathbf{F}\mathbf{a}_0 \cdot \mathbf{F}\mathbf{a}_0} = \sqrt{\mathbf{a}_0 \cdot \mathbf{F}^T \mathbf{F}\mathbf{a}_0} = \sqrt{\mathbf{a}_0 \cdot \mathbf{C}\mathbf{a}_0}. \quad (3.15)$$

There is always a set of three mutually orthogonal directions along which the material undergoes a pure stretch (changes in length without changes in angles between them). These directions are called the principal reference directions, they are the eigenvectors of \mathbf{U} and are denoted by $\hat{\mathbf{N}}_{i=1,2,3}$. The associated eigenvalues are the principal stretches and are denoted by $\lambda_{i=1,2,3}$. The principal reference directions are also the eigenvectors of \mathbf{C} and the eigenvalues of this tensor are the stretches squared

$$\mathbf{U}^2 \hat{\mathbf{N}}_i = \lambda_i^2 \hat{\mathbf{N}}_i = \mathbf{C} \hat{\mathbf{N}}_i, \quad i = 1, 2, 3. \quad (3.16)$$

Following the same reasoning, it is possible to define the eigenvalue problem for \mathbf{V} and \mathbf{B} as

$$\mathbf{V}^2(\mathbf{R}\hat{\mathbf{N}}_i) = \lambda_i^2(\mathbf{R}\hat{\mathbf{N}}_i) = \mathbf{B}(\mathbf{R}\hat{\mathbf{N}}_i), \quad i = 1, 2, 3. \quad (3.17)$$

Is it possible now to conclude that the eigenvectors of \mathbf{V} and \mathbf{B} are equal to the eigenvectors of \mathbf{U} and \mathbf{C} but rotated with \mathbf{R} . Therefore, the eigenvectors of \mathbf{V} and \mathbf{B} are the principal spatial directions denoted by $\hat{\mathbf{n}}_{i=1,2,3}$ and defined as

$$\hat{\mathbf{n}} = \mathbf{R}\hat{\mathbf{N}}. \quad (3.18)$$

Finally, it is relevant to express the deformation gradient and the other deformation tensors in terms of principal stretches:

$$\begin{aligned}\mathbf{F} &= \sum_{i=1}^3 \lambda_i \hat{\mathbf{n}}_i \otimes \hat{\mathbf{N}}_i \\ \mathbf{C} = \mathbf{U}^2 &= \sum_{i=1}^3 \lambda_i^2 \hat{\mathbf{N}}_i \otimes \hat{\mathbf{N}}_i \\ \mathbf{B} = \mathbf{V}^2 &= \sum_{i=1}^3 \lambda_i^2 \hat{\mathbf{n}}_i \otimes \hat{\mathbf{n}}_i.\end{aligned}\tag{3.19}$$

Special attention has to be taken to the deformation gradient because it is a non-symmetric tensor, so the principal stretches are not their eigenvalues.

3.1.5 Strain Measures

Strain is a measure of deformation which characterizes the relative displacement between particles (changes in distances and angles). To properly quantify straining, proper strain measures are defined in the literature. The choice is influenced by mathematical and physical convenience, but two main classes of strain measures are of great importance: the Lagrangian strain tensors and the Eulerian strain tensors. The former ones are also called generalized strain tensors in the material description and are based on the right Cauchy-Green stretch tensor and are defined as

$${}^{(m)}\mathbf{E} = \begin{cases} \frac{1}{m}(\mathbf{U}^m - \mathbf{I}) & \text{if } m \neq 0 \\ \ln \mathbf{U} & \text{if } m = 0 \end{cases}.\tag{3.20}$$

The other important family of strain tensors can be defined with the the left Cauchy-Green stretch tensor. They are also called the generalized strain tensors in spatial description and are expressed as

$${}^{(m)}\mathbf{e} = \begin{cases} \frac{1}{m}(\mathbf{V}^m - \mathbf{I}) & \text{if } m \neq 0 \\ \ln \mathbf{V} & \text{if } m = 0 \end{cases}.\tag{3.21}$$

For $m = 2$ and $m = -2$, two well-known strain tensors are obtained: the Green-Lagrange strain tensor \mathbf{E} , given by

$${}^{(2)}\mathbf{E} = \mathbf{E} = \frac{1}{2}(\mathbf{F}^T \mathbf{F} - \mathbf{I}) = \frac{1}{2}(\mathbf{C} - \mathbf{I})\tag{3.22}$$

and the Euler-Almansi strain tensor \mathbf{e} , given by

$${}^{(-2)}\mathbf{e} = \mathbf{e} = \frac{1}{2}(\mathbf{I} - \mathbf{F}^{-T} \mathbf{F}^{-1}) = \frac{1}{2}(\mathbf{I} - \mathbf{B}^{-1}).\tag{3.23}$$

These two strain tensors are independent of rotations and just contain information about stretches. Therefore, they are suited to analyse the strain of bodies submitted to large displacements, rotations and deformations. The Euler-Almansi strain tensor is less used, even though it allows for strain analysis from the deformed configuration.

For $m = 0$, the logarithmic strain tensors in material and spatial descriptions are obtained.

Figure 3.1 shows the difference between some of the most common strain measures, where ϵ is the infinitesimal strain tensor, usually used for deformations where the gradient of the displacement is small compared to unity. It is possible to observe that for values of stretch around 1, all the strain tensors give nearly the same value of strain, but for higher stretches, the differences are notorious, which shows the importance of selecting the appropriate strain measure for the case in analysis.

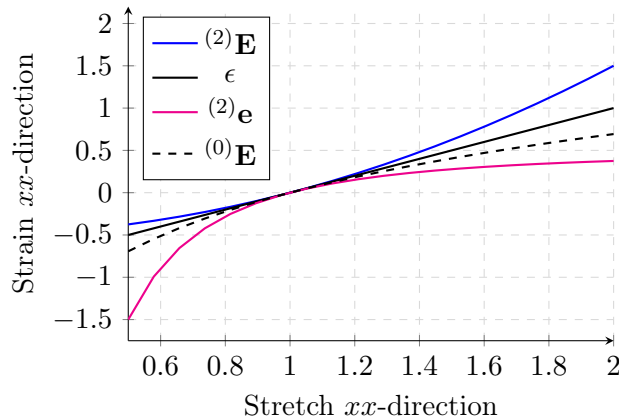


Figure 3.1: Comparison between strain measures for a uniaxial stretch in the x direction

3.1.6 Directional Derivative and Lie Time Derivatives

Before proceeding to the following section, it is important to introduce the concepts of directional derivative and Lie time derivatives, which are relevant for the linearization of equilibrium equations.

The directional derivative is a generalisation of a common derivative because it gives the change in a field due to a slight variation in a variable on which the field depends.

A given scalar field Φ that varies in a three-dimensional space has a gradient ($\nabla\Phi = \partial\Phi/\partial\mathbf{x}$) that defines a vector perpendicular to the surface of constant Φ . If a unit vector \mathbf{n} is defined at a point \mathbf{x} normal to the surface (direction of $\nabla\Phi$) and another vector \mathbf{v} is defined in the same point \mathbf{x} but inclined at an angle θ of $\nabla\Phi$, it is possible to define the concept of directional derivative as the dot product between the gradient of the scalar field and the vector \mathbf{v} , given by

$$D_{\mathbf{v}}\Phi(\mathbf{x}) = \nabla\Phi \cdot \mathbf{v} = \frac{d}{d\epsilon}\Phi(\mathbf{x} + \epsilon\mathbf{v})|_{\epsilon=0}. \quad (3.24)$$

The directional derivative characterizes the rate of change of Φ along the line that goes through \mathbf{x} and in the direction \mathbf{v} .

Additionally, the directional derivative of a general operator Φ applied to the motion $\phi(\mathbf{X}, t)$ in the direction of \mathbf{v} coincides with the time derivative of Φ :

$$D_{\mathbf{v}}\Phi(\mathbf{x}) = \frac{d}{dt}\Phi[\phi(\mathbf{X}, t)]. \quad (3.25)$$

The Lie time derivative of a spatial field $f(\mathbf{x}, t)$ gives the change of the field relative to a vector field \mathbf{v} and is denoted by $\mathcal{L}_{\mathbf{v}}(f)$. To obtain the Lie time derivative, the following steps are required:

1. Compute the pull-back of f to the reference configuration $\mathcal{F}(\mathbf{X}, t) = \chi_*^{-1}(f(\mathbf{x}, t))$, where χ_*^{-1} denotes the pull-back operator;
2. Take the time derivative: $\dot{\mathcal{F}}$;
3. Compute the push-forward of the result to the current configuration: $\chi_*(\dot{\mathcal{F}})$.

Thus, the Lie time derivative can be defined as

$$\mathcal{L}_{\mathbf{v}}(f) = \chi_* \left(\frac{\partial}{\partial t} \chi_*^{-1}(f) \right) = \chi_*(\dot{\mathcal{F}}). \quad (3.26)$$

As mentioned before, the material time derivative can be obtained from the directional derivative. Thus, the Lie time derivative can be expressed as

$$\mathcal{L}_{\mathbf{v}}(f) = \chi_* (D_{\mathbf{v}}\chi_*^{-1}(f)) = \chi_*(D_{\mathbf{v}}\mathcal{F}). \quad (3.27)$$

Therefore, the Lie time derivative of a spatial field f is the push-forward of the directional derivative of the associated material field \mathcal{F} in the direction of the velocity vector \mathbf{v} . This concept is extremely useful to compute stress rates and in the process of linearization and variation, as it is going to be explained in the following sections.

3.1.7 Material and Spatial Time Derivatives

Velocity and material time derivatives are necessary to analyse time-dependent nonlinear processes. In addition, it is convenient to establish the equilibrium equations in terms of virtual velocities and associated virtual time-dependent quantities, even if the process is not rate-dependent.

The velocity and the acceleration field are defined as:

$$\mathbf{v}(\mathbf{X}, t) = \frac{\partial \phi(\mathbf{X}, t)}{\partial t} \quad (3.28)$$

$$\mathbf{a}(\mathbf{X}, t) = \frac{\partial \mathbf{v}(\mathbf{X}, t)}{\partial t} = \frac{\partial^2 \phi(\mathbf{X}, t)}{\partial t^2}. \quad (3.29)$$

The spatial velocity gradient tensor \mathbf{l} is the derivative of the velocity field given in the spatial description

$$\mathbf{l} = \frac{\partial \mathbf{v}(\mathbf{x}, t)}{\partial \mathbf{x}} = \nabla_x \mathbf{v}. \quad (3.30)$$

The material time derivative of \mathbf{F} is expressed as

$$\dot{\mathbf{F}} = \frac{\partial}{\partial t} \left(\frac{\partial \phi(\mathbf{X}, t)}{\partial \mathbf{X}} \right) = \frac{\partial}{\partial \mathbf{X}} \left(\frac{\partial \phi(\mathbf{X}, t)}{\partial t} \right) = \frac{\partial \mathbf{v}(\mathbf{X}, t)}{\partial \mathbf{X}}. \quad (3.31)$$

Introducing the spatial velocity gradient, it is possible to obtain the relation given by

$$\dot{\mathbf{F}} = \frac{\partial \mathbf{v}(\mathbf{X}, t)}{\partial \mathbf{x}} \frac{\partial \phi(\mathbf{X}, t)}{\partial \mathbf{X}} = \mathbf{lF}. \quad (3.32)$$

Thus, another expression for the spatial velocity gradient can be obtained as

$$\mathbf{l} = \dot{\mathbf{F}}\mathbf{F}^{-1}. \quad (3.33)$$

The spatial velocity gradient can be decomposed into two different tensors: the rate of deformation tensor \mathbf{d} and the rate of rotation tensor \mathbf{w} :

$$\mathbf{l} = \mathbf{d} + \mathbf{w}. \quad (3.34)$$

The former is also known as the stretch tensor, is the symmetric part of \mathbf{l} and is given by

$$\mathbf{d} = \frac{1}{2}(\mathbf{l} + \mathbf{l}^T) \quad (3.35)$$

whereas the latter is the antisymmetric part of \mathbf{l} , also known as spin or vorticity tensor and given by

$$\mathbf{w} = \frac{1}{2}(\mathbf{l} - \mathbf{l}^T). \quad (3.36)$$

The rate of deformation tensor quantifies the rate of stretching of a material's fibre in the deformed solid and the rate of rotation tensor measures the average angular velocity of all material fibres passing through a material point. Both tensors are spatial fields and they only involve quantities acting on the current configuration.

Another important rate tensor is the so-called material strain rate tensor, which is the derivative of the Green-Lagrange strain tensor with respect to time and is given by

$$\dot{\mathbf{E}} = \frac{1}{2}\dot{\mathbf{C}} = \frac{1}{2}(\mathbf{F}^T \dot{\mathbf{F}} + \dot{\mathbf{F}}^T \mathbf{F}). \quad (3.37)$$

This tensor gives the current rate of stretching in terms of the initial elemental vectors and it is essentially a pull-back of the already introduced rate of deformation tensor \mathbf{d} . Thus,

the two tensors are related as

$$\begin{aligned}\dot{\mathbf{E}} &= \mathbf{F}^T \mathbf{dF} \\ \mathbf{d} &= \mathbf{F}^{-T} \dot{\mathbf{E}} \mathbf{F}^{-1}.\end{aligned}\tag{3.38}$$

As mentioned in Section 3.1.6, the directional derivative of \mathbf{E} in the direction of \mathbf{v} is equal to the material strain rate tensor:

$$D_{\mathbf{v}} \mathbf{E} = \dot{\mathbf{E}}.\tag{3.39}$$

The material time derivative of the Euler-Almansi strain tensor is given by

$$\dot{\mathbf{e}} = \mathbf{d} - \mathbf{l}^T \mathbf{e} - \mathbf{e} \mathbf{l}.\tag{3.40}$$

The time rate of change of the right and left Cauchy-Green tensors are easily obtained with the previous definitions and they are given by

$$\begin{aligned}\dot{\mathbf{C}} &= 2\dot{\mathbf{E}} \\ \dot{\mathbf{B}} &= \mathbf{I} \mathbf{B} + \mathbf{B} \mathbf{l}^T.\end{aligned}\tag{3.41}$$

It is also interesting to provide the relation for the material time derivative of the volume ratio J , given by

$$\dot{J} = \frac{\partial J}{\partial \mathbf{F}} : \dot{\mathbf{F}} = J \mathbf{F}^{-T} : \dot{\mathbf{F}}.\tag{3.42}$$

With equalities from eq. (3.42) it is possible to get alternative statements to $J = 1$ that characterize necessary and sufficient conditions for an isochoric motion. One of the most important for the treatment of incompressible solids is $\mathbf{F}^{-T} : \dot{\mathbf{F}} = 0$.

3.2 Stress Measures

During motion, deformations occur and they cause interactions between the body's particles. The result is stress and for each strain measure, there is a different stress measure. One of the most commonly used stress measures is the Cauchy or true stress tensor $\boldsymbol{\sigma}$.

The Cauchy or surface traction vector \mathbf{t} measures the force \mathbf{f} applied in an element surface defined in the current configuration with a unitary normal vector \mathbf{n} and is given by

$$\mathbf{t} = \lim_{da \rightarrow 0} \frac{d\mathbf{f}}{da}.\tag{3.43}$$

The Cauchy stress tensor enables the mapping of a vector normal to a surface to the traction vector acting on that surface. This mapping is known as Cauchy's Law and is given by

$$\mathbf{t} = \boldsymbol{\sigma} \mathbf{n}.\tag{3.44}$$

This stress tensor is symmetric, widely used in mechanics, and is expressed in terms of spatial coordinates. Since in most problems, the current configuration is not known, this stress tensor is not the most convenient one to use. If the deformations are small, the current configuration is somehow similar to the reference one and the Cauchy stress tensor is a reasonable way of describing the action of surface forces. However, for large deformations, it is necessary to use other stress tensors expressed in terms of the material coordinates.

The Kirchhoff stress tensor $\boldsymbol{\tau}$ is one of those tensors. It has no physical meaning, but it is useful, for example, for metal plasticity and it is given by

$$\boldsymbol{\tau} = J\boldsymbol{\sigma}. \quad (3.45)$$

Finally, there are the Piola-Kirchhoff stress tensors that express stress in terms of material coordinates and are suitable for finite strain applications. The first Piola-Kirchhoff traction vector (\mathbf{T}) measures the force \mathbf{f} applied to an element surface defined in the reference configurations with a normal vector \mathbf{N} and is given by

$$\mathbf{T} = \lim_{dA \rightarrow 0} \frac{d\mathbf{f}}{dA}. \quad (3.46)$$

A relation between this traction vector and the first Piola-Kirchhoff stress tensor \mathbf{P} is given by

$$\mathbf{T} = \mathbf{P}\mathbf{N}. \quad (3.47)$$

The first Piola-Kirchhoff stress tensor is a two-point tensor since it relates forces acting in the current configuration with a surface element in the reference configuration and is defined as

$$\mathbf{P} = \boldsymbol{\tau} \mathbf{F}^{-T}. \quad (3.48)$$

However, this tensor is not symmetric. This can be easily observed with eq. (3.48) since the deformation gradient is not symmetric.

Contrarily, the second Piola-Kirchhoff stress tensor \mathbf{S} is symmetric and it relates forces in the current configuration with areas in the reference configuration and is given by

$$\mathbf{S} = \mathbf{F}^{-1}\boldsymbol{\tau}\mathbf{F}^{-T}. \quad (3.49)$$

A relation between the two Piola-Kirchhoff stress tensors can also be obtained as

$$\mathbf{P} = \mathbf{F}\mathbf{S}. \quad (3.50)$$

It is also possible to relate the Piola-Kirchhoff stress tensors with the Cauchy stress

tensor:

$$\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-\text{T}} \quad (3.51)$$

$$\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-\text{T}}. \quad (3.52)$$

The Piola-Kirchhoff stress tensors are useful stress measures for computational mechanics and to formulate constitutive relations because they are work conjugates of some presented strain measures: (\mathbf{P} with \mathbf{F} and \mathbf{S} with \mathbf{E}).

In Figure 3.2, it is shown the difference between some of the stress measures presented.

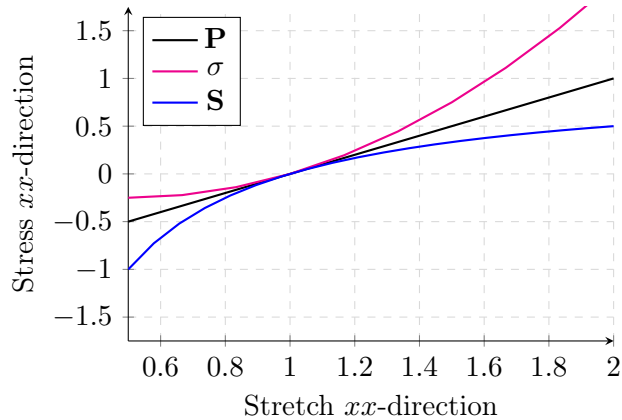


Figure 3.2: Comparison between stress measures for a uniaxial stretch in the x direction

3.3 Balance Principles

In this section, the mathematical description of some fundamental laws of physics that govern the motion of a continuum is going to be addressed. The mathematical representations previously introduced are not enough to make predictions if there is no relation between them, which highlights the importance of balance and conservation principles for continuum mechanics.

3.3.1 Conservation of Mass

From eq. (3.8), it is possible to define the element of mass in terms of initial and current densities as

$$dm = \rho_0 dV = \rho dv \quad (3.53)$$

and the conservation of mass can be expressed as

$$(\rho_0 - \rho J)dV = 0. \quad (3.54)$$

Finally, taking the material derivative of eq. (3.54) for the current density ρ , the equation for the conservation of mass can be written as

$$\frac{d\rho}{dt} + \rho \operatorname{div}_x \dot{\mathbf{u}} = 0. \quad (3.55)$$

3.3.2 Balance of Momentum

The linear momentum (\mathbf{L}) is defined as

$$\mathbf{L}(t) = \int_{\Omega} \rho(\mathbf{x}, t) \mathbf{v}(\mathbf{x}, t) dv. \quad (3.56)$$

The principle of conservation of linear momentum states that the rate of change of the total linear momentum of a continuum $\dot{\mathbf{L}}$ is equal to the resultant force \mathbf{F}_r and is given by

$$\dot{\mathbf{L}}(t) = \int_{\Omega} \rho(\mathbf{x}, t) \dot{\mathbf{v}}(\mathbf{x}, t) dv = \mathbf{F}_r(t). \quad (3.57)$$

The resultant force on the body in the current configuration is given by

$$\mathbf{F}_r(t) = \int_{\partial\Omega} \mathbf{t} ds + \int_{\Omega} \mathbf{b} dv \quad (3.58)$$

where \mathbf{b} is a spatial vector field called body force that is defined per unit of the current volume of the region Ω acting on a particle and \mathbf{t} is the already introduced Cauchy traction vector that acts on a boundary surface $\partial\Omega$ of an arbitrary region Ω .

With the integral of Cauchy's stress theorem given by eq. (3.44) and the divergence theorem, it is possible to get the relation given by

$$\int_{\partial\Omega} \mathbf{t} ds = \int_{\partial\Omega} \boldsymbol{\sigma} \mathbf{n} ds = \int_{\Omega} \operatorname{div}_x \boldsymbol{\sigma} dv. \quad (3.59)$$

Therefore, it is now possible to obtain the strong form of the equilibrium equation, which describes the equilibrium between internal and external forces in the body. The strong form of the equilibrium equation is given by

$$\operatorname{div}_x \boldsymbol{\sigma} + \mathbf{b} = \rho \ddot{\mathbf{u}}. \quad (3.60)$$

Alternatively, it is possible to express eq. (3.60) with the material description as follows

$$\operatorname{div}_X \mathbf{P} + \mathbf{B}_f = \rho_0 \ddot{\mathbf{u}} \quad (3.61)$$

where \mathbf{B}_f is the reference body force.

The use of the strong formulation of the equilibrium equation in complex problems is sometimes difficult because it involves differentiations. This led to the development of the

weak formulation derived from the principle of virtual work which is going to be explained in further detail in a subsequent section.

3.3.3 Thermodynamic Laws

The balance of thermal energy states that the rate of work done on the continuum body, which is the sum of the rate of internal mechanical work \mathcal{P}_{int} and the rate of thermal work \mathcal{Q} , is equal to the rate of internal energy \mathcal{E} :

$$\mathcal{P}_{\text{int}}(t) + \mathcal{Q}(t) = \mathcal{E}(t) \quad (3.62)$$

where each one of the rates present in the above equation is given by

$$\mathcal{P}_{\text{int}}(t) = \int_{\Omega} \boldsymbol{\sigma} : \mathbf{d} \, dv \quad (3.63)$$

$$\mathcal{Q}(t) = \int_{\Omega} \mathbf{q} \, dv + \int_{\Omega} r \, dv \quad (3.64)$$

$$\mathcal{E}(t) = \int_{\Omega} \dot{e} \, dv. \quad (3.65)$$

Combining the previous equation with the balance of mechanical energy, it is possible to obtain the first law of thermodynamics that imposes the conservation of energy and is expressed as

$$\dot{e} = \boldsymbol{\sigma} : \mathbf{d} + r - \text{div}_x \mathbf{q} \quad (3.66)$$

where e is the internal energy, r is the heat source per unit time and per unit of the current volume and \mathbf{q} is Cauchy heat flux defined per unit surface area in the current configuration. Throughout this dissertation, the temperature remains constant in all the problems analysed. Hence, eq. (3.66) reduces to

$$\dot{e} = \boldsymbol{\sigma} : \mathbf{d}. \quad (3.67)$$

The first law of thermodynamics can also be defined in the material description as

$$\dot{e} = \mathbf{P} : \dot{\mathbf{F}}. \quad (3.68)$$

It is known that heat always flows from the warmest region of a body to the coldest one, and a key concept associated with this is entropy, which can be viewed as the quantitative measure of microscopic randomness and disorder. The first law of thermodynamics is not sufficient to ensure the irreversibility of the entropy productions and this is why there is

the second law of thermodynamics, given by

$$\dot{s} \geq \frac{r}{T} - \frac{\operatorname{div}_x \mathbf{q}}{T} + \frac{\mathbf{q} \cdot \operatorname{div}_x T}{T^2} \quad (3.69)$$

where s is the entropy and T the temperature. For isothermal processes, the second law of thermodynamics reduces to

$$T\dot{s} \geq 0. \quad (3.70)$$

The Helmholtz potential represents the amount of work that a closed thermodynamic system can do at constant temperature and volume and it can be written as

$$\Psi = e - Ts. \quad (3.71)$$

The two laws of thermodynamics can be combined with eq. (3.71) to obtain the the Clausius–Duhem inequality given by

$$\boldsymbol{\sigma} : \mathbf{d} - \frac{\mathbf{q} \cdot \nabla_x T}{T} \geq (\dot{\Psi} + \dot{T}s). \quad (3.72)$$

This inequality tells that an isolated system tends toward a state of equilibrium at minimum free energy. It can also be expressed in the reference configuration as

$$\mathbf{P} : \dot{\mathbf{F}} - \frac{\mathbf{Q} \cdot \nabla_X T}{T} \geq (\dot{\Psi} + \dot{T}s). \quad (3.73)$$

Considering an isothermal process, it is possible to obtain the Clausius-Planck inequality, which is extremely useful to derive constitutive laws and is given by

$$\mathbf{P} : \dot{\mathbf{F}} - \dot{\Psi} \geq 0. \quad (3.74)$$

3.4 Constitutive Equations

The equations presented in Section 3.1 and Section 3.2 hold for any continuum body at all times. However, they do not distinguish different materials and they are alone insufficient to determine the material response. Therefore, it is necessary to establish a constitutive model that should approximate the observed physical behaviour of a real material under specific conditions of interest. This constitutive model is a set of equations that express stress in terms of other fields, such as strain. In addition, these equations must satisfy some physics laws but, in solid mechanics, a phenomenological approach is usually followed, where the main concern is fitting the mathematical equations to experimental data. The constitutive equations must obey the thermodynamic laws and satisfy objectivity (frame indifference), which guarantees that the reference frame used to describe the behaviour of a material does not change the physical laws and equations that characterize it. In this

section, nonlinear constitutive equations are presented for different types of hyperelastic materials.

3.4.1 Objectivity Rates

Objectivity is an important topic in solid mechanics because material properties must be independent of changes in the observer's point of view. Some quantities that describe the behaviour of a solid, such as the distance between two particles, remain unchanged from the point of view of an observer attached to and rotating with the body. In this case, such quantities are objective even if their spatial description change [58].

To show this concept, a motion which differs from the one given by the vector field ϕ by a superposed rigid body motion is considered:

$$\mathbf{x}^+ = \mathbf{Q}(t)\mathbf{x} \quad (3.75)$$

where \mathbf{Q} is an orthogonal tensor representing a rigid body rotation of the deformed state. The plus sign in the superscript is used to describe the quantities already introduced in the previous sections but associated with the motion given by eq. (3.75). The vector field ϕ^+ describes the new motion and takes the points from the reference configuration to the rotated configuration described by \mathbf{x}^+ :

$$\phi^+ = \mathbf{Q}(t)\phi. \quad (3.76)$$

The vector \mathbf{x} is different from \mathbf{x}^+ but they have equal magnitude. This means that \mathbf{x} is objective or frame-indifferent under rigid body motions. However, there are some quantities, such as the velocity and acceleration fields, that are not objective. Stress and strain tensors used to describe material behaviour must be objective. The deformation gradient, the right and left stretch tensors and the three stress tensors introduced, namely the Cauchy stress tensor and the Piola-Kirchhoff stress tensors, are objective.

The fact that all these tensors are objective is good, but their material time derivatives are not objective in general. For example, the time derivative of the Cauchy stress tensor is not an objective tensor, despite the objectivity of the tensor itself. The time derivative of the Cauchy stress tensor is given by

$$\dot{\sigma}^+ = \dot{\mathbf{Q}}\sigma\mathbf{Q}^T + \mathbf{Q}\dot{\sigma}\mathbf{Q}^T + \mathbf{Q}\sigma\dot{\mathbf{Q}}^T. \quad (3.77)$$

The time derivative of the Cauchy stress is not equal to $\mathbf{Q}\dot{\sigma}\mathbf{Q}^T$, which shows that this rate is not objective. If the material time derivative does not retain objectivity, these quantities are not suitable to formulate constitutive equations in rate form. Thus, it is necessary to use objective time derivatives called objective rates.

The objective stress rates are of bigger importance since the choice of a suitable one is essential for the formulation of constitutive rate equations. There are different ones defined in the literature, with some of the most used ones being the following ones [57–59]:

Truesdell Stress Rate The objective stress rate proposed by Truesdell [60] is one of the simplest ones because it is based on the objectivity of the second Piola–Kirchhoff tensor. It is defined as the push forward of the time derivative of the second Piola–Kirchhoff stress tensor scaled by the inverse of the volume ratio and is given by

$$\boldsymbol{\sigma}^{Trues} = J^{-1} \mathbf{F} \dot{\mathbf{S}} \mathbf{F}^T = \dot{\boldsymbol{\sigma}} - \mathbf{l} \boldsymbol{\sigma} - \boldsymbol{\sigma} \mathbf{l}^T + \boldsymbol{\sigma} \operatorname{tr}(\mathbf{d}). \quad (3.78)$$

The Truesdell stress rate is useful to obtain a constitutive relation in the reference configuration between \mathbf{S} and \mathbf{E} . However, if a constitutive equation in the current configuration is desired, this objective stress rate may not be useful.

Oldroyd Stress Rate The Oldroyd Stress Rate was proposed by Oldroyd and Wilson [61] and is derived from the Lie time derivative of the Cauchy stress as

$$\boldsymbol{\sigma}^{Oldr} = \mathcal{L}_\phi(\boldsymbol{\sigma}) = \mathbf{F} \left[\frac{D(\mathbf{F}^{-1} \boldsymbol{\sigma} \mathbf{F}^{-T})}{Dt} \right] \mathbf{F}^T = \dot{\boldsymbol{\sigma}} - \mathbf{l} \boldsymbol{\sigma} - \boldsymbol{\sigma} \mathbf{l}^T. \quad (3.79)$$

A relation between the Oldroyd and the Truesdell stress rates can be obtained as

$$\boldsymbol{\sigma}^{Oldr} = \boldsymbol{\sigma}^{Trues} - \boldsymbol{\sigma} \operatorname{tr} \mathbf{d}. \quad (3.80)$$

Green-Naghdi Stress Rate This stress rate tensor was proposed in [62] and is a special case of the one proposed by Oldroyd because it consists of the Lie time derivative of the Cauchy stress tensor but \mathbf{F} is replaced by \mathbf{R} . Thus, it ignores the stretch component of the deformation gradient and performs pull-back and push-forward operations only with the rotation tensor \mathbf{R} :

$$\boldsymbol{\sigma}^{GN} = \mathbf{R} \left[\frac{D(\mathbf{R}^T \boldsymbol{\sigma} \mathbf{R})}{Dt} \right] \mathbf{R}^T = \dot{\boldsymbol{\sigma}} - \mathbf{R} \dot{\mathbf{R}}^T \boldsymbol{\sigma} + \boldsymbol{\sigma} \dot{\mathbf{R}} \mathbf{R}^T. \quad (3.81)$$

Jaumann Stress Rate The Jaumann stress rate is widely used because if it is zero for a given motion, the invariant of the Cauchy stress tensor is stationary during that motion [63]. However, this stress rate tensor has not a conjugate measure of finite strain associated with it. It is given by

$$\boldsymbol{\sigma}^{Jau} = \dot{\boldsymbol{\sigma}} - \mathbf{w} \boldsymbol{\sigma} + \boldsymbol{\sigma} \mathbf{w}. \quad (3.82)$$

3.4.2 Hyperelasticity

The constitutive behaviour of elastic materials depends only on the current state of deformation. When the work done by stresses during deformation is only dependent on the

initial and final configuration, the material is path-independent. This is a particular case of elasticity, called hyperelasticity. A Helmholtz potential exists for hyperelastic materials, also referred to as free-energy function. If Ψ is only a function of \mathbf{F} or other strain tensors, it is often called SEF. It represents the work done by the stresses from the initial to the current configuration and if the material is homogeneous, the SEF does not depend on the position of a point in the medium. In such cases, The time differentiation of it is given by

$$\dot{\Psi}(\mathbf{F}) = \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} : \dot{\mathbf{F}} \quad (3.83)$$

and with the Clausius-Planck inequality, it is possible to obtain the relation given by

$$\Psi(\mathbf{F}) = \int_{t_0}^{t_1} \mathbf{P} : \dot{\mathbf{F}} dt. \quad (3.84)$$

Recalling the restriction imposed by objectivity and frame indifference, Ψ must remain invariant under rigid body rotations. Thus, it only depends on \mathbf{F} via the stretch component (\mathbf{U} or \mathbf{V}) and it can be expressed as a function of \mathbf{C} or \mathbf{B} for convenience. In such case, Ψ can be expressed in a totally Lagrangian equation given by

$$\dot{\Psi}(\mathbf{C}) = \frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} : \dot{\mathbf{C}} = \frac{1}{2} \mathbf{S} : \dot{\mathbf{C}} \quad (3.85)$$

where the work conjugacy between $\dot{\mathbf{E}}$ and \mathbf{S} is used.

It is now possible to define a relation between the Piola-Kirchhoff stress tensors and the strain-energy function as

$$\begin{aligned} \mathbf{P} &= \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} = 2\mathbf{F} \frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} \\ \mathbf{S} &= 2 \frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} = \frac{\partial \Psi(\mathbf{E})}{\partial \mathbf{E}}. \end{aligned} \quad (3.86)$$

It is also possible to express the Cauchy stress tensor in terms of the strain-energy function as

$$\boldsymbol{\sigma} = J^{-1} \mathbf{F} \left(\frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} \right)^T = 2J^{-1} \mathbf{F} \left(\frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} \right) \mathbf{F}^T. \quad (3.87)$$

3.4.2.1 Isotropic Hyperelasticity

When the material's response to deformation is independent of the direction, the material is said to be isotropic. The previous constitutive equations can now be restricted to isotropic materials, implying that the relationship between Ψ and \mathbf{C} does not depend on the material axes chosen, this is, Ψ is only a function of the invariants of \mathbf{C} :

$$\Psi(\mathbf{C}) = \Psi(I_1(\mathbf{C}), I_2(\mathbf{C}), I_3(\mathbf{C})) \quad (3.88)$$

where the invariants can be defined with respect to \mathbf{C} :

$$I_1(\mathbf{C}) = \text{tr}(\mathbf{C}) \quad (3.89)$$

$$I_2(\mathbf{C}) = \frac{1}{2}(I_1^2 - \text{tr}(\mathbf{C}^2)) \quad (3.90)$$

$$I_3(\mathbf{C}) = \det(\mathbf{C}) = J^2. \quad (3.91)$$

The differentiation of the three invariants with respect to \mathbf{C} is important to obtain the constitutive equation and they are given by

$$\frac{\partial I_1(\mathbf{C})}{\partial \mathbf{C}} = \mathbf{I} \quad (3.92)$$

$$\frac{\partial I_2(\mathbf{C})}{\partial \mathbf{C}} = I_1 \mathbf{I} - \mathbf{C} \quad (3.93)$$

$$\frac{\partial I_3(\mathbf{C})}{\partial \mathbf{C}} = I_3 \mathbf{C}^{-1}. \quad (3.94)$$

After applying the chain rule differentiation and some algebraic manipulations, the constitutive equation for isotropic materials is defined as

$$\mathbf{S} = 2 \left[\left(\frac{\partial \Psi}{\partial I_1} + I_1 \frac{\partial \Psi}{\partial I_2} \right) \mathbf{I} - \frac{\partial \Psi}{\partial I_2} \mathbf{C} + I_3 \frac{\partial \Psi}{\partial I_3} \mathbf{C}^{-1} \right]. \quad (3.95)$$

It is also interesting to get the constitutive equation for these materials expressed in spatial coordinates:

$$\boldsymbol{\sigma} = 2J^{-1} \left[\left(\frac{\partial \Psi}{\partial I_1} + I_1 \frac{\partial \Psi}{\partial I_2} \right) \mathbf{B} - \frac{\partial \Psi}{\partial I_2} \mathbf{B}^2 + I_3 \frac{\partial \Psi}{\partial I_3} \mathbf{I} \right]. \quad (3.96)$$

Alternatively, the invariants can be defined with respect to the eigenvalues of \mathbf{C} which are the stretches squared (λ_i^2). In such cases, they are given by

$$\begin{aligned} I_1(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1^2 + \lambda_2^2 + \lambda_3^2 \\ I_2(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1^2 \lambda_2^2 + \lambda_1^2 \lambda_3^2 + \lambda_2^2 \lambda_3^2 \\ I_3(\lambda_1, \lambda_2, \lambda_3) &= \lambda_1^2 \lambda_2^2 \lambda_3^2 \end{aligned} \quad (3.97)$$

and the constitutive equations expressed with the Piola-Kirchhoff and Cauchy stress tensors are given by

$$\begin{aligned} \mathbf{S} &= \sum_{i=1}^3 \frac{1}{\lambda_i} \frac{\partial \Psi}{\partial \lambda_i} \hat{\mathbf{N}}_i \otimes \hat{\mathbf{N}}_i \\ \mathbf{P} &= \sum_{i=1}^3 \frac{\partial \Psi}{\partial \lambda_i} \hat{\mathbf{n}}_i \otimes \hat{\mathbf{N}}_i \\ \boldsymbol{\sigma} &= \sum_{i=1}^3 J^{-1} \lambda_i \frac{\partial \Psi}{\partial \lambda_i} \hat{\mathbf{n}}_i \otimes \hat{\mathbf{n}}_i. \end{aligned} \quad (3.98)$$

3.4.2.2 Incompressible Hyperelasticity

Some materials keep the volume nearly constant when subjected to finite strains. Therefore, they are often assumed as incompressible and only isochoric motions are possible. This is an idealisation applied in computational mechanics that facilitates the numerical treatment of these materials, which can be characterized by the incompressibility constraint $J = 1$ if assumed incompressible. For these constrained materials, the SEF is defined as

$$\Psi = \Psi(\mathbf{F}) - p(J - 1) \quad (3.99)$$

where a Lagrange multiplier p is introduced. It can be identified as a hydrostatic pressure, and its determination is possible with the equilibrium equations and the boundary conditions.

With the strain-energy function defined, it is possible to obtain the constitutive equations for these materials with the different stress measures presented:

$$\mathbf{P} = -p\mathbf{F}^T + \frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}} \quad (3.100)$$

$$\mathbf{S} = -p\mathbf{C}^{-1} + 2\frac{\partial\Psi(\mathbf{C})}{\partial\mathbf{C}} \quad (3.101)$$

$$\sigma = -p\mathbf{I} + \mathbf{F} \left(\frac{\partial\Psi(\mathbf{F})}{\partial\mathbf{F}} \right)^T. \quad (3.102)$$

3.4.2.3 Compressible Hyperelasticity

On the other hand, if the material changes its volume throughout the motion, it is said to be compressible and different constitutive equations have to be obtained. Since these materials normally have different behaviours in bulk and shear, it is useful to split the deformation into a volumetric or dilational component (related to pure volumetric deformations) and an isochoric or distortional component (related to volume-preserving deformations). To do so, a decomposition of \mathbf{F} and \mathbf{C} is performed:

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_{iso}\mathbf{F}_v & \mathbf{C} &= \mathbf{C}_{iso}\mathbf{C}_v \\ \mathbf{F}_{iso} &= J^{-1/3}\mathbf{F} & \mathbf{C}_{iso} &= J^{-2/3}\mathbf{C} \\ \mathbf{F}_v &= J^{1/3}\mathbf{I} & \mathbf{C}_v &= J^{2/3}\mathbf{I}. \end{aligned} \quad (3.103)$$

The distortional components \mathbf{F}_{iso} and \mathbf{C}_{iso} are called the modified deformation gradient and the modified right Cauchy Green tensor, respectively, and, since they are associated with volume preserving deformation, their determinant is equal to 1. When a material requires higher exterior work for dilational changes than distortional ones, the material is called nearly incompressible and the effect of compressibility is small. Following the same

reasoning, the SEF can also be decomposed as

$$\Psi(\mathbf{C}) = \Psi_v(J) + \Psi_{iso}(\mathbf{C}_{iso}) \quad (3.104)$$

where the first term describes the volumetric elastic response and the second one describes the isochoric response. The derivative of Ψ with respect to time is given by

$$\dot{\Psi} = \frac{d\Psi_v(J)}{dJ} j + \frac{\partial \Psi_{iso}(\mathbf{C}_{iso})}{\partial \mathbf{C}_{iso}} : \dot{\mathbf{C}}_{iso}. \quad (3.105)$$

To get the constitutive equation for these materials, it is first necessary to get the derivatives for the volumetric part present in eq. (3.105). They are defined as

$$\begin{aligned} \frac{\partial J^2}{\partial \mathbf{C}} &= J^2 \mathbf{C}^{-1} \\ \frac{\partial J}{\partial \mathbf{C}} &= \frac{J}{2} \mathbf{C}^{-1} \\ j &= \frac{\partial J}{\partial \mathbf{C}} : \dot{\mathbf{C}} = J \mathbf{C}^{-1} : \frac{\dot{\mathbf{C}}}{2} \end{aligned} \quad (3.106)$$

and for the isochoric part:

$$\begin{aligned} \frac{\partial J^{-2/3}}{\partial \mathbf{C}} &= -\frac{1}{3} J^{-2/3} \mathbf{C}^{-1} \\ \frac{\partial \mathbf{C}_{iso}}{\partial \mathbf{C}} &= \frac{\partial (J^{-2/3} \mathbf{C})}{\partial \mathbf{C}} = J^{-2/3} (\mathbb{I} - \frac{1}{3} \mathbf{C} \otimes \mathbf{C}^{-1}) = J^{-2/3} \mathbb{P}^T \\ \dot{\mathbf{C}}_{iso} &= 2 \left(\frac{\partial \dot{\mathbf{C}}_{iso}}{\partial \mathbf{C}} \right) : \frac{\dot{\mathbf{C}}}{2} \end{aligned} \quad (3.107)$$

where \mathbb{P} is a fourth-order tensor called the projection tensor with respect to the reference configuration and \mathbb{I} is the fourth-order identity tensor.

The constitutive relation for these materials can now be obtained as

$$\mathbf{S} = 2 \frac{\partial \Psi(\mathbf{C})}{\partial \mathbf{C}} = \mathbf{S}_v + \mathbf{S}_{iso}. \quad (3.108)$$

The volumetric part \mathbf{S}_v is given by

$$\mathbf{S}_v = 2 \frac{\partial \Psi_v(J)}{\partial \mathbf{C}} = J p \mathbf{C}^{-1} \quad (3.109)$$

where the hydrostatic pressure p is now defined with a constitutive equation instead of being calculated from boundary conditions:

$$p = \frac{d\Psi_v(J)}{dJ}. \quad (3.110)$$

The isochoric part \mathbf{S}_{iso} is given by

$$\mathbf{S}_{iso} = 2 \frac{\partial \Psi_{iso}(\mathbf{C}_{iso})}{\partial \mathbf{C}} = J^{-2/3} \mathbb{P} : \bar{\mathbf{S}} \quad (3.111)$$

where $\bar{\mathbf{S}}$ the fictitious second Piola-Kirchhoff stress tensor, defined as

$$\bar{\mathbf{S}} = 2 \frac{\partial \Psi_{iso}(\mathbf{C}_{iso})}{\partial \mathbf{C}_{iso}}. \quad (3.112)$$

Analogously and applying the same reasoning, it is possible to get a constitutive equation for the current configuration. To do so, the left Cauchy-Green tensor \mathbf{B} is also decomposed into volumetric and isochoric parts. For the sake of completeness and simplicity, only the constitutive equation is presented, which is defined as

$$\boldsymbol{\sigma} = 2J^{-1} \mathbf{B} \frac{\partial \Psi(\mathbf{B})}{\partial \mathbf{B}} = \boldsymbol{\sigma}_v + \boldsymbol{\sigma}_{iso} \quad (3.113)$$

where $\boldsymbol{\sigma}_v$ is given by:

$$\boldsymbol{\sigma}_v = 2J^{-1} \mathbf{B} \frac{\partial \Psi_v(J)}{\partial \mathbf{B}} = p \mathbf{I} \quad (3.114)$$

and $\boldsymbol{\sigma}_{iso}$ is given by

$$\boldsymbol{\sigma}_{iso} = 2J^{-1} \mathbf{B} \frac{\partial \Psi_{iso}(\mathbf{B}_{iso})}{\partial \mathbf{B}} = (\mathbb{I} - \frac{1}{3} \mathbf{I} \otimes \mathbf{I}) : \bar{\boldsymbol{\sigma}} = \mathbb{P} : \bar{\boldsymbol{\sigma}} \quad (3.115)$$

where \mathbb{P} is the projection tensor in the Eulerian description and $\bar{\boldsymbol{\sigma}}$ is the fictitious Cauchy stress tensor defined by

$$\bar{\boldsymbol{\sigma}} = 2J^{-1} \frac{\partial \Psi_{iso}(\mathbf{B}_{iso})}{\partial \mathbf{B}_{iso}} \mathbf{B}_{iso}. \quad (3.116)$$

Finally, it is also possible to get the constitutive equation for compressible hyperelastic materials as a function of the strain invariants. The modified invariants can be defined as

$$\bar{I}_1 = \text{tr}(\mathbf{C}_{iso}) = \text{tr}(\mathbf{B}_{iso}) = J^{-2/3} I_1 \quad (3.117)$$

$$\bar{I}_2 = \frac{1}{2} [(\text{tr}(\mathbf{C}_{iso}))^2 - \text{tr}(\mathbf{C}_{iso}^2)] = \frac{1}{2} [(\text{tr}(\mathbf{B}_{iso}))^2 - \text{tr}(\mathbf{B}_{iso}^2)] = J^{-4/3} I_2 \quad (3.118)$$

$$\bar{I}_3 = \det(\mathbf{C}_{iso}) = \det(\mathbf{B}_{iso}) = 1. \quad (3.119)$$

The SEF is expressed in terms of the modified strain invariants as

$$\Psi = \Psi_v(J) + \Psi_{iso}(\bar{I}_1, \bar{I}_2) \quad (3.120)$$

and the constitutive equation can easily be obtained with eq. (3.108), where \mathbf{S}_v is given

once again by eq. (3.109) and \mathbf{S}_{iso} is defined as

$$\mathbf{S}_{iso} = 2 \frac{\partial \Psi_{iso}(\bar{I}_1, \bar{I}_2)}{\partial \mathbf{C}} = J^{-2/3} \mathbb{P} : \bar{\mathbf{S}}. \quad (3.121)$$

The fictitious second Piola-Kirchhoff stress tensor is now given by

$$\bar{\mathbf{S}} = 2 \frac{\partial \Psi_{iso}(\bar{I}_1, \bar{I}_2)}{\partial \mathbf{C}_{iso}} = \bar{\gamma}_1 \mathbf{I} + \bar{\gamma}_2 \mathbf{C}_{iso} \quad (3.122)$$

where the two coefficients $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are given by

$$\begin{aligned} \bar{\gamma}_1 &= 2 \left(\frac{\partial \Psi_{iso}(\bar{I}_1, \bar{I}_2)}{\partial \bar{I}_1} + \bar{I}_1 \frac{\partial \Psi_{iso}(\bar{I}_1, \bar{I}_2)}{\partial \bar{I}_2} \right) \\ \bar{\gamma}_2 &= -2 \frac{\partial \Psi_{iso}(\bar{I}_1, \bar{I}_2)}{\partial \bar{I}_2}. \end{aligned} \quad (3.123)$$

3.4.2.4 Transversely Isotropic Hyperelasticity

Some materials are composed of a matrix material or ground substance and families of fibres. These materials are called composite materials or fibre-reinforced composites and they have high stiffness and strength, low weight and resistance to corrosion. They also have strong directional properties and their mechanical response is anisotropic.

Some soft biological tissues can be regarded as composite materials because they are composed of collagen fibres and a ground substance matrix with components such as elastin and proteoglycans.

If a composite material is reinforced with only one family of fibres, it has a single preferred direction and the stiffness in the fibre direction is much higher than the one in the orthogonal directions to the fibres. In this case, the material is called transversely isotropic with respect to the fibre direction and the material response along the directions orthogonal to the preferred direction is isotropic.

For a transversely isotropic material, the stress is dependent on the deformation gradient \mathbf{F} and on the direction of the fibres, which is defined by a unit vector in the reference configuration $\mathbf{a}_0(\mathbf{X})$. After deformation, the fibre changes its direction and is then defined by a unit vector in the deformed configuration $\mathbf{a}(\mathbf{x}, t)$. With the definitions provided before in Section 3.1.4, it is possible to relate the fibre directions in the reference and current configurations as

$$\lambda \mathbf{a} = \mathbf{F} \mathbf{a}_0. \quad (3.124)$$

The SEF for transversely isotropic hyperelastic materials depends on \mathbf{C} and on the fibre direction \mathbf{a}_0 , which is immaterial. Thus, the SEF can be written as

$$\Psi = \Psi(\mathbf{C}, \mathbf{a}_0 \otimes \mathbf{a}_0). \quad (3.125)$$

As referred previously, the SEF must be frame-indifferent and to satisfy this condition, two additional invariants are introduced for transversely isotropic materials. Therefore, these materials can be represented by the three invariants used for isotropic materials and two additional ones called pseudo-invariants of \mathbf{C} and $\mathbf{a}_0 \otimes \mathbf{a}_0$ defined as

$$\begin{aligned} I_4(\mathbf{C}, \mathbf{a}_0) &= \mathbf{a}_0 \cdot (\mathbf{C}\mathbf{a}_0) = \lambda^2 \\ I_5(\mathbf{C}, \mathbf{a}_0) &= \mathbf{a}_0 \cdot (\mathbf{C}^2\mathbf{a}_0). \end{aligned} \quad (3.126)$$

Therefore, the SEF for a transversely isotropic material can be written in terms of the five invariants as

$$\Psi = \Psi[I_1(\mathbf{C}), I_2(\mathbf{C}), I_3(\mathbf{C}), I_4(\mathbf{C}, \mathbf{a}_0), I_5(\mathbf{C}, \mathbf{a}_0)]. \quad (3.127)$$

The second Piola-Kirchhoff stress tensor for these materials can be obtained as

$$\mathbf{S} = 2 \frac{\partial \Psi(\mathbf{C}, \mathbf{a}_0 \otimes \mathbf{a}_0)}{\partial \mathbf{C}} = 2 \sum_{i=1}^5 \frac{\partial \Psi(\mathbf{C}, \mathbf{a}_0 \otimes \mathbf{a}_0)}{\partial I_i} \frac{\partial I_i}{\partial \mathbf{C}}. \quad (3.128)$$

The derivatives of the first three invariants with respect to \mathbf{C} are given in eq. (3.92) and for the pseudo-invariants they are defined as

$$\begin{aligned} \frac{\partial I_4}{\partial \mathbf{C}} &= \mathbf{a}_0 \otimes \mathbf{a}_0 \\ \frac{\partial I_5}{\partial \mathbf{C}} &= \mathbf{a}_0 \otimes \mathbf{C}\mathbf{a}_0 + \mathbf{a}_0\mathbf{C} \otimes \mathbf{a}_0. \end{aligned} \quad (3.129)$$

Therefore, the second Piola-Kirchhoff stress tensor is given by

$$\begin{aligned} \mathbf{S} &= 2 \left[\left(\frac{\partial \Psi}{\partial I_1} + I_1 \frac{\partial \Psi}{\partial I_2} \right) \mathbf{I} - \frac{\partial \Psi}{\partial I_2} \mathbf{C} + I_3 \frac{\partial \Psi}{\partial I_3} \mathbf{C}^{-1} \right. \\ &\quad \left. + \frac{\partial \Psi}{\partial I_4} \mathbf{a}_0 \otimes \mathbf{a}_0 + \frac{\partial \Psi}{\partial I_5} (\mathbf{a}_0 \otimes \mathbf{C}\mathbf{a}_0 + \mathbf{a}_0\mathbf{C} \otimes \mathbf{a}_0) \right]. \end{aligned} \quad (3.130)$$

An expression for $\boldsymbol{\sigma}$ can be obtained with a push forward operation on \mathbf{S} :

$$\begin{aligned} \boldsymbol{\sigma} &= 2J^{-1} \left[I_3 \frac{\partial \Psi}{\partial I_3} \mathbf{I} + \left(\frac{\partial \Psi}{\partial I_1} + I_1 \frac{\partial \Psi}{\partial I_2} \right) \mathbf{B} - \frac{\partial \Psi}{\partial I_2} \mathbf{B}^2 \right. \\ &\quad \left. + I_4 \frac{\partial \Psi}{\partial I_4} \mathbf{a} \otimes \mathbf{a} + I_4 \frac{\partial \Psi}{\partial I_5} (\mathbf{a} \otimes \mathbf{B}\mathbf{a} + \mathbf{a}\mathbf{B} \otimes \mathbf{a}) \right]. \end{aligned} \quad (3.131)$$

In the case of transversely isotropic materials with an incompressible isotropic matrix material, the embedded fibres can be extensible or inextensible. For both cases, the isotropic matrix material is incompressible, this is $I_3=1$, and Ψ is a function of the other four independent invariants. Due to the incompressibility constraint, it is also necessary to introduce an indeterminate Lagrange multiplier $p/2$, which is identified as a reaction

pressure. Thus, the SEF can be written as

$$\Psi = \Psi[I_1(\mathbf{C}), I_2(\mathbf{C}), I_4(\mathbf{C}, \mathbf{a}_0), I_5(\mathbf{C}, \mathbf{a}_0)] - \frac{1}{2}p(I_3 - 1). \quad (3.132)$$

If the embedded fibres are inextensible, the fourth invariant and the stretch are also equal to one ($\lambda = I_4 = 1$). Therefore, Ψ is a function of I_1 and I_2 , which are related to the isotropic matrix material, and I_5 , which is related to the fibres. Due to the additional constraint, another Lagrange multiplier $q/2$ is added, which is defined as

$$\Psi = \Psi[I_1(\mathbf{C}), I_2(\mathbf{C}), I_5(\mathbf{C}, \mathbf{a}_0)] - \frac{1}{2}p(I_3 - 1) - \frac{1}{2}q(I_4 - 1). \quad (3.133)$$

The constitutive equation in the material is given by

$$\begin{aligned} \mathbf{S} = & -p\mathbf{C}^{-1} - q\mathbf{a}_0 \otimes \mathbf{a}_0 + 2 \left(\frac{\partial \Psi}{\partial I_1} + I_1 \frac{\partial \Psi}{\partial I_2} \right) \mathbf{I} - 2 \frac{\partial \Psi}{\partial I_2} \mathbf{C} \\ & + 2 \frac{\partial \Psi}{\partial I_5} (\mathbf{a}_0 \otimes \mathbf{C}\mathbf{a}_0 + \mathbf{a}_0 \mathbf{C} \otimes \mathbf{a}_0) \end{aligned} \quad (3.134)$$

and in spatial descriptions by

$$\boldsymbol{\sigma} = -p\mathbf{I} - q\mathbf{a} \otimes \mathbf{a} + 2 \frac{\partial \Psi}{\partial I_1} \mathbf{B} - 2 \frac{\partial \Psi}{\partial I_2} \mathbf{B}^{-1} + 2 \frac{\partial \Psi}{\partial I_5} (\mathbf{a} \otimes \mathbf{B}\mathbf{a} + \mathbf{a}\mathbf{B} \otimes \mathbf{a}). \quad (3.135)$$

In some materials, two families of fibres can be distinguished, which means that there are two distinct preferred directions in the reference configuration, \mathbf{a}_0 and \mathbf{a}_0' . In such case, three additional invariants are introduced: I_6 and I_7 , analogous to I_4 and I_5 and I_8 and I_9 , which couple the two families of fibres. They are defined as

$$\begin{aligned} I_6(\mathbf{C}, \mathbf{a}_0') &= \mathbf{a}_0' \cdot (\mathbf{C}\mathbf{a}_0') \\ I_7(\mathbf{C}, \mathbf{a}_0') &= \mathbf{a}_0' \cdot (\mathbf{C}^2 \mathbf{a}_0') \\ I_8(\mathbf{C}, \mathbf{a}_0, \mathbf{a}_0') &= (\mathbf{a}_0 \cdot \mathbf{a}_0') \mathbf{a}_0 \cdot \mathbf{C}\mathbf{a}_0' \\ I_9(\mathbf{a}_0, \mathbf{a}_0') &= (\mathbf{a}_0 \cdot \mathbf{a}_0')^2. \end{aligned} \quad (3.136)$$

The invariant I_9 is the square of the dot product between \mathbf{a}_0 and \mathbf{a}_0' , which determines the cosine of the angle between the two fibre directions in the undeformed configuration. Thus, I_9 is independent of deformation and can be disregarded. The derivatives of the introduced invariants are given by

$$\begin{aligned} \frac{\partial I_6}{\partial \mathbf{C}} &= \mathbf{a}_0' \otimes \mathbf{a}_0' \\ \frac{\partial I_7}{\partial \mathbf{C}} &= \mathbf{a}_0' \otimes \mathbf{C}\mathbf{a}_0' + \mathbf{a}_0' \mathbf{C} \otimes \mathbf{a}_0' \\ \frac{\partial I_8}{\partial \mathbf{C}} &= \frac{1}{2} (\mathbf{a}_0 \cdot \mathbf{a}_0') (\mathbf{a}_0 \otimes \mathbf{a}_0' + \mathbf{a}_0' \otimes \mathbf{a}_0). \end{aligned} \quad (3.137)$$

The constitutive equations presented before can simply be adapted and extended for the case of two families of fibres.

3.4.3 Strain Energy Functions

As it was demonstrated before, the constitutive equations for hyperelastic materials are obtained from the SEF. Therefore, it is essential to define it properly according to the material in analysis, which led to the proposal of several SEFs in the literature. In this section, some of these functions are going to be presented, particularly the ones that are well-trying within the constitutive theory of finite elasticity and are more frequently employed.

3.4.3.1 Volumetric Contribution

In computational analysis, incompressible materials are usually considered nearly incompressible to avoid numerical problems related to incompressibility and a penalty method is used. In such cases, the SEF is given in a decoupled way:

$$\Psi(\mathbf{C}) = \Psi_v(J) + \Psi_{iso}(\mathbf{C}_{iso}) \quad (3.138)$$

where the volumetric part is characterized by a penalty parameter $1/D_1$ that can be viewed as the bulk modulus. This parameter is independent of deformation and a large value is used to model incompressible materials as slightly compressible ones:

$$\Psi_v(J) = \frac{1}{D_1}(J - 1)^2. \quad (3.139)$$

The meaning of Ψ_v in eq. (3.138) for nearly incompressible materials is different from the one presented in eq. (3.104) for compressible materials, where Ψ_v has physical relevance. Alternative expressions to eq. (3.139) can be found in the literature, but all of them ensure the same: working with a slightly compressible material to overcome potential numerical problems of incompressible materials

3.4.3.2 Isotropic Materials

Mooney Model In 1940, Mooney [64] proposed a SEF derived on the basis of mathematical arguments and employing considerations of symmetry:

$$\Psi = C_{10}(I_1 - 3) + C_{01}(I_2 - 3) \quad (3.140)$$

where C_{10} and C_{01} are material constants.

Mooney-Rivlin Model In 1948, Rivlin and Rideal [65] extended the previous model by considering the SEF as a polynomial series of $(I_1 - 3)$ and $(I_2 - 3)$ as

$$\Psi = \sum_{i=0, j=0}^{\infty} C_{ij} (I_1 - 3)^i (I_2 - 3)^j \quad (3.141)$$

where C_{ij} are, once again, material parameters, with $C_{00} = 0$. The series is usually truncated to terms of the second or third order since higher order requires the determination of many material parameters. This form of strain energy is classically used for large strain problems [66].

Ogden Model In 1972, Ogden and Hill [67] proposed the so-called Ogden model, which was developed to obtain an adequate representation of the non-linear mechanical response of rubberlike materials. It is expressed as a function of the principal stretches and of some constants that are material-specific parameters and are typically determined by fitting the model to experimental data:

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \sum_{p=1}^N \frac{\mu_p}{\alpha_p} (\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_3^{\alpha_p} - 3) \quad (3.142)$$

where N is a positive integer that determines the number of terms in the strain-energy function, α_p are dimensionless constants and μ_p are constant shear moduli. The classical shear modulus in the reference configuration μ , known from the linear theory, can be obtained as

$$2\mu = \sum_{p=1}^N \mu_p \alpha_p. \quad (3.143)$$

This model is important for the theory of finite elasticity and to model materials that undergo finite strains in relation to an equilibrium state, such as biomaterials. In addition, Ogden showed that a 6 parameters model ($N=3$) had an excellent agreement with simple tension, pure shear and equi-biaxial tension data from experiments performed by Treloar [68] in vulcanized rubber over a very large strain range.

Neo-Hookean Model All the 3 previous models are phenomenological models that are issued from mathematical developments of the SEF. In this type of models, the material parameters are usually hard to obtain and they can lead to inaccurate results when used outside the deformation range in which their parameters were identified [66]. An alternative is the so-called physically-based models. Their main drawbacks are the mathematical complications that can arise in their formulation.

One of the most widely used and known model of this type is the Neo-Hookean model proposed by Treloar [15]. It is simple and a reliable way to describe the non-linear deformation behaviour of isotropic rubber-like materials. Treloar derived it from molecular chain statistics and used a Gaussian statistical distribution to obtain the SEF defined as

$$\Psi = \frac{1}{2}nkT(I_1 - 3) = C_{10}(I_1 - 3) \quad (3.144)$$

where n is the chain density per unit of volume, k is the Boltzmann constant and T is the absolute temperature. This model matches the Mooney-Rivlin model with only one material parameter ($C_{01} = 0$) and it can also be derived from the Ogden model by setting $N=1$ and $\alpha_1 = 2$:

$$\Psi = \frac{\mu_1}{2}(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 3). \quad (3.145)$$

3.4.3.3 Transversely Isotropic Materials

Considering the theory presented before in Section 3.4.2.4, the isochoric part of Ψ can be separated into different components: one that accounts for the isotropic contribution of the ground matrix (Ψ_{mat}) and another that accounts for anisotropy of the embedded family of fibres (Ψ_{fib}):

$$\Psi = \Psi_{fib} + \Psi_{mat} + \Psi_v. \quad (3.146)$$

Humphrey and Yin Model Humphrey and Yin [69] proposed in 1987 a constitutive model to characterize the passive mechanical response of the cardiac tissue and it was the first anisotropic invariant-based model that accounts for fibre structure. The SEF of this model is the sum of two exponentials, one in I_1 and one in I_4 , and it is given

$$\Psi = c\{\exp[b(I_1 - 3)] - 1\} + A\{\exp[a(\sqrt{I_4} - 1)^2] - 1\} \quad (3.147)$$

where c , b , A and a are material parameters.

Martins Model Martins et al. [70] proposed and developed a model for the simulation of skeletal muscles. It is a generalized version of the model proposed by Humphrey and Yin [69], where the SEF is divided into three different contributions, as it is given by eq. (3.146).

The volumetric part Ψ_v is defined with eq. (3.139) and the contribution of the isotropic embedding matrix is given by

$$\Psi_{mat} = c\{\exp[b(\bar{I}_1 - 3)] - 1\} \quad (3.148)$$

where c and b are material constants.

The contribution of the anisotropy of the embedded family of fibres can be divided into a passive elastic part (Ψ_{fib}^{PE}) and an active one related to the muscle contraction (Ψ_{fib}^{SE}).

The passive component is given by

$$\Psi_{fib}^{PE}(\bar{\lambda}_f) = T_0^M \int_1^{\bar{\lambda}_f} f_{PE}(\lambda) d\lambda \quad (3.149)$$

and the active component is defined as

$$\Psi_{fib}^{SE}(\bar{\lambda}_f, \zeta^{CE}) = T_0^M \int_1^{\bar{\lambda}_f} f_{SE}(\lambda, \zeta^{CE}) d\lambda. \quad (3.150)$$

In the above equations, $\bar{\lambda}_f$ is the isochoric fibre stretch ratio in the direction of the undeformed fibre, ζ^{CE} is a non-dimensional quantity proportional to the strain of the contractile element and T_0^M is the muscle peak stress defined as

$$T_0^M = \frac{F_0^M}{A_0} \quad (3.151)$$

where A_0 is the physiological cross-section area and F_0^M is the peak isometric muscle force.

The function $f_{PE}(\lambda)$ is given by

$$f_{PE}(\bar{\lambda}_f) = \begin{cases} 4(\lambda^M - 1)^2, & \text{if } \lambda^M > 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.152)$$

where λ^M is the muscle stretch, given by the ratio between the muscle length and the rest length of the muscle, whereas $f_{SE}(\lambda)$ is defined as

$$f_{SE}(\bar{\lambda}_f, \zeta^{CE}) = \begin{cases} 0.1\{\exp[100(\lambda^M - 1 - \zeta^{CE})] - 1\} & , \text{if } \lambda^M > 1 + \zeta^{CE} \\ 0 & , \text{otherwise} \end{cases}. \quad (3.153)$$

Holzappel-Gasser-Ogden Model Holzappel et al. [71] proposed a model to describe the passive mechanical response of arterial tissue. This model is commonly referred to as Holzappel-Gasser-Ogden (HGO) model and separates the strain-energy function into two parts, as the previous two models. Taking the example of a nearly incompressible material with two families of fibres, such as the arterial walls, the SEF is expressed as

$$\Psi(\mathbf{C}_{iso}, \mathbf{a}_0 \otimes \mathbf{a}_0, \mathbf{a}_0' \otimes \mathbf{a}_0') = \Psi_{mat}(\mathbf{C}_{iso}) + \Psi_{fib}(\mathbf{C}_{iso}, \mathbf{a}_0 \otimes \mathbf{a}_0, \mathbf{a}_0' \otimes \mathbf{a}_0') + \Psi_v. \quad (3.154)$$

Alternatively, for an incompressible matrix material reinforced with two families of fibres, eq. (3.154) can be defined in terms of invariants as

$$\Psi(\bar{I}_1, \bar{I}_2, \bar{I}_4, \bar{I}_5, \bar{I}_6, \bar{I}_7, \bar{I}_8) = \Psi_{mat}(\bar{I}_1, \bar{I}_2) + \Psi_{fib}(\bar{I}_1, \bar{I}_2, \bar{I}_4, \bar{I}_5, \bar{I}_6, \bar{I}_7, \bar{I}_8). \quad (3.155)$$

A simple way to capture the isotropy of the ground substance is through \bar{I}_1 and the transverse isotropy associated with the two families of fibres can be captured with \bar{I}_4 and

\bar{I}_6 . Thus, eq. (3.155) reduces to

$$\Psi(\bar{I}_1, \bar{I}_4, \bar{I}_6) = \Psi_{mat}(\bar{I}_1) + \Psi_{fib}(\bar{I}_4, \bar{I}_6). \quad (3.156)$$

The ground substance can be modelled, for example, with the Neo-Hookean material, depending on the type of material of the ground substance. Regarding the family of fibres, Ψ_{fib} is defined with an exponential relation given by

$$\Psi_{fib}(\bar{I}_4, \bar{I}_6) = \frac{k_1}{2k_2} \sum_{i=4,6} \{ \exp [k_2(\bar{I}_i - 1)^2] - 1 \} \quad (3.157)$$

where $k_1 > 0$ is a stress-like material parameter and $k_2 > 0$ is a dimensionless parameter.

In 2006, this model was further developed by Gasser et al. [72] to account for the orientation and distribution of the family of fibres. To do so, a symmetric generalized structure tensor \mathbf{H} is defined as

$$\mathbf{H} = \frac{1}{4\pi} \int_w \rho_f(\mathbf{M}) \mathbf{M} \otimes \mathbf{M} \, dw \quad (3.158)$$

where \mathbf{M} is an arbitrary unit vector located in three-dimensional Eulerian space and expressed in terms of two Eulerian angles ($\Theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$) and ρ_f is the orientation distribution density function normalized according to

$$\frac{1}{4\pi} \int_w \rho_f(\mathbf{M}(\Theta, \phi)) \, dw = 1. \quad (3.159)$$

The generalized structure tensor is the mean of the structure tensor $\mathbf{a}_0 \otimes \mathbf{a}_0$ over the unit sphere w weighted by the orientation distribution density ρ_f . In general, this tensor has six independent components but for a transversely isotropic distribution, it involves only a single constant and it is given by

$$\mathbf{H} = \kappa \mathbf{I} + (1 - 3\kappa) \mathbf{a}_0 \otimes \mathbf{a}_0 \quad (3.160)$$

where κ is the radial dispersion parameter given by

$$\kappa = \frac{1}{4} \int_0^\pi \rho_f(\Theta) \sin^3 \Theta \, d\Theta. \quad (3.161)$$

The SEF of the two families of fibres is now given by

$$\Psi(\mathbf{C}_{iso}, \mathbf{H}_i) = \frac{k_1}{2k_2} \left[\exp \left(k_2 \bar{E}_i^2 \right) - 1 \right], \quad i = 4, 6 \quad (3.162)$$

where \bar{E}_i is an invariant of \mathbf{C}_{iso} and \mathbf{H} given by

$$\bar{E}_i = \mathbf{H}_i : \mathbf{C}_{iso} - 1 = \kappa \bar{I}_1 + (1 - 3\kappa) \bar{I}_i - 1, \quad i = 4, 6. \quad (3.163)$$

The radial dispersion parameter κ can range from 0 to 1/3. In the lower limit, the fibres are perfectly aligned, without dispersion and it coincides with the HGO model proposed in 2000. In the upper limit, the fibres describe the isotropic distribution of the collagen fibres.

An important hypothesis introduced in this model is that the collagen fibres do not support compressive loads since they would buckle in that situation. Thus, it is assumed that the fibres contribute to the SEF only in extension, which means that if there are compressive loads, the anisotropic part of the strain energy is zero and the material is assumed as purely isotropic.

3.4.4 Elasticity Tensors

The relationship between stress and strain for hyperelastic materials is nonlinear, and to obtain a solution to problems with these materials it is often necessary to use an incremental/iterative solution technique such as the Newton-Raphson method. To apply such techniques, it is necessary to linearize the constitutive equation with respect to an increment \mathbf{u} . With the chain rule, it is possible to obtain a linear relationship between the directional derivative of \mathbf{S} and \mathbf{E} , given by

$$\begin{aligned} D_{\mathbf{u}}S_{IJ}(\mathbf{X}) &= \frac{d}{d\epsilon} S_{IJ}[E_{KL}(\mathbf{X} + \epsilon\mathbf{u})]|_{\epsilon=0} = \sum_{K,L=1}^3 \frac{\partial S_{IJ}}{\partial E_{KL}} \frac{d}{d\epsilon} E_{KL}(\mathbf{X} + \epsilon\mathbf{u})|_{\epsilon=0} \\ &= \sum_{K,L=1}^3 \frac{\partial S_{IJ}}{\partial E_{KL}} D_{\mathbf{u}}E_{KL}(\mathbf{X}). \end{aligned} \quad (3.164)$$

It is possible to express eq. (3.164) more concisely with the introduction of the symmetric fourth-order tensor \mathbb{C} , known as Lagrangian or material elasticity tensor, given by

$$D_{\mathbf{u}}\mathbf{S} = \mathbb{C} : D_{\mathbf{u}}\mathbf{E} \quad (3.165)$$

where \mathbb{C} is defined as

$$\begin{aligned} C_{ijkl} &= \frac{\partial S_{ij}}{\partial E_{kl}} = 2 \frac{\partial S_{ij}}{\partial C_{kl}} = 4 \frac{\partial^2 \Psi}{\partial C_{ij} \partial C_{kl}} \\ \mathbb{C} &= \frac{\partial \mathbf{S}}{\partial \mathbf{E}} = 2 \frac{\partial \mathbf{S}}{\partial \mathbf{C}} = 4 \frac{\partial^2 \Psi}{\partial \mathbf{C} \partial \mathbf{C}}. \end{aligned} \quad (3.166)$$

The material elasticity tensor relates the work conjugate pairs of stress and strain tensors and measures the change in stress that results from changes in strain. Generally, \mathbb{C} has minor symmetries ($C_{IJKL} = C_{JIKL} = C_{IJLK}$). For hyperelastic materials, \mathbb{C} can be derived from Ψ and in such case it is said that \mathbb{C} possesses major symmetries, which means that it has only 21 independent components at each strain state ($C_{IJKL} = C_{KLIJ}$). The condition of major symmetries is a necessary and sufficient condition for a material to be hyperelastic.

To obtain the elasticity tensor in spatial coordinates \mathbf{c} , it is necessary to do a push-forward operation of \mathbb{C} . This tensor is called spatial or Eulerian elasticity tensor and is given by

$$c_{ijkl} = J^{-1} F_{iI} F_{jJ} F_{kK} F_{lL} C_{IJKL}. \quad (3.167)$$

The same reasoning about the symmetries is applied to the spatial elasticity tensor.

It is also possible to get a decoupled representation of the elasticity tensor, where it is split into volumetric and isochoric components:

$$\mathbb{C} = \mathbb{C}_v + \mathbb{C}_{iso}. \quad (3.168)$$

The volumetric part \mathbb{C}_v is given by

$$\mathbb{C}_v = J \tilde{p} \mathbf{C}^{-1} \otimes \mathbf{C}^{-1} - 2J \tilde{p} \mathbf{C}^{-1} \odot \mathbf{C}^{-1} \quad (3.169)$$

where the symbol \odot denotes a tensor product defined as

$$(\mathbf{C}^{-1} \odot \mathbf{C}^{-1})_{ijkl} = \frac{1}{2} (C_{ik}^{-1} C_{jl}^{-1} + C_{il}^{-1} C_{jk}^{-1}) \quad (3.170)$$

and \tilde{p} is a scalar function defined as

$$\tilde{p} = p + J \frac{dp}{dJ}. \quad (3.171)$$

The isochoric part \mathbb{C}_{iso} is given by

$$\mathbb{C}_{iso} = \mathbb{P} : \bar{\mathbb{C}} : \mathbb{P}^T + \frac{2}{3} \text{Tr}(J^{-2/3} \bar{\mathbb{S}}) \tilde{\mathbb{P}} - \frac{2}{3} (\mathbf{C}^{-1} \otimes \mathbf{S}_{iso}) \quad (3.172)$$

where $\bar{\mathbb{C}}$, $\tilde{\mathbb{P}}$ and $\text{Tr}(\bullet)$ are the fictitious elasticity tensor in the material description, the modified projection tensor of fourth order and a special operator, respectively:

$$\begin{aligned} \bar{\mathbb{C}} &= 2J^{-4/3} \frac{\partial \bar{\mathbb{S}}}{\partial \mathbf{C}_{iso}} = 4J^{-4/3} \frac{\partial^2 \Psi_{iso}(\mathbf{C}_{iso})}{\partial \mathbf{C}_{iso} \partial \mathbf{C}_{iso}} \\ \tilde{\mathbb{P}} &= \mathbf{C}^{-1} \odot \mathbf{C}^{-1} - \frac{1}{3} \mathbf{C}^{-1} \otimes \mathbf{C}^{-1} \\ \text{Tr}(\bullet) &= (\bullet) : \mathbf{C}. \end{aligned} \quad (3.173)$$

The FE software Abaqus uses a co-rotational frame based on the Jaumann rate for solid elements and to implement a constitutive model in a UMAT, it is necessary to provide a Jacobian matrix (DDSDDE) as mentioned before. In order to update this matrix, the model must be expressed in terms of the Jaumann rate of the Kirchhoff stress tensor. Therefore, the Jacobian matrix does not correspond directly to the spatial elasticity tensor

\mathbb{C} and is in fact given by

$$\text{DDSDDE}_{ijkl} = \mathbb{C}_{ijkl} + \frac{1}{2}(I_{ac}\sigma_{bd} + \sigma_{ac}I_{bd} + I_{ad}\sigma_{bc} + \sigma_{ad}I_{bc}). \quad (3.174)$$

3.5 Finite Element Method

3.5.1 Principle of Virtual Work

Variational principles, such as the principle of virtual work, are powerful tools to evaluate continuous bodies and are fundamental in mathematics and mechanics. The FE method does not necessarily depend on the existence of a variational principle, but a good approximation is often related to the weak forms of field equations, which can be obtained with variational principles.

The principle of virtual work is one of these variational principles that is essential for FE formulations. To introduce it, a deformable body occupying the region Ω and the following boundary conditions are considered:

$$\begin{aligned} \mathbf{u} &= \bar{\mathbf{u}} \quad \text{on } \Omega_u \\ \mathbf{t} &= \bar{\mathbf{t}} \quad \text{on } \Omega_t \end{aligned} \quad (3.175)$$

where $\bar{\mathbf{u}}$ is a prescribed displacement on a surface of the body denoted by $\partial\Omega_u$, which is a Dirichlet or essential boundary condition and $\bar{\mathbf{t}}$ is a prescribed Cauchy traction vector applied in the rest of the surface boundary denoted by $\partial\Omega_t$, which is a Neumann or natural boundary condition.

Considering a time-independent problem and one of the most important balance laws, known as Cauchy's first equation of motion and given by eq. (3.60), it is possible to formulate a strong form of a nonlinear boundary-value problem, given by

$$\begin{cases} \text{div}_x \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0} & , \\ \mathbf{u} = \bar{\mathbf{u}} & , \text{ on } \partial\Omega_u \\ \mathbf{t} = \bar{\mathbf{t}} & , \text{ on } \partial\Omega_t \end{cases} \quad (3.176)$$

To develop the principle of virtual work, a vector-valued function $\boldsymbol{\eta}(\mathbf{x})$ is defined in the current configuration and the integration the Cauchy's first equation of motion over Ω is performed:

$$f(\mathbf{u}, \boldsymbol{\eta}) = \int_{\Omega} (\text{div}_x \boldsymbol{\sigma} + \mathbf{b}) \cdot \boldsymbol{\eta} \, dv = 0. \quad (3.177)$$

The function $\boldsymbol{\eta}$ is smooth with $\boldsymbol{\eta} = 0$ on $\partial\Omega_u$ and with the divergence theorem, it is possible to obtain the weak form of the boundary-value problem, given by

$$f(\mathbf{u}, \boldsymbol{\eta}) = \int_{\Omega} (\boldsymbol{\sigma} : \nabla \boldsymbol{\eta} - \mathbf{b} \cdot \boldsymbol{\eta}) \, dv - \int_{\partial\Omega_t} \bar{\mathbf{t}} \cdot \boldsymbol{\eta} \, ds = 0. \quad (3.178)$$

The introduced function $\boldsymbol{\eta}$ is arbitrary, but if it is taken as a virtual displacement field $\delta\mathbf{u}$ on the current configuration, eq. (3.178) leads to the principle of virtual work in the spatial description, defined as

$$f(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} (\boldsymbol{\sigma} : \delta\mathbf{e} - \mathbf{b} \cdot \delta\mathbf{u}) \, dv - \int_{\partial\Omega_t} \bar{\mathbf{t}} \cdot \delta\mathbf{u} \, ds = 0. \quad (3.179)$$

The principle of virtual work states that the internal virtual work is the same as the external virtual work for any kinematically admissible virtual displacement field. The stress $\boldsymbol{\sigma}$ does internal work along the virtual strain $\delta\mathbf{e}$, whereas the body force \mathbf{b} and surface traction $\bar{\mathbf{t}}$ do external work along the virtual displacement $\delta\mathbf{u}$. If the accelerations are null, such as is the case for static problems, the external work δW_{ext} is equal to the internal work δW_{int} :

$$\delta W_{int}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} \boldsymbol{\sigma} : \delta\mathbf{e} \, dv \quad (3.180)$$

$$\delta W_{ext}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} \mathbf{b} \cdot \delta\mathbf{u} \, dv + \int_{\partial\Omega} \bar{\mathbf{t}} \cdot \delta\mathbf{u} \, ds \quad (3.181)$$

$$\delta W_{ext} = \delta W_{int} \quad , \text{ if } \ddot{\mathbf{u}}=0. \quad (3.182)$$

The principle of virtual work states the equilibrium of a deformable body and is the basis for FE discretization. It is sometimes convenient to express this principle in material description, defined as

$$\mathcal{F}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega_0} (\mathbf{S} : \delta\mathbf{E} - \mathbf{B}_f \cdot \delta\mathbf{u}) \, dV - \int_{\partial\Omega_{0t}} \bar{\mathbf{T}} \cdot \delta\mathbf{u} \, dS = 0. \quad (3.183)$$

3.5.2 Linearization of the Principle of Virtual Work

The principle of virtual work is generally nonlinear in the unknown displacement \mathbf{u} . The non-linearities can come from the geometry, the material, the loads and the boundary conditions and the exact solution to these problems is only available for very simple problems.

A common technique to solve nonlinear equations is to use Newton-Raphson, which requires a consistent linearization of all quantities involved in the problem.

The linearization based on the first-order Taylor's expansion of a given nonlinear and smooth function $\mathcal{F}(\mathbf{u})$ is given by

$$\mathcal{F}(\mathbf{u}, \Delta\mathbf{u}) = \mathcal{F}(\mathbf{u}) + \Delta\mathcal{F}(\mathbf{u}, \Delta\mathbf{u}) + o(\Delta\mathbf{u}) \quad (3.184)$$

where $\Delta(\mathbf{u})$ denotes the increment of the displacement field and the remainder $o(\Delta\mathbf{u})$ is a small error that tends to zero faster than $\Delta\mathbf{u}$.

In the Newton-Raphson method, eq. (3.184) is truncated after the first derivative of \mathcal{F} . Thus, the first term is an approximate solution for a given state \mathbf{u} and the second term is the linearization of \mathcal{F} at \mathbf{u} . The linear change in \mathcal{F} due to an increment $\Delta\mathbf{u}$ at \mathbf{u} is equal to the directional derivative of \mathcal{F} at a given \mathbf{u} in the direction of the incremental displacement field $\Delta\mathbf{u}$:

$$\Delta\mathcal{F}(\mathbf{u}, \Delta\mathbf{u}) = D_{\Delta\mathbf{u}}\mathcal{F}(\mathbf{u}) = \frac{d}{d\epsilon}\mathcal{F}(\mathbf{u} + \epsilon\Delta\mathbf{u})|_{\epsilon=0}. \quad (3.185)$$

Now that the concept of linearization is introduced, it is possible to linearize the principle of virtual work in the spatial or material description. The linearization of external virtual work vanishes ($D_{\Delta\mathbf{u}}\delta W_{ext}(\mathbf{u}, \delta\mathbf{u}) = 0$) if it is considered a purely static problem ($\ddot{\mathbf{u}} = \mathbf{0}$) and that \mathbf{B}_f and $\bar{\mathbf{t}}$ are independent of the motion of the body. Thus, only the internal virtual work is affected by linearization. Considering the expression for internal virtual work from eq. (3.183), it is possible to obtain its linearization as

$$D_{\Delta\mathbf{u}}\delta W_{int}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega_0} [\mathbf{S} : D_{\Delta\mathbf{u}}\delta\mathbf{E} + \delta\mathbf{E} : D_{\Delta\mathbf{u}}\mathbf{S}] dV \quad (3.186)$$

where the linearization of S is given by

$$D_{\Delta\mathbf{u}}\mathbf{S} = \frac{\partial\mathbf{S}}{\partial\mathbf{E}} : D_{\Delta\mathbf{u}}\mathbf{E} = \mathbb{C} : D_{\Delta\mathbf{u}}\mathbf{E}. \quad (3.187)$$

After some manipulation, it is possible to obtain the final expression for the linearization of the internal virtual work in the material description, or the so-called total-Lagrangian formulation, given by

$$D_{\Delta\mathbf{u}}\delta W_{int}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega_0} (\nabla_X\delta\mathbf{u} : \nabla_X\Delta\mathbf{u} \mathbf{S} + \mathbf{F}^T\nabla_X\delta\mathbf{u} : \mathbb{C} : \mathbf{F}^T\nabla_X\Delta\mathbf{u})dV. \quad (3.188)$$

The first term in eq. (3.188) comes from the current state of stress and represents the initial stress contribution to the linearization. It can be seen as the initial stress at every increment in an iterative solution technique, whereas the second term is the constitutive contribution to the linearization.

Alternatively, the principle of virtual work can also be linearized in the spatial description. The same assumptions used for the material description are considered and the idea is to do a push forward to the linearized terms from eq. (3.186). After some manipulations and rearranging, the linearized internal virtual work in the spatial description, or the so-called updated-Lagrangian formulation, is given by

$$D_{\Delta\mathbf{u}}\delta W_{int}(\mathbf{u}, \delta\mathbf{u}) = \int_{\Omega} (\nabla_x\delta\mathbf{u} : \nabla_x\Delta\mathbf{u} \boldsymbol{\sigma} + \nabla_x\delta\mathbf{u} : \mathbb{c} : \nabla_x\Delta\mathbf{u})dv. \quad (3.189)$$

The linearized principle of virtual work is the starting point for approximation techniques such as the FE method.

Chapter 4

Hyperelastic Surrogate Modelling

In this chapter, the process to get surrogate models to predict the isochoric part of $\boldsymbol{\sigma}$ and of \mathbf{c} expeditiously from \mathbf{F} is going to be explained. Figure 4.1 shows the architecture of the models used, where the input is the deformation gradient and the outputs are $\boldsymbol{\sigma}$ and \mathbf{c} . The surrogate models were obtained with PyTorch Lightning, an open-source Python library that is a PyTorch wrapper and handles data loading, distributed training, and logging [73]. The development workflow is going to be explained, from the generation of the data to the training process. Then, the results obtained with the trained surrogate models are shown, firstly considering a Neo-Hookean material and then a transversely isotropic material with a family of fibres embedded.

It is important to mention that all the results that are going to be presented are valid for any system of units that is consistent. As an example, if the material parameter used for the Neo-Hookean model was $C_{10} = 2$ and if the unit considered for the material parameter is MPa, all the stresses shown are in MPa, displacements in mm and forces in N.

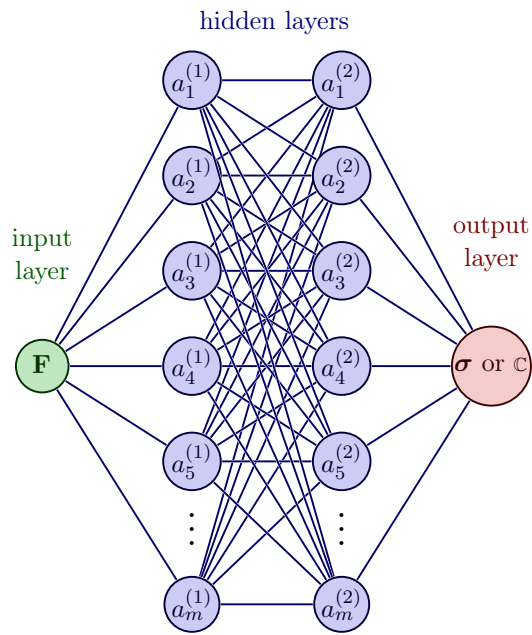


Figure 4.1: ANN architecture

4.1 Development Workflow

4.1.1 Dataset Generation

To train the surrogate models, it was necessary to generate the data that was going to be used to train the NNs. Firstly, it is explained the approach used to generate the deformation gradients, followed by the presentation of the reasoning applied to generate $\boldsymbol{\sigma}$ and \mathbf{c} .

4.1.1.1 Deformation Gradient

The deformation gradients were obtained as

$$\mathbf{F} = \prod_{\theta}^m ({}^e\mathbf{R}_m \mathbf{F}_m {}^e\mathbf{R}_m^T), \quad \text{with } \begin{cases} m = \{uni, bi, ss\} \\ \theta \in [0^\circ, 180^\circ] \\ e = \{Ox, y, z\} \end{cases} \quad (4.1)$$

where \mathbf{F}_m are the deformation gradients representative of three homogeneous loading cases: uniaxial load ($m = uni$), biaxial load ($m = bi$) and simple shear load ($m = ss$). Each of these deformation gradients has the general representative form defined as

$$\mathbf{F}_{uni} = \begin{bmatrix} \lambda_{uni} & 0 & 0 \\ 0 & \lambda_{uni}^{-0.5} & 0 \\ 0 & 0 & \lambda_{uni}^{-0.5} \end{bmatrix} \quad (4.2)$$

$$\mathbf{F}_{bi} = \begin{bmatrix} \lambda_{bi_1} & 0 & 0 \\ 0 & \lambda_{bi_2} & 0 \\ 0 & 0 & \lambda_{bi_1}^{-1} \cdot \lambda_{bi_2}^{-1} \end{bmatrix} \quad (4.3)$$

$$\mathbf{F}_{ss} = \begin{bmatrix} 1 & \gamma & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.4)$$

For each value of m there is an associated rotation matrix ${}^e\mathbf{R}_m$, defined by an axis of rotation e and an angle θ .

If a dataset with only uniaxial loading cases in a given direction is desired, \mathbf{F}_{bi} and \mathbf{F}_{ss} are equal to the identity matrix \mathbf{I} , as well as their corresponding rotation matrix \mathbf{R}_{bi} and \mathbf{R}_{ss} . A summary of the combinations used is shown in Table 4.1.

	$m = uni$		$m = bi$		$m = ss$	
	\mathbf{R}	\mathbf{F}	\mathbf{R}	\mathbf{F}	\mathbf{R}	\mathbf{F}
Uniaxial	$\theta = 0^\circ$ $e = Ox$	\mathbf{F}_{uni}	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}
Biaxial	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}	$\theta = 0^\circ$ $e = Ox$	\mathbf{F}_{bi}	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}
Simple shear	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}	$\theta = 0^\circ$ $e = Ox$	\mathbf{I}	$\theta = 0^\circ$ $e = Ox$	\mathbf{F}_{ss}
Random	$\theta = [0^\circ, 180^\circ]$ $e = \{Ox, y, z\}$	\mathbf{F}_{uni}	$\theta = [0^\circ, 180^\circ]$ $e = \{Ox, y, z\}$	\mathbf{F}_{bi}	$\theta = [0^\circ, 180^\circ[$ $e = \{Ox, y, z\}$	\mathbf{F}_{ss}

Table 4.1: Combinations to obtain the deformation gradients

To use the trained surrogate models, it was important to train them with a wide range of deformations in order to make it viable to use them in complex numerical examples, where the deformation gradient is much different from the ones present in eqs. (4.2) to (4.4). Therefore, a random deformation gradient could be obtained with eq. (4.1), where the following assumptions and considerations were taken:

1. It was considered that the final \mathbf{F} was obtained by multiplying the deformation gradient representative of three homogeneous deformations (\mathbf{F}_m), namely the uniaxial, the biaxial and the simple shear;
2. Each \mathbf{F}_m were rotated by a random rotation matrix \mathbf{R}_m ;
3. The random rotation matrixes were obtained considering that the axis of rotation was coincident with one of the coordinate system directions (x, y or z direction) and the angle of rotation that could go from 0 to 180°, with increments of 5°. The function defined to obtain these matrixes is the following one:

```

1 def random_rotation():
2     axis = np.random.randint(0,3)
3     increment=5*np.pi/180 #angle increment in radians
4     angle = np.random.choice(np.arange(0,np.pi+increment,increment))
5
6     # Compute Rotation matrix based on the axis and the angle
7     R = np.zeros((3,3))
8     if axis == 0: # x-axis
9         R = np.array([[1, 0, 0],
10                      [0, np.cos(angle), -np.sin(angle)],
11                      [0, np.sin(angle), np.cos(angle)]])
12     elif axis == 1: # y-axis
13         R = np.array([[np.cos(angle), 0, np.sin(angle)],
14                      [0, 1, 0],
15                      [-np.sin(angle), 0, np.cos(angle)]])
16     else: # z-axis
17         R = np.array([[np.cos(angle), -np.sin(angle), 0],
18                      [np.sin(angle), np.cos(angle), 0],

```

```

19         [0, 0, 1]])
20     return R

```

Listing 4.1: Random rotation matrix

4. It was assumed that the deformation was isochoric, which implied that $J = 1$. To ensure this, it was necessary to rescale \mathbf{F} after applying eq. (4.1): $\mathbf{F} = \mathbf{F}/\det(\mathbf{F}^{1/3})$;

To better understand the reasoning applied to obtain a random deformation gradient, an example is shown below:

$$\mathbf{F}_{uni} = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5^{-0.5} & 0 \\ 0 & 0 & 1.5^{-0.5} \end{bmatrix} \text{ and } \mathbf{R}_{uni} = \begin{bmatrix} \cos(5^\circ) & -\sin(5^\circ) & 0 \\ \sin(5^\circ) & \cos(5^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

↓

$$\mathbf{F}_{uni} = \begin{bmatrix} 1.495 & 0.0593 & 0 \\ 0.0593 & 0.822 & 0 \\ 0 & 0 & 0.817 \end{bmatrix}$$

$$\mathbf{F}_{bi} = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 1.4 & 0 \\ 0 & 0 & (0.8 \cdot 1.4)^{-1} \end{bmatrix} \text{ and } \mathbf{R}_{bi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(135^\circ) & -\sin(135^\circ) \\ 0 & \sin(135^\circ) & \cos(135^\circ) \end{bmatrix}$$

↓

$$\mathbf{F}_{bi} = \begin{bmatrix} 0.817 & 0 & 0.0356 \\ 0 & 1.4 & 0 \\ 0.0356 & 0 & 0.876 \end{bmatrix}$$

$$\mathbf{F}_{ss} = \begin{bmatrix} 1 & -0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathbf{R}_{ss} = \begin{bmatrix} \cos(25^\circ) & 0 & \sin(25^\circ) \\ 0 & 1 & 0 \\ -\sin(25^\circ) & 0 & \cos(25^\circ) \end{bmatrix}$$

↓

$$\mathbf{F}_{ss} = \begin{bmatrix} 1 & 0.354 & -0.354 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{F} = \mathbf{F}_{uni}\mathbf{F}_{bi}\mathbf{F}_{ss} = \begin{bmatrix} 1.221 & 0.480 & -0.403 \\ 0.0831 & 1.180 & -0.0294 \\ 0.0532 & 0.0209 & 0.697 \end{bmatrix}$$

In Figure 4.2 it is shown this random deformation applied to a unitary cube.

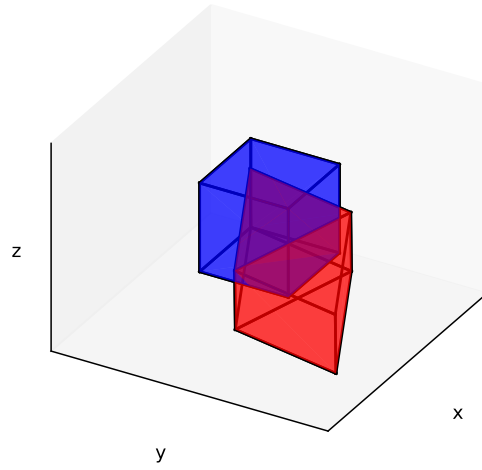


Figure 4.2: Example of random deformation gradient

The function defined in Python to get the deformation gradients is shown below:

```
1 def random_deformation_gradient(min_stretch, max_stretch, min_shear,
2 max_shear):
3
4     suniaxial = np.random.uniform(low=min_stretch, high=max_stretch)
5     sshear = np.random.uniform(low=min_shear, high=max_shear)
6     sbiaxial1 = np.random.uniform(low=min_stretch, high=max_stretch)
7     sbiaxial2 = np.random.uniform(low=min_stretch, high=max_stretch)
8
9     # Random stretch factors for uniaxial deformation
10    s1 = suniaxial
11    s2 = 1/np.sqrt(s1)
12    s3 = s2
13
14    # Random shear factors for simple shear deformation
15    r = sshear
16
17    # Random stretch factors for biaxial deformation
18    s7 = sbiaxial1
19    s8 = sbiaxial2
20    s9 = 1.0/(s7*s8)
21
22    F_uni = np.array([[s1, 0, 0],
23                    [0, s2, 0],
24                    [0, 0, s3]])
25    F_bi = np.array([[s7, 0, 0],
26                  [0, s8, 0],
27                  [0, 0, s9]])
28    F_ss = np.array([[1, r, 0],
29                  [0, 1, 0],
30                  [0, 0, 1]])
31
32    R_uni = random_rotation()
33    R_bi = random_rotation()
34    R_ss = random_rotation()
35
36    # Rotate deformation gradients
37    F_uni = np.matmul(np.matmul(R_uni, F_uni), R_uni.T)
38    F_bi = np.matmul(np.matmul(R_bi, F_bi), R_bi.T)
39    F_ss = np.matmul(np.matmul(R_ss, F_ss), R_ss.T)
40
41    F = np.matmul(np.matmul(F_uni, F_bi), F_ss)
42    # Isochoric deformation gradient
43    det = np.linalg.det(F)
44    F_rescaled = F / det**(1.0/3.0)
45
46    return F_rescaled
```

Listing 4.2: Random deformation gradient generation

4.1.1.2 Cauchy Stress and Spatial Elasticity Tensor

To obtain the isochoric part of σ and \mathbb{C} for each \mathbf{F} generated, it was used a Python library named `Pandarallel` that speeds up computations performed in `pandas`. The deformation gradients were placed in a `Pandas` data frame, and a function was applied to each row of the data frame to compute the desired tensor for the deformation gradient present in the given row. To obtain the isochoric part of σ , the SEFs of the material models in analysis (defined in Section 3.4.3) were used, as well as the eqs. presented in Section 3.4.2.3. The values for the material parameters required to compute the SEF were considered known. The same reasoning was applied to obtain the isochoric part of \mathbb{C} , where the eqs. presented in Section 3.4.4 were used. The Python code used to obtain these two tensors is shown below:

```
1 def stress(F):
2     #... Equations to obtain Cauchy stress tensor or the spatial elasticity
3     #... tensor with the SEF of the material model considered
4     return Variable
5
6 def generate_df(n):
7     pandarallel.initialize(progress_bar=True)
8     df=pd.DataFrame()
9     df["F"]=[random_deformation_gradient(0.8, 1.6, -0.4, 0.4) for j in
10             range(n)]
11     df['Variable'] = df.parallel_apply(lambda x: stress(x.F), axis=1)
12     return df
13
14 df=generate_df(n)
```

Listing 4.3: Stress and elasticity tensors generation

4.1.2 Loading the Data

After having the dataset properly generated, it was necessary some pre-processing before proceeding to the training stage. It was required to transform the `numpy.arrays` to `torch.tensors` and to split the dataset into training, validation and testing data. The training data is used to train the model and to fit the parameters, whereas the validation data is used to improve the model performance by fine-tuning after each epoch. The test data has information that the NN never saw, so it is possible to evaluate the model in an unbiased way. The percentages of the whole dataset considered for each split are shown in Table 4.2.

	Training Data	Validation Data	Test Data
Percentage of the whole dataset (%)	80	10	10

Table 4.2: Dataset split

The Python code to accomplish the pre-processing of the dataset is shown below:

```
1 class Dataset():
2     def __init__(self, features, labels):
3         self.features=torch.tensor(features, dtype=torch.float32)
4         self.labels=torch.tensor(labels, dtype=torch.float32)
5
6     def __len__(self):
7         return len(self.labels)
8
9     def __getitem__(self, idx):
10        return self.features[idx],self.labels[idx]
11
12 def pre_process(n_inputs, n_outputs, test_split, val_split):
13     df=pd.read_pickle(f"(...).pkl")
14     n=df.shape[0]
15     features=np.reshape(np.vstack(df["F"].to_numpy()),(n,n_inputs))
16     labels=np.reshape(np.vstack(df["SIGMA"].to_numpy()),n,n_outputs))
17
18     train_features, test_features, train_labels, test_labels =
19         train_test_split(features, labels, test_size = test_split)
20     val_split=val_split/(1-test_split)
21     train_features, valid_features, train_labels, valid_labels =
22         train_test_split(train_features, train_labels, test_size =
23         val_split)
24
25     train_set=Dataset(train_features, train_labels)
26     test_set=Dataset(test_features, test_labels)
27     valid_set=Dataset(valid_features, valid_labels)
28
29     return train_set, valid_set, test_set
```

Listing 4.4: Data pre-processing

Afterwards, it is possible to use the class `LightningDataModule`, where the batch size was defined in the `__init__` method. Then the `train_dataloader()`, the `val_dataloader()` and the `test_dataloader()` methods were used to define the training, validation and testing data, respectively. For each one of them, the `DataLoader` module was used, which took as input the respective data set and the batch size [73, 74].

```
1 class DataModule(pytorch_lightning.LightningDataModule):
2     def __init__(self, batch_size):
3         super().__init__()
4         self.batch_size = batch_size
5     def train_dataloader(self):
6         return torch.utils.data.DataLoader(train_set, batch_size=self.
7         batch_size)
8     def val_dataloader(self):
9         return torch.utils.data.DataLoader(valid_set, batch_size=self.
10        batch_size)
```



```

9     def test_dataloader(self):
10         return torch.utils.data.DataLoader(test_set, batch_size=self.
            batch_size)

```

Listing 4.5: Create DataModule

4.1.3 Training with PyTorch Lightning

As mentioned before, the models were trained with PyTorch Lightning and to do so, two stages were necessary:

Configuring the model PyTorch Lightning is based on PyTorch but offers increased structurability to the code and it is easier to scale models across multiple graphics processing units and, thus, speed up the training of the model.

The models are created in the class `LightningModule`, based on the PyTorch `nn.Module`. Both of them provide the building blocks to create ANNs, but the `LightningModule` provides additional functionality. It can be organized into the following main sections [73, 74]:

- Initialization (`__init__` and `setup()`), where the architecture of the NN can be defined. Then the `forward()` method takes the inputs, passes them by the multiple layers and activation functions defined, and gives the outputs of the model.

```

1     def __init__(self, learning_rate):
2         super(Model, self).__init__()
3         self.layers = torch.nn.Sequential(
4             # 1st hidden layer
5             torch.nn.Linear(n_inputs, layer1), torch.nn.ReLU(),
6             # 2nd hidden layer
7             torch.nn.Linear(layer1, layer2), torch.nn.ReLU(),
8             # output layer
9             torch.nn.Linear(layer2, n_outputs))
10        self.lr = learning_rate
11    def forward(self, x):
12        return self.layers(x)

```

Listing 4.6: Initialization

- Training Loop (`training_step()`), where the training is completed. This method takes as input two variables: `batch` that consists of the features and the targets present in the training dataset, and `batch_idx` which is the index number for the batch of data. With the `forward()` method, the prediction of the model and a loss function can be calculated;

```

1     def training_step(self, train_batch, batch_idx):
2         features, labels = train_batch
3         logits = self.forward(features)
4         loss = torch.nn.functional.mse_loss(logits, labels)

```

```

5     #Logging options can be added
6     return loss

```

Listing 4.7: Training

- Validation Loop (`validation_step()`), where the validation occurs. It is similar to the `training_step()` method, but here the validation dataset is used instead of the training one;

```

1     def validation_step(self, valid_batch, batch_idx):
2         features, labels = valid_batch
3         logits = self.forward(features)
4         loss = torch.nn.functional.mse_loss(logits, labels)
5         #Logging options can be added

```

Listing 4.8: Validation

- Testing Loop (`test_step()`), where the testing of the model is completed analogously to the previous two methods, but this time using the testing dataset;

```

1     def test_step(self, test_batch, batch_idx):
2         features, labels = test_batch
3         logits = self.forward(features)
4         loss = F.mse_loss(logits, labels)
5         #Logging options can be added
6

```

Listing 4.9: Testing

- Optimizers (`configure_optimizers()`) where the optimizers can be configured. The ADAM optimizer is used here and it is also in this method that it is possible to define the learning rate.

```

1     def configure_optimizers(self):
2         optimizer = torch.optim.Adam(self.parameters(), lr=self.lr)
3         return optimizer

```

Listing 4.10: Optimization

Each of these methods should be placed inside the `LightningModule` class:

```

1 class Model(pytorch_lightning.LightningModule):
2     def __init__(self, learning_rate):
3         ...
4     def forward(self, x):
5         ...
6     def training_step(self, train_batch, batch_idx):
7         ...
8     def validation_step(self, valid_batch, batch_idx):
9         ...

```

```

10     def test_step(self, test_batch, batch_idx):
11         ...
12     def configure_optimizers(self):
13         ...

```

Listing 4.11: LightningModule

Training and testing the model The models built in PyTorch Lightning can be trained with the `Trainer` class that loops over the datasets and clears the gradients, for example. It supports many functionalities such as logging, callbacks, epochs, etc. Once this class is defined, the `fit` method is called, which receives the defined model and the dataset. Finally, the `test` method can be used to test the trained model with data that was not used to train [73, 74].

```

1 trainer = pytorch_lightning.Trainer()
2 model = Model(learning_rate)
3 datamodule = DataModule(batch_size)
4 trainer.fit(model, datamodule=datamodule)
5 trainer.test(model, datamodule=datamodule)

```

Listing 4.12: Trainer fit and test

4.2 Neo-Hookean Model

To begin, it was considered a Neo-Hookean material and it was assumed a fixed material parameter ($C_{10} = 2$). For this material, different models were developed, with increasing complexity. Firstly, they were trained models to predict the isochoric Cauchy stress tensor, some for homogeneous loading cases and another for an arbitrary deformation. For the arbitrary deformation, the hyperparameters, namely the architecture of the NN and the batch size, were tuned, followed by a search for the optimal learning rate. Afterwards, it was trained a model for the computation of the isochoric spatial elasticity tensor.

4.2.1 Homogeneous Deformations

A surrogate model was developed for homogeneous deformation cases: uniaxial tension, biaxial tension and simple shear. For each case, a NN was trained and tested as it is going to be presented. Each one of the datasets used for these models has 5×10^{-5} deformation gradients representative of the deformation in analysis and the corresponding isochoric part of the Cauchy stress tensor.

The hyperparameters used for every homogeneous deformation case were the same: a batch size of 64, an input layer with 9 inputs that correspond to all the components of the deformation gradient, two hidden layers with 64 and 32 neurons each and a final output layer with 9 outputs that correspond to the components of the Cauchy stress tensor, as it is summarized in Table 4.3. The maximum number of epochs was different for each homogeneous deformation since the `EarlyStopping` callback was used, which monitors the validation loss and stops the training when no significant improvement is observed. Additionally, the activation function used was the ReLU, the loss function was the MSE and the ADAM optimizer was used. Instead of defining a learning rate, a learning rate finder available in PyTorch Lightning was used. It does a small run with a learning rate that increases after each processed batch and then the corresponding loss is logged, which gives guidance for the optimal learning rate.

	Size	Activation Function
Input Layer	9	-
First Hidden Layer	64	ReLU
Second Hidden Layer	32	ReLU
Output Layer	9	-

Table 4.3: NN architecture

The results obtained for each homogeneous loading case are shown in Figure 4.3, where each row is related to each one of the deformations considered. In the first column, an example of the deformation in the analysis is displayed, where the blue cube represents the undeformed configurations and the red one is the deformed configuration. The second

column shows the evolution of the training and validation loss and in the third column, the losses in the last epoch are shown.

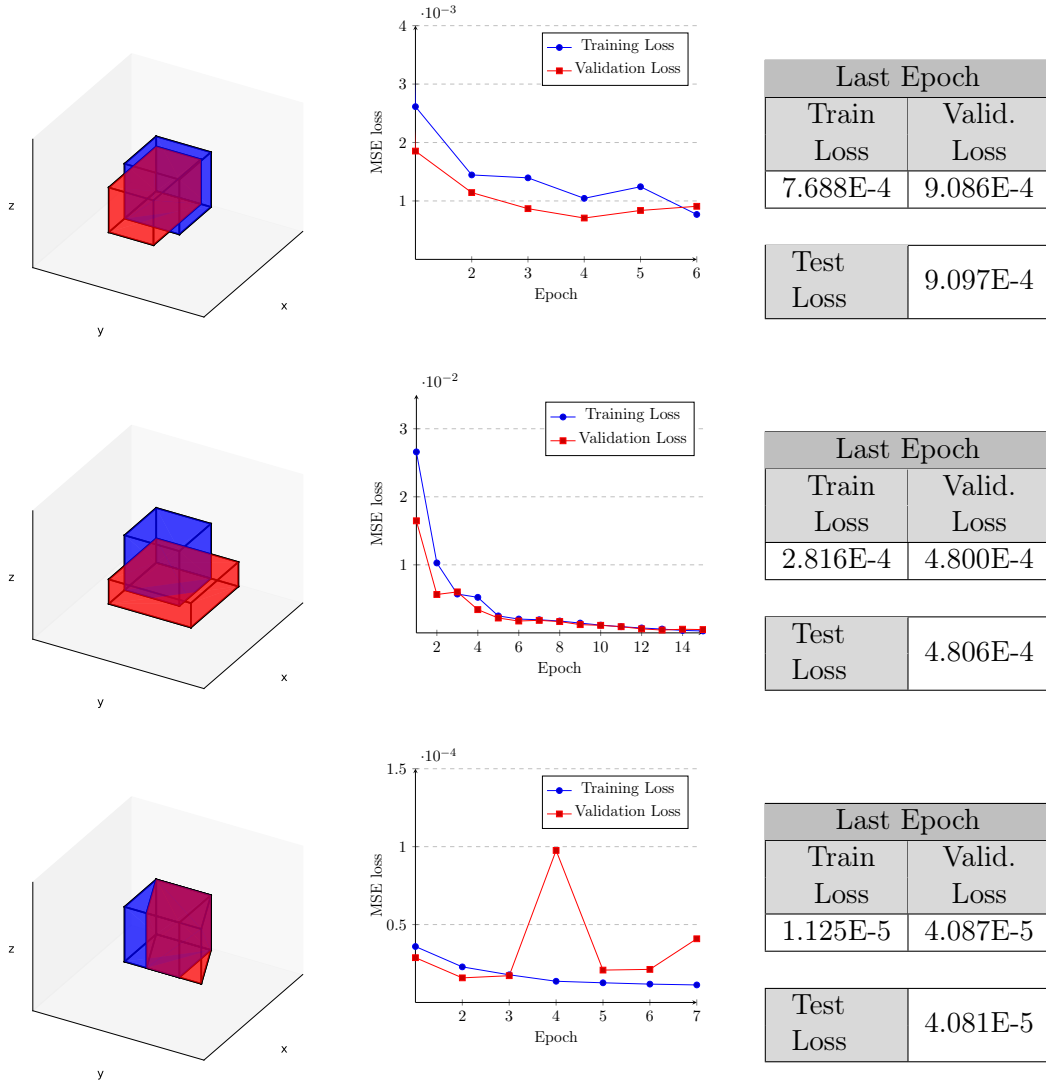


Figure 4.3: Results for each NN trained

In the graph with the evolution of the training and validation losses for each deformation, it is possible to observe the expected decaying behaviour of both losses as the number of epochs increases. The test loss is small, which shows that the model is able to accurately predict the components of the Cauchy stress tensor for a given homogeneous deformation.

4.2.2 Arbitrary Deformation

Despite the excellent results presented, there are some limitations. The developed models showed good results when tested with deformation gradients representative of the corresponding homogeneous loading case, but they are not suitable for a general loading case. Therefore, to overcome this limitation, a model was trained to predict the isochoric part of the Cauchy stress tensor for a given random deformation gradient. Considering that Cauchy stress tensor is symmetric for an isochoric deformation gradient, it is possible to reduce the outputs of the NN from 9 to 6.

The dataset used for this model is composed of 5×10^7 random deformation gradients and the corresponding stress. The split of the dataset was done with the values from Table 4.2, the optimizer used was the ADAM algorithm and the loss function used was the MSE. The NN architecture and the batch size were tuned with Optuna.

With the NN architecture defined, the learning rate was defined. To do so, the learning rate finder (`lr_find()`) implemented in PyTorch Lightning was used. Such a learning rate finder increases the learning rate in each processed batch and the corresponding loss is logged, which produces a plot with the loss as a function of the learning rate and suggests a good value for the learning rate. The learning rate is the hyperparameter with more importance for this problem and is influenced by the NN architecture. This was the reason to first tune the architecture and the batch size, and only then define a good value for the learning rate.

To tune the desired hyperparameters with Optuna, an objective function was defined and the possible values for the hyperparameters were defined with the `suggest` API. The hyperparameters and the possible values are shown in Table 4.4: the number of layers is an integer value chosen within a specified interval, the number of neurons and the batch size are also integers, but they are chosen from a given list. The objective function runs every trial and it gives the evaluation metric, which is the validation loss in this case. The study was also defined, where the direction of the evaluation criteria was defined (minimize in this case), as well as the pruning strategy. Finally, the `optimize` API was invoked, with the objective function and the number of trials as inputs.

Hyperparameter	Type	Values
Number of layers	Integer	[2,4]
Number of neurons	Categorical	{4, 8, 16, 32, 64, 128, 256}
Batch size	Categorical	{16, 32, 64, 128, 256}

Table 4.4: Hyperparameters - Possible values

In this analysis, only a study was carried out with 100 trials and applied to a random 1 % of the training set in order to speed up the tuning process. The results are shown in Table 4.5.

	Study results
Number of layers	3
Number of neurons	256, 256, 256
Batch size	64

Table 4.5: Hyperparameter tuning results - Stress Neo-Hook

The evolution of the test and validation losses is shown in Figure 4.4 and the values of the final losses are presented in Table 4.6.

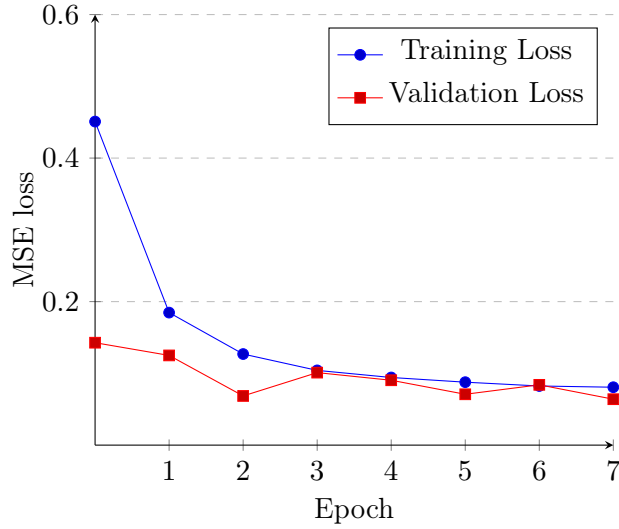


Figure 4.4: Evolution of training and validation losses - Stress Neo-Hook

	Last Epoch		
Loss	Train	Valid.	Test
	0.0807	0.0640	0.0680

Table 4.6: Losses in last epoch - Stress Neo-Hook

The constitutive behaviour of a hyperelastic material is described by the relationship between stresses and strains, but to apply the FE method, it is necessary to compute the tangent stiffness matrix. To do it, it is required to calculate the spatial elasticity tensor. Therefore, it is necessary to get a model that predicts the isochoric part of the spatial elasticity tensor, which is then used to compute the tangent stiffness matrix. The reasoning applied and described to obtain the Cauchy stress tensor for a general deformation gradient is also applied here. The dataset used for this model is composed of 1×10^7 random deformation gradients and the corresponding spatial elasticity tensor, where the major symmetries were considered to reduce the number of outputs from 36 to 21. Everything else is equal to the previous model and regarding the NN architecture and batch size, another tuning was performed, with the associated results shown in Table 4.7.

The learning rate was also tuned and the results of the training are shown in Figure 4.5 and Table 4.8.

	Study results
Number of layers	2
Number of neurons	256,128
Batch size	16

Table 4.7: Hyperparameter tuning results - Elasticity tensor Neo-Hook

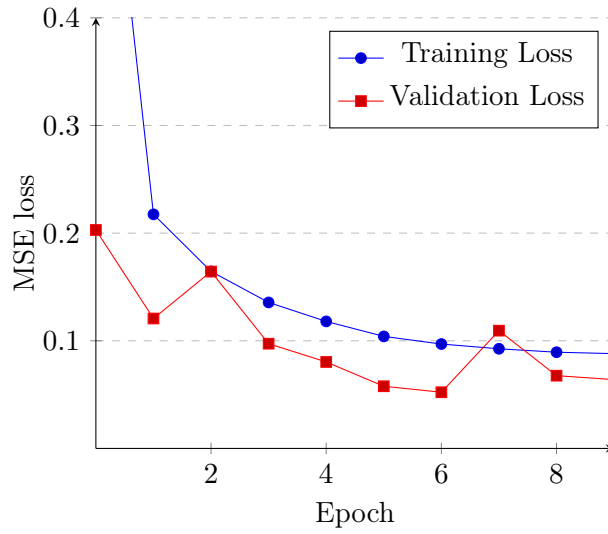


Figure 4.5: Evolution of training and validation losses - Elasticity tensor Neo-Hook

	Last Epoch		
Loss	Train	Valid.	Test
	0.0880	0.0671	0.0698

Table 4.8: Losses in last epoch - Elasticity tensor Neo-Hook

4.3 HGO Model

Soft biological tissues and other hyperelastic materials have fibres that confer anisotropy to the material. To describe these materials, some models available in the literature were already introduced, such as the HGO model. This model is more complex than the Neo-Hookean model and it is going to be used to further prove the capabilities of modelling hyperelastic materials with a ML-based approach, even when the material has one preferred direction.

The dataset consisted of 1×10^7 deformation gradients only with uniaxial loading cases (generated with the first combination of the first row of Table 4.1), the 6 independent components of the Cauchy stress tensor and two angles that describe the direction of the fibre or the preferred direction: $\theta \in [0^\circ, 180^\circ[$ and $\psi \in [0^\circ, 90^\circ[$. These angles and a radial distance $r = 1$ define a vector in spherical coordinates, that can then be used to get the direction in Cartesian coordinates, given by

$$\begin{aligned} x &= r \cdot \cos(\theta) \cdot \sin(\psi) \\ y &= r \cdot \sin(\theta) \cdot \sin(\psi) \\ z &= r \cdot \cos(\psi). \end{aligned} \tag{4.5}$$

Firstly, total freedom to the angles was given, but this resulted in a really complex behaviour that could not be learned by a NN with the amount of data considered because a different direction changes completely the constitutive behaviour of the material. Thus, the angle was varied in increments of 45° . The material parameters used for the HGO model are the ones shown in Table 4.9.

C_{10}	k_1	k_2	κ
2	1	1	0.2

Table 4.9: Material parameters HGO model

The trained model for the SEF proposed by HGO was also tuned to get a good architecture. The batch size selected from the tuning process was 16 and the NN architecture is the one shown in Table 4.10.

	Size	Activation Function
Input Layer	11	-
First Hidden Layer	256	ReLU
Second Hidden Layer	256	ReLU
Third Hidden Layer	32	ReLU
Output Layer	6	-

Table 4.10: NN architecture - HGO

Since all the deformation gradients consisted of uniaxial deformations in the x-direction, the first input of the NN was coincident with the stretch applied. Therefore, it was possible to apply a stratification to split the data into the training, testing and validation datasets, which can improve the outcome of the trained model and its generalization capabilities. The distribution of the train+validation set and the test set with and without stratification is shown in Figure 4.6. The training results are shown in Figure 4.7 and Table 4.11.

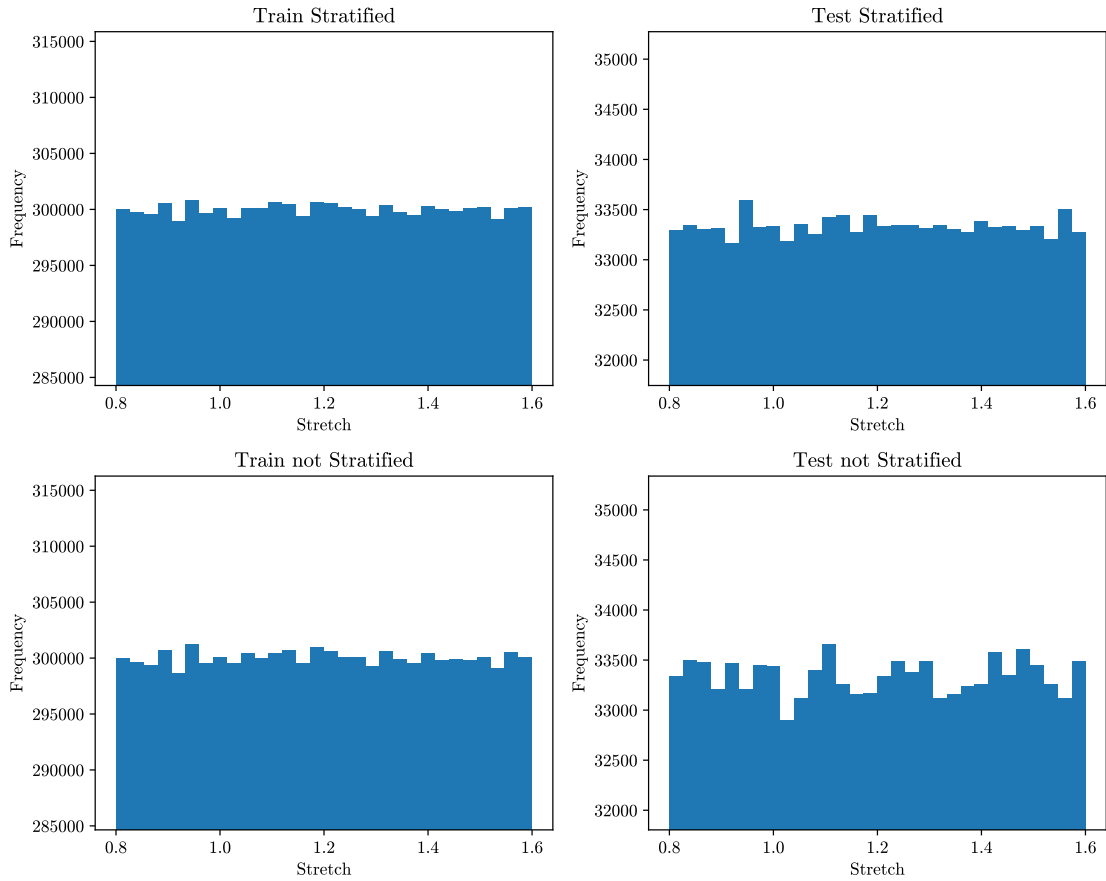


Figure 4.6: Stratification

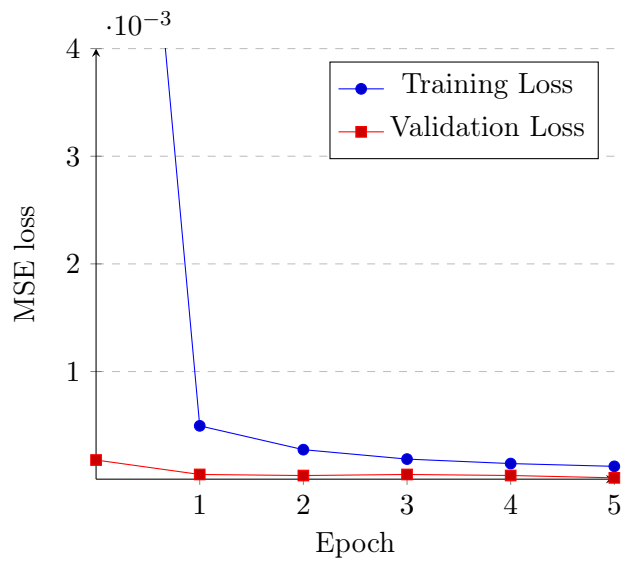


Figure 4.7: Evolution of training and validation losses - Stress HGO

Last Epoch			
Loss	Train	Valid.	Test
	1.191E-4	1.246E-5	1.276E-5

Table 4.11: Losses in last epoch - Stress HGO

Chapter 5

Finite Element Method Integration

The trained models showed a small value for the test loss, which is indicative that they can predict accurately the desired outputs. However, the main goal of this dissertation is to use the trained surrogate models in Abaqus to analyse problems with hyperelastic materials. Thus, it was necessary to pass the trained models to the Abaqus environment, which then make it possible to employ the FE method and to use all its capabilities, such as modelling the problem, managing, monitoring and visualizing results. The procedure to accomplish such a task is explained in this Chapter, as well as the comparison of the results obtained with a ML approach and with the conventional way.

5.1 Converting Forward Pass to Fortran

As mentioned before, it is possible to define a UMAT in Abaqus, where the constitutive behaviour of the material in the analysis is modelled. The UMAT is written in Fortran and, conventionally, it requires the definition of the constitutive equations. The constitutive equations involve many partial derivatives, and Fortran does not have a built-in function to do it. Therefore, an alternative to expressing the constitutive equations is to have a ML-based approach, where the trained surrogate models are used.

To use the trained models in Fortran, it was necessary to use the values of the weights and biases of every neuron and to write the forward pass equations in a subroutine. A Python function was defined, which takes as input the `torch.nn` sequential container of the model and outputs a dictionary with a length equal to the total number of layers, where the item of each key is a vector with the output of the respective layer as a function of the input of that layer. To write the subroutine in Fortran, the output of the defined function, which is a Sympy expression, is translated into Fortran code with the Sympy submodule `fcode`. The biggest difficulty in this process was how to deal with and implement the activation function. With Sympy, it was possible to define a piecewise function to define the ReLU activation function, but it was not possible to directly transfer it to Fortran code with the `fcode` module. Therefore, after writing the equation for the forward pass of each

neuron of a layer with the module `fcode`, it was necessary to apply the activation function, which was accomplished with the evaluation of a logical expression in Fortran code.

This process was validated and, for a given deformation gradient, the stress values obtained with the trained model in PyTorch Lightning and with the subroutine created in Fortran were exactly the same.

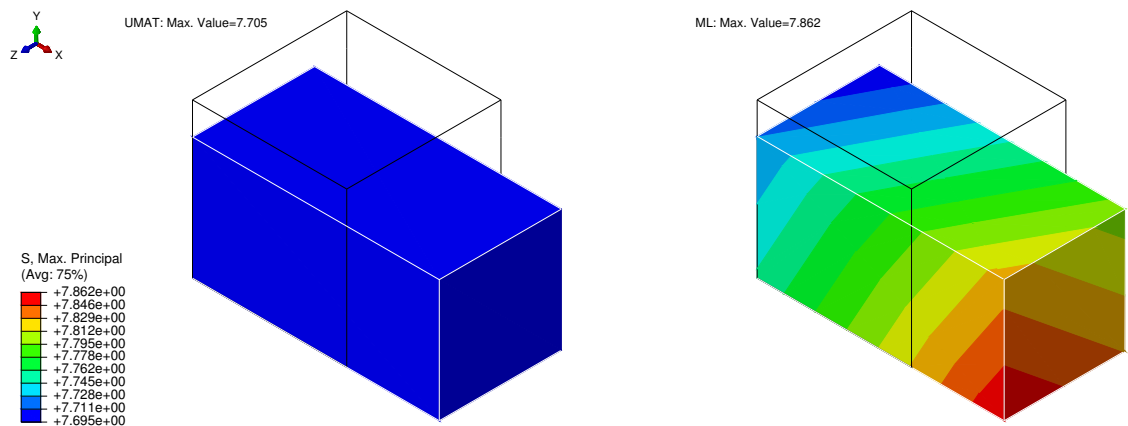
5.2 Neo-Hook Numerical Examples

To validate the integration of the trained surrogate models with the FE method, some numerical examples were analysed. For all of them, the principal stresses and the magnitude of the displacement were computed and compared. The variable in comparison is displayed in the deformed shape and the results obtained with the conventional Abaqus approach (denoted by UMAT from now on) are shown on the left-hand side and the ones obtained with the use of the surrogate models (denoted by ML from now on) are shown on the right-hand side. To better compare the results obtained, a deformed figure with the absolute difference between the two approaches is also displayed. In addition, for each time increment, the relative error between all the components of the Cauchy stress tensor was computed for all the integration points and they were displayed with a box-and-whisker plot, where the dispersion and skewness of the results are described by the minimum, the maximum, the sample median (black line), the first and third quartiles and the average (black arrow). The absolute error was also computed and displayed in a box-and-whisker plot for each problem. Finally, the total number of iterations and the total central processing unit (CPU) time required to complete the analysis were also compared.

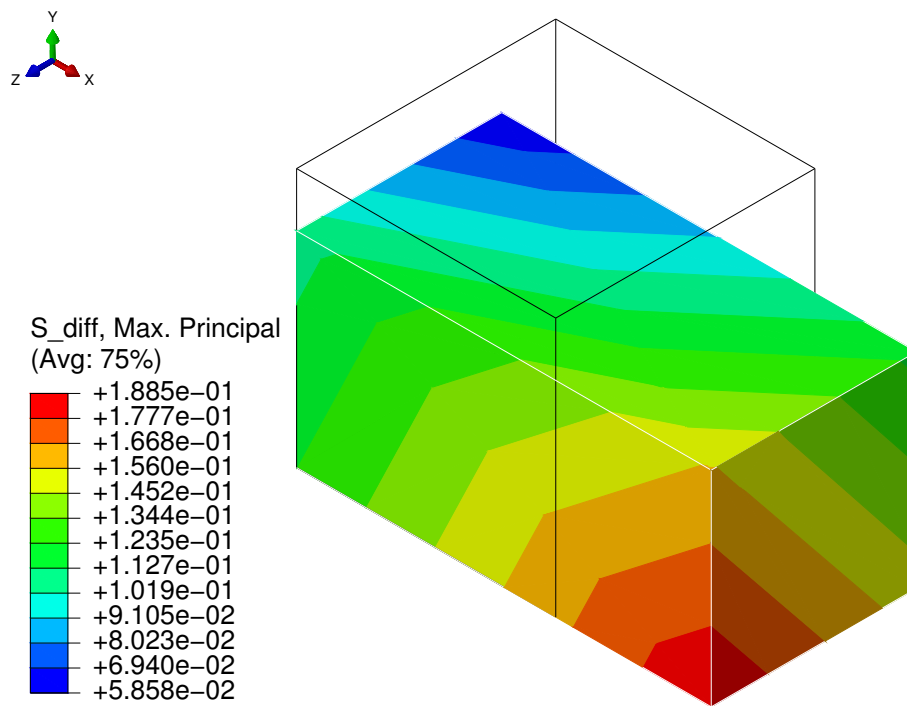
Regarding the numerical examples, the results were firstly compared in an elementary cube subjected to different loading conditions and then, more complex problems were considered.

5.2.1 One Element Cube

A cube with one hexahedral element with 8 linear nodes was submitted to three homogeneous deformations (uniaxial, biaxial and simple shear) and to a random deformation. The results obtained are shown in Figures 5.1 to 5.20.

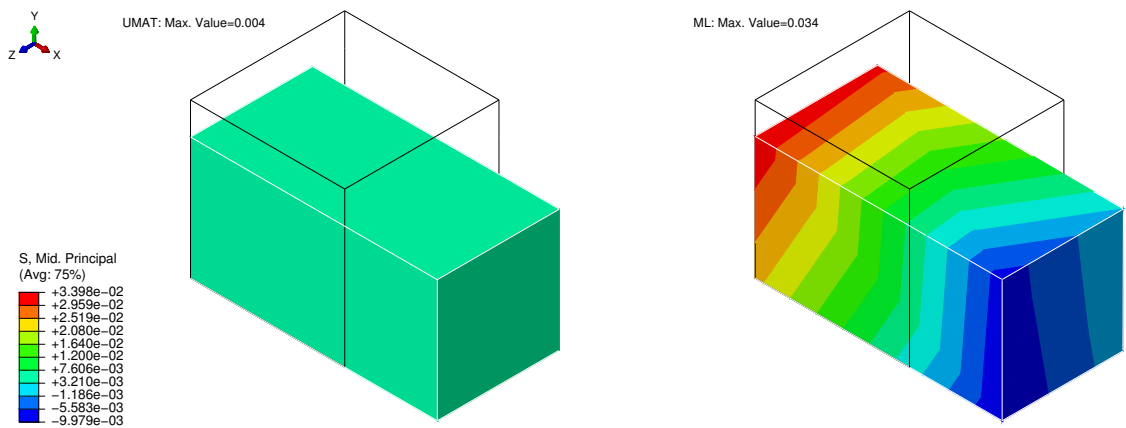


(a) UMAT (left) and ML (right)

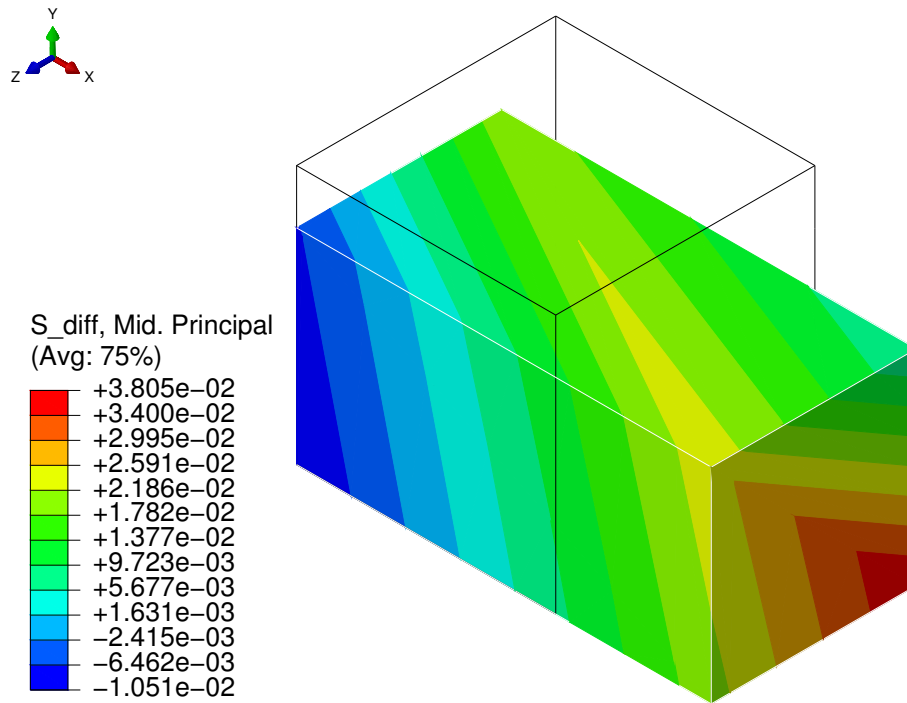


(b) Absolute difference

Figure 5.1: Maximum principal stress - Cube with uniaxial load

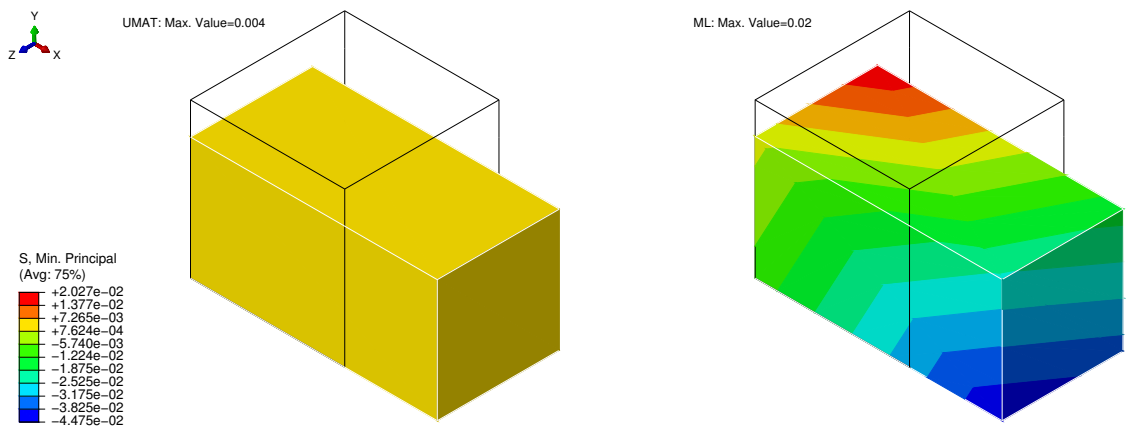


(a) UMAT (left) and ML (right)

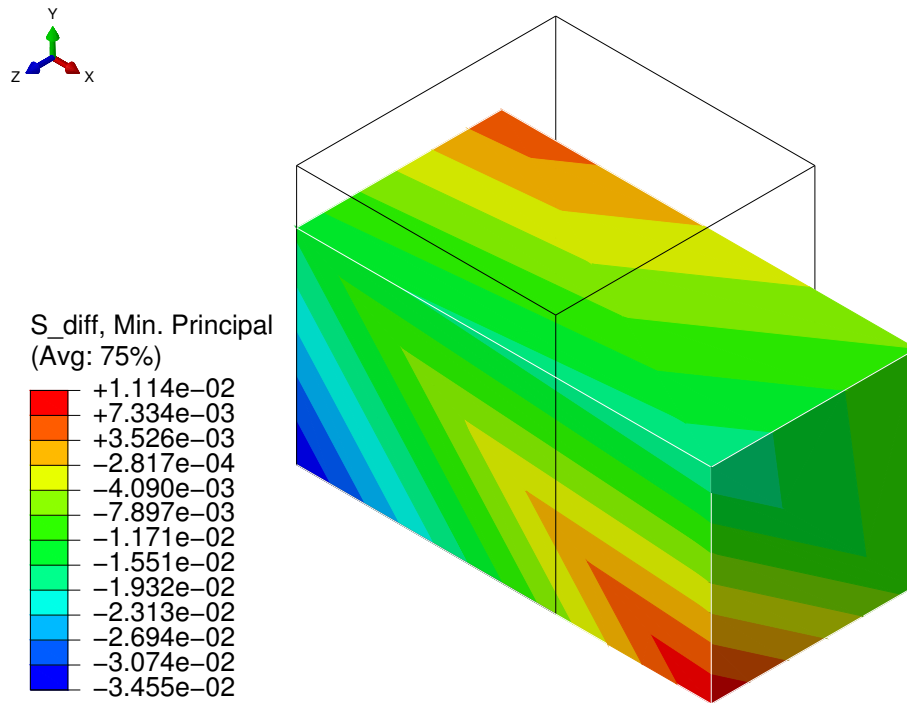


(b) Absolute difference

Figure 5.2: Middle principal stress - Cube with uniaxial load

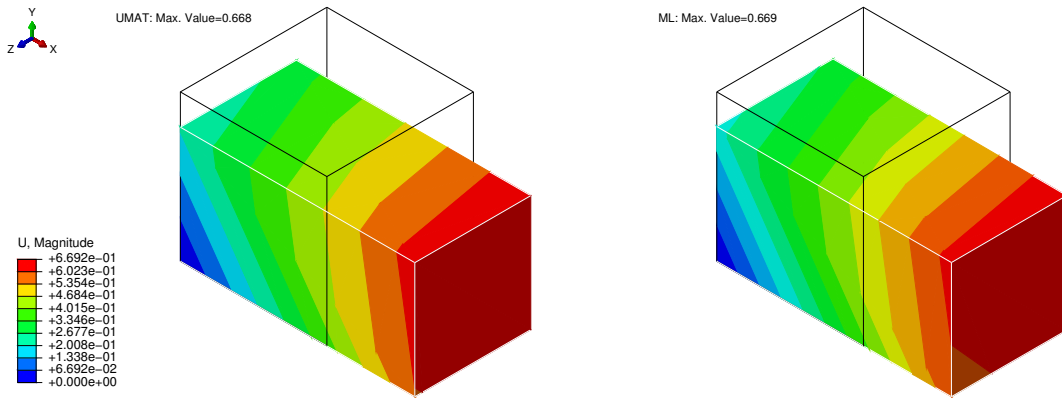


(a) UMAT (left) and ML (right)

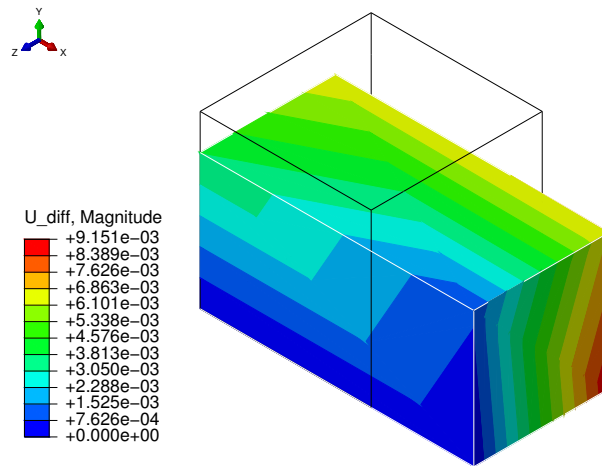


(b) Absolute difference

Figure 5.3: Minimum principal stress - Cube with uniaxial load

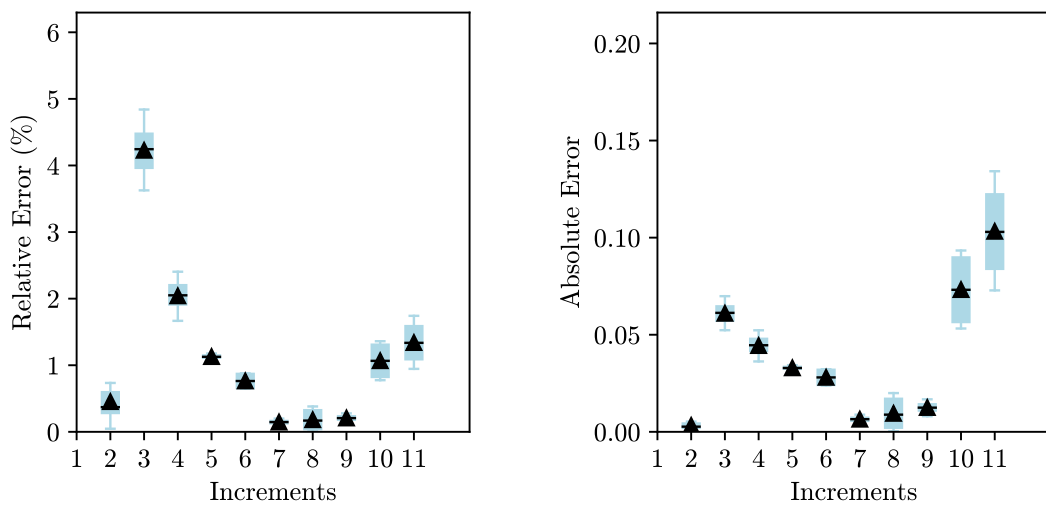


(a) UMAT (left) and ML (right)



(b) Absolute difference

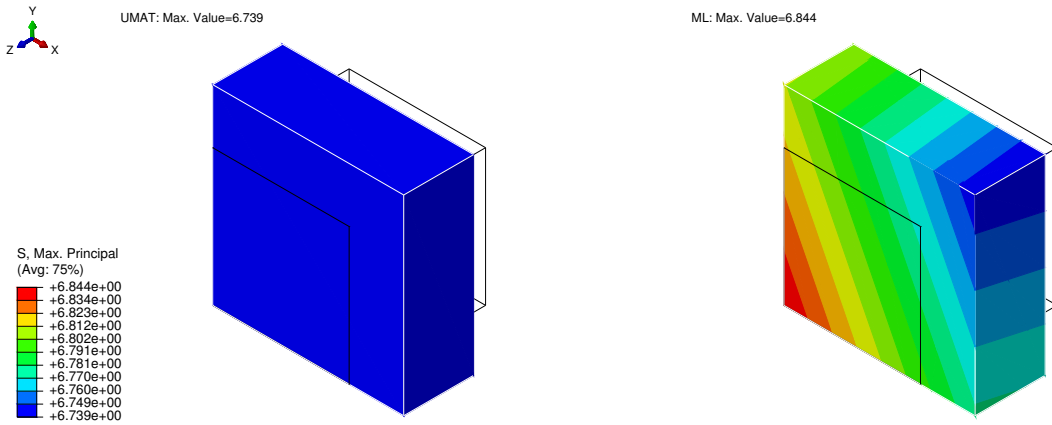
Figure 5.4: Displacement magnitude - Cube with uniaxial load



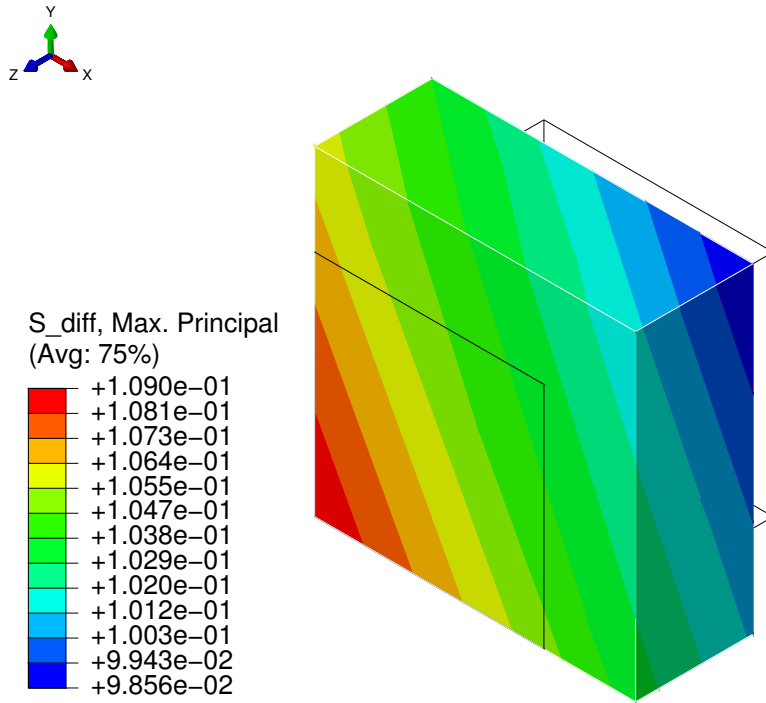
(a) Relative error

(b) Absolute error

Figure 5.5: Error - Cube with uniaxial load

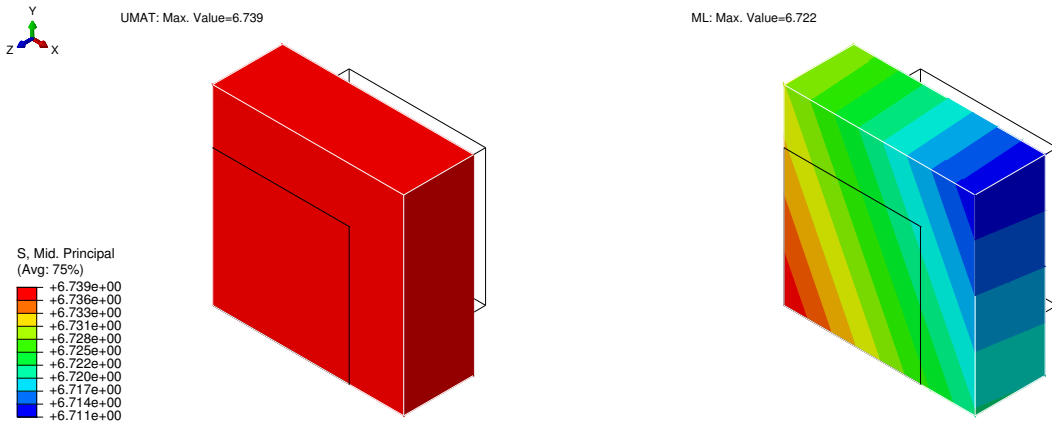


(a) UMAT (left) and ML (right)

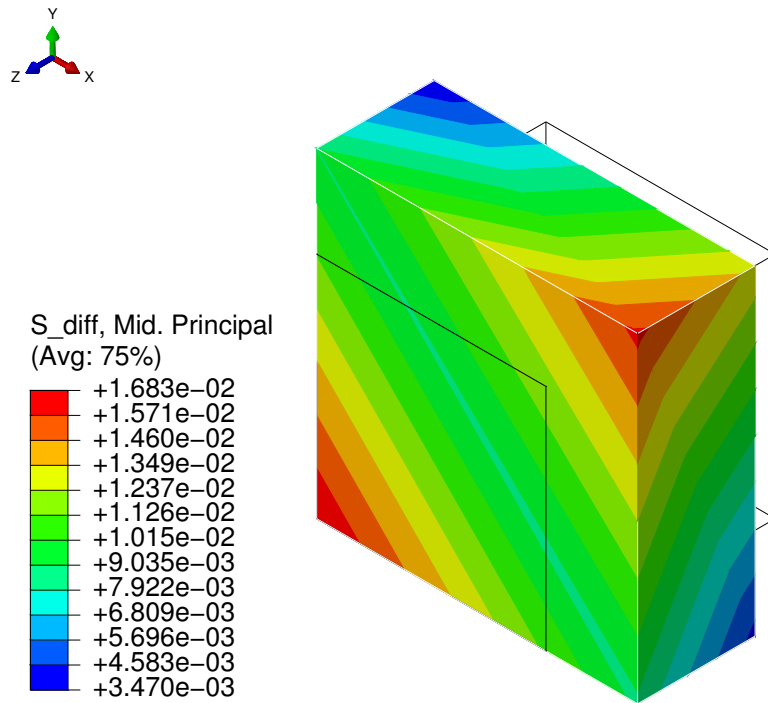


(b) Absolute difference

Figure 5.6: Maximum principal stress - Cube with biaxial load

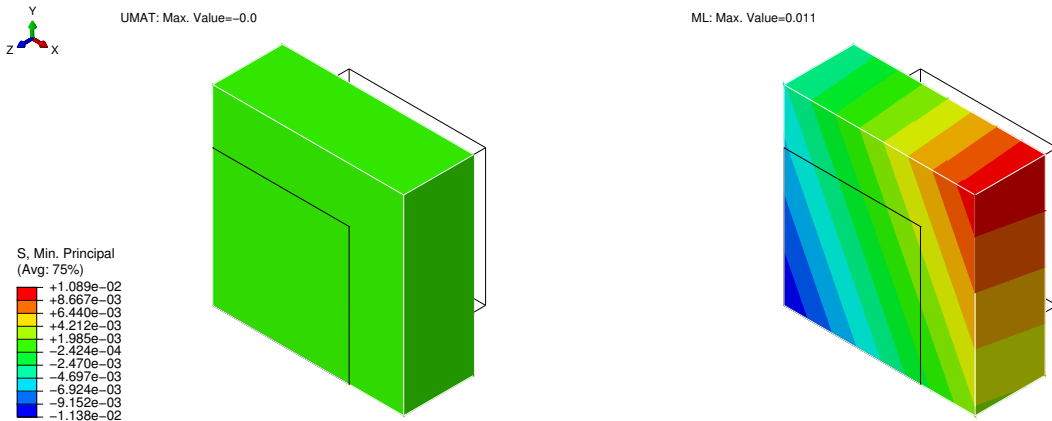


(a) UMAT (left) and ML (right)

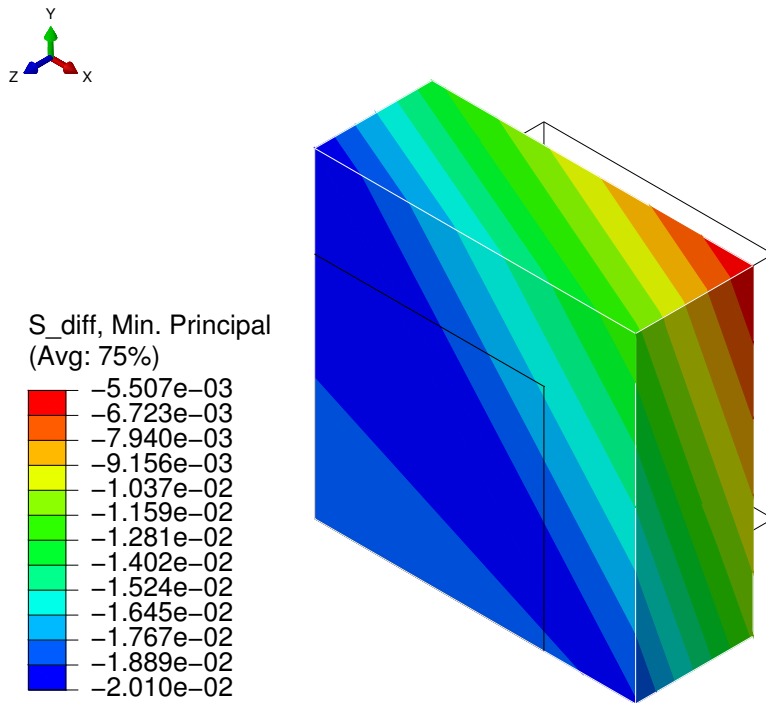


(b) Absolute difference

Figure 5.7: Middle principal stress - Cube with biaxial load

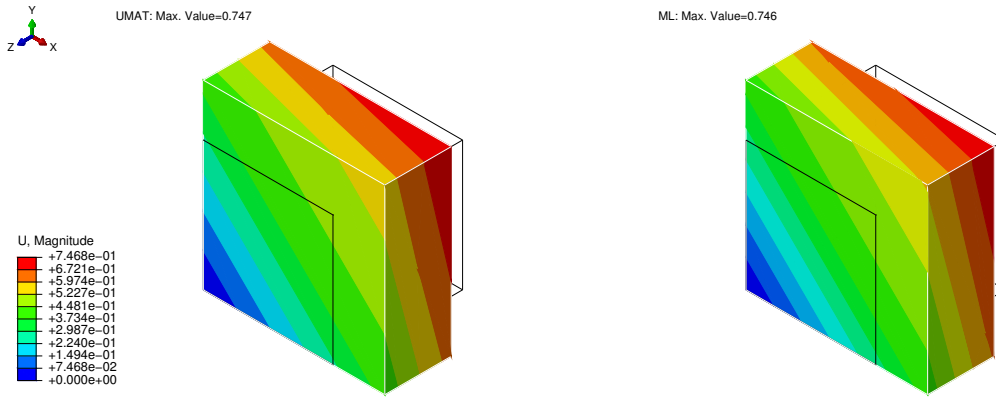


(a) UMAT (left) and ML (right)

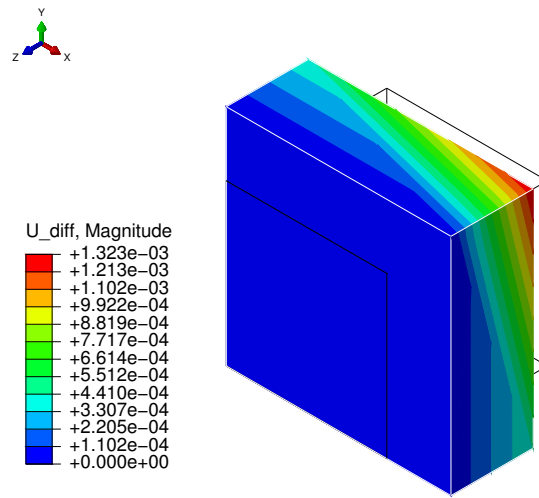


(b) Absolute difference

Figure 5.8: Minimum principal stress - Cube with biaxial load

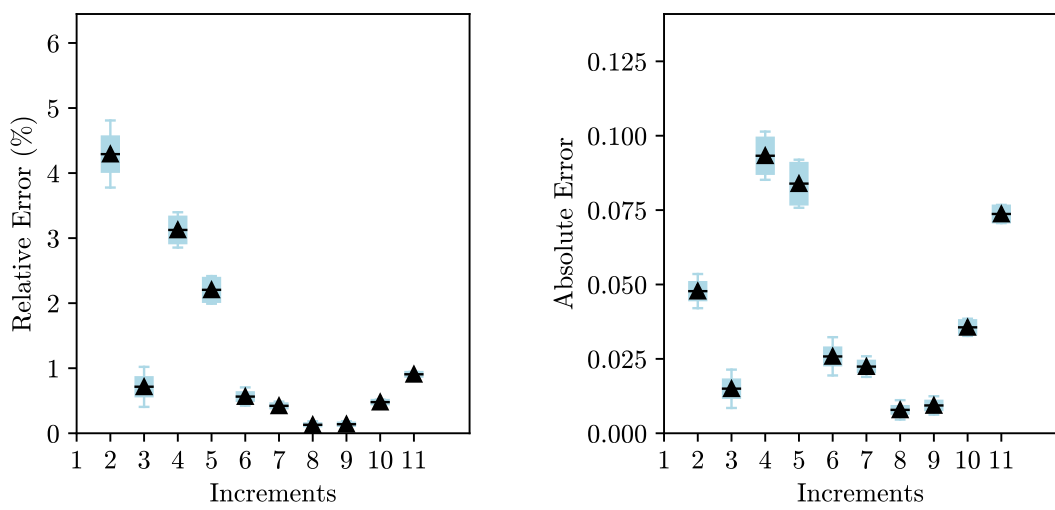


(a) UMAT (left) and ML (right)



(b) Absolute difference

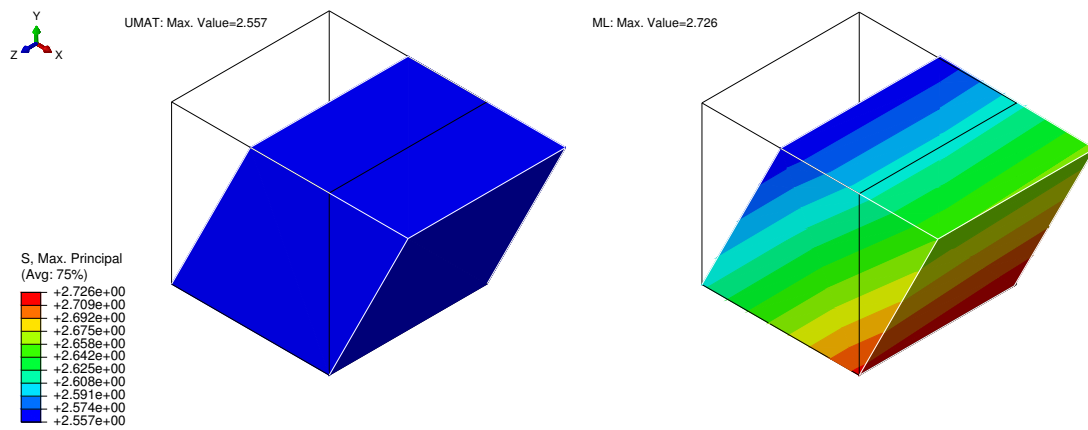
Figure 5.9: Displacement magnitude - Cube with biaxial load



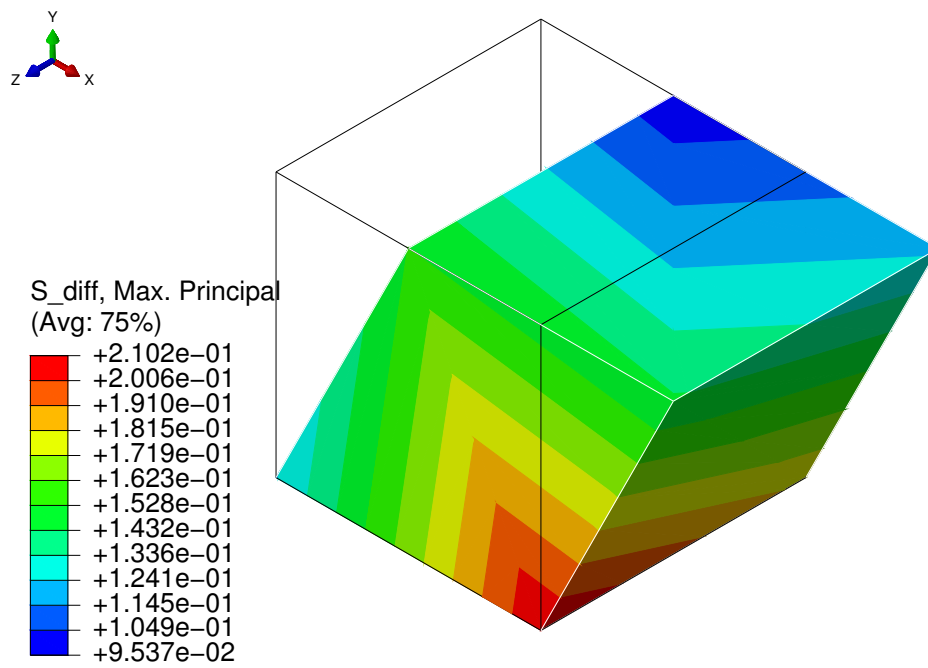
(a) Relative error

(b) Absolute error

Figure 5.10: Error - Cube with biaxial load

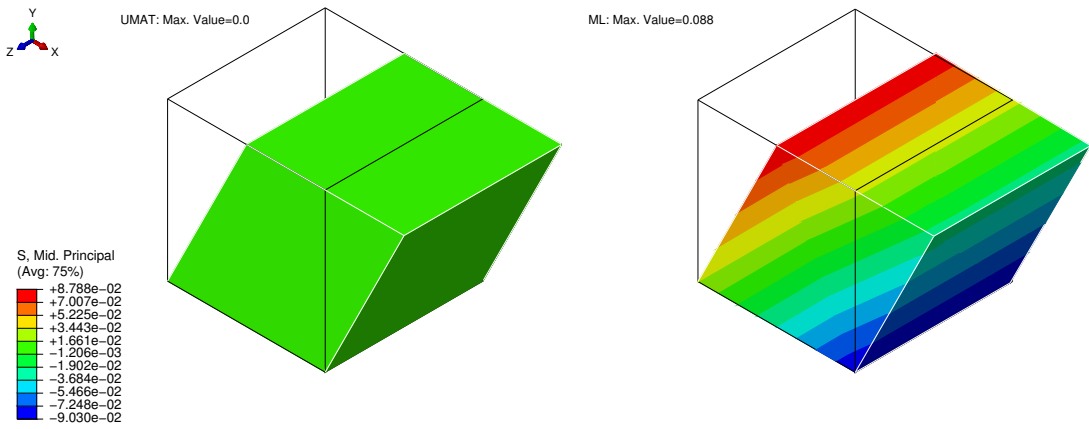


(a) UMAT (left) and ML (right)

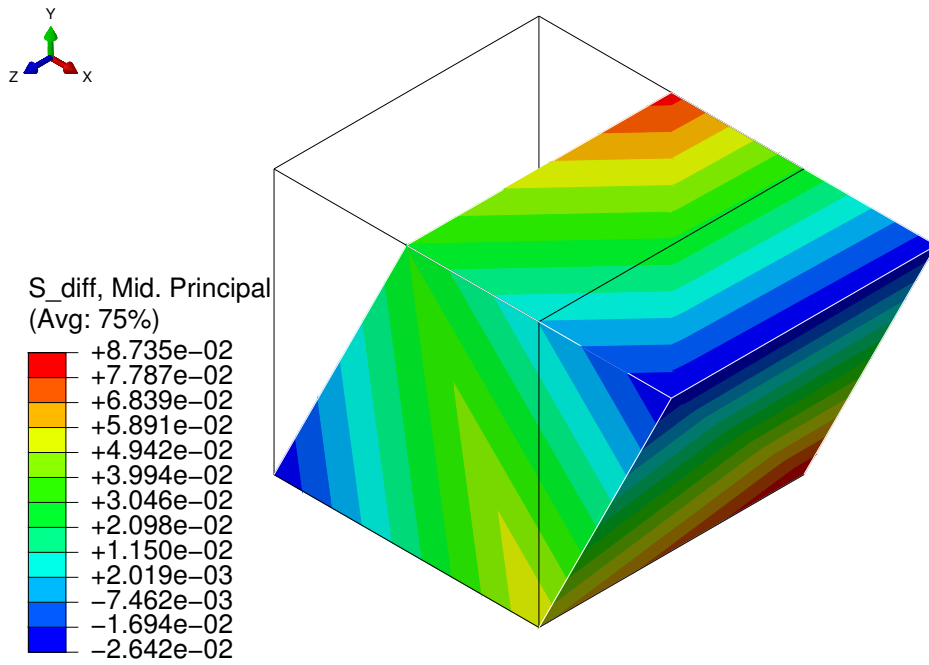


(b) Absolute difference

Figure 5.11: Maximum principal stress - Cube with simple shear load

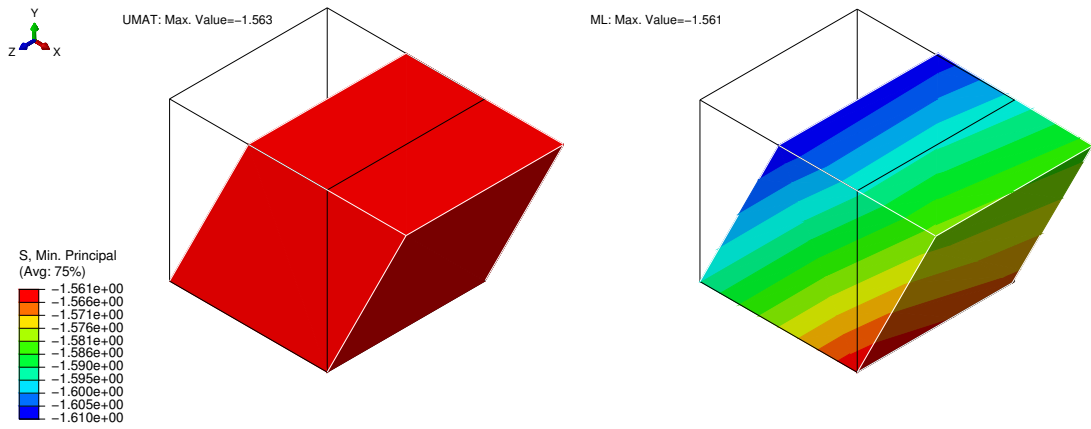


(a) UMAT (left) and ML (right)

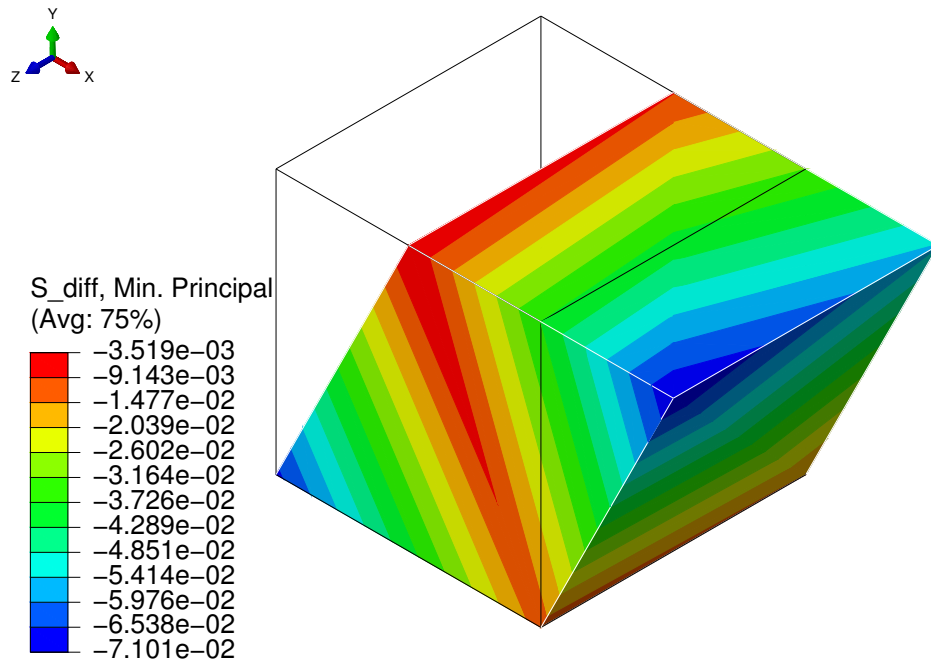


(b) Absolute difference

Figure 5.12: Middle principal stress - Cube with simple shear load

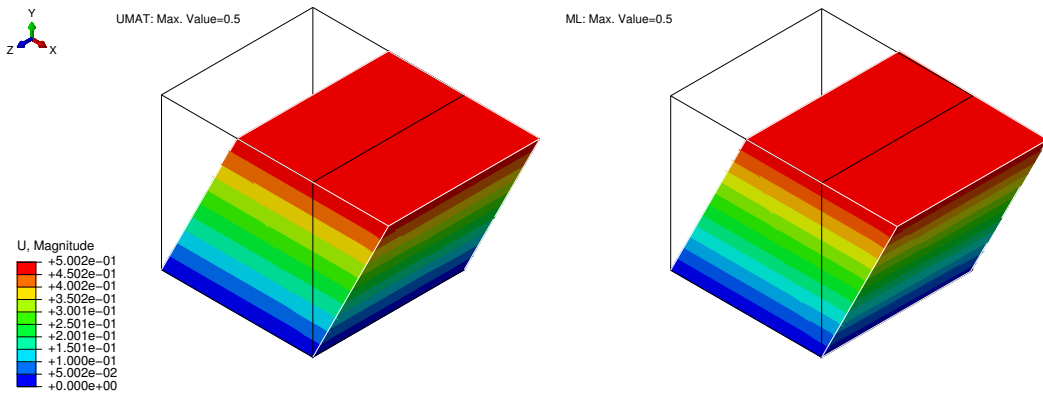


(a) UMAT (left) and ML (right)

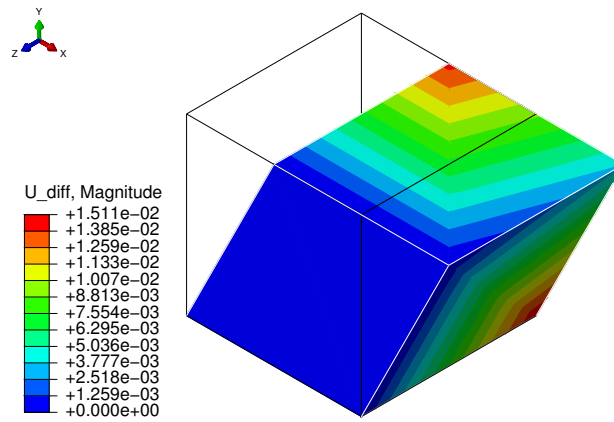


(b) Absolute difference

Figure 5.13: Minimum principal stress - Cube with simple shear load

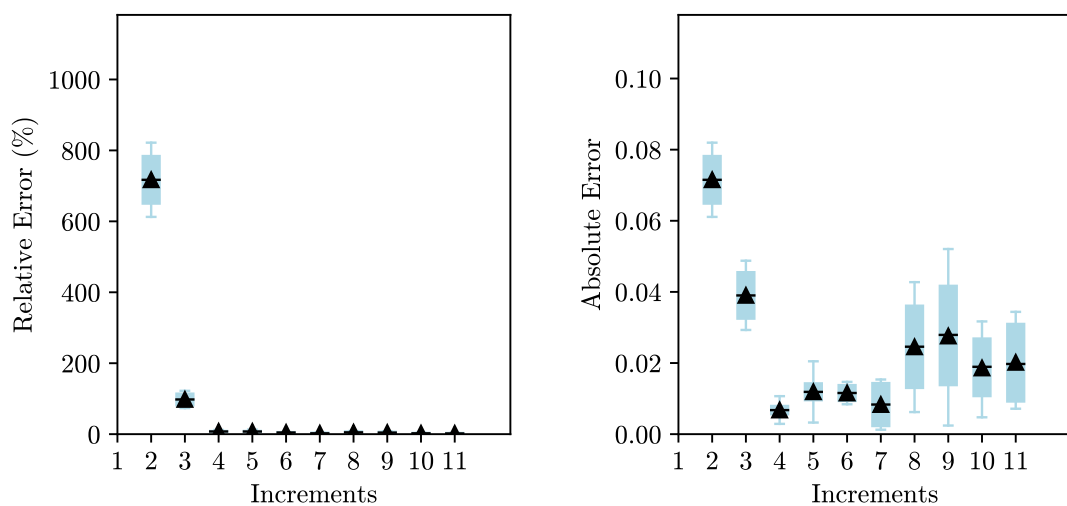


(a) UMAT (left) and ML (right)



(b) Absolute difference

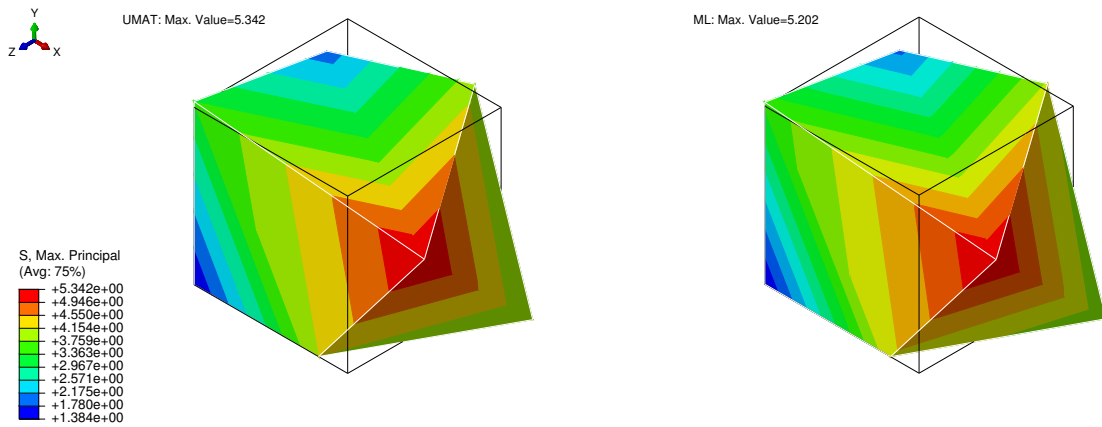
Figure 5.14: Displacement magnitude - Cube with simple shear load



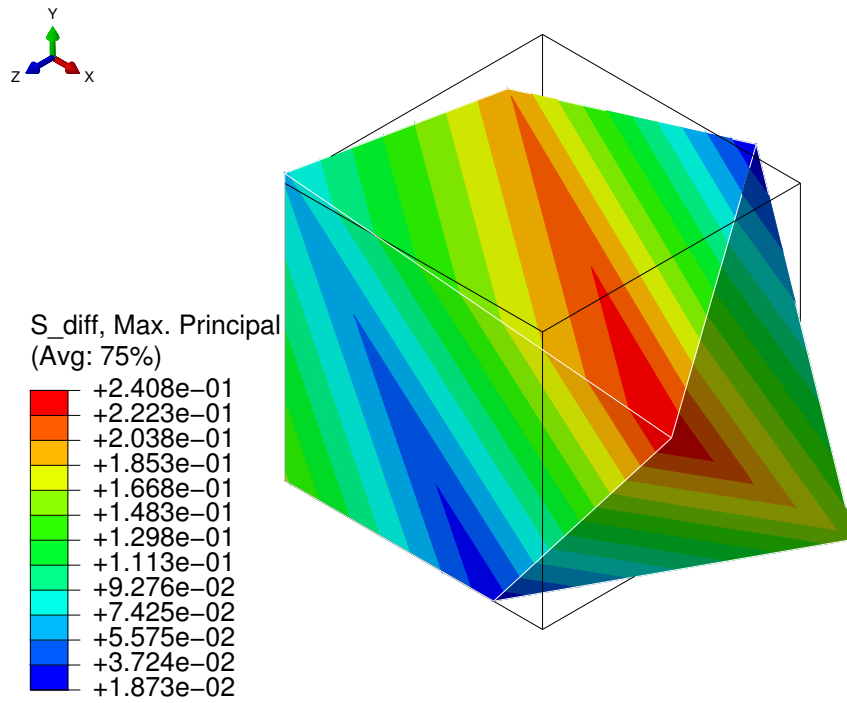
(a) Relative error

(b) Absolute error

Figure 5.15: Error - Cube with simple shear load

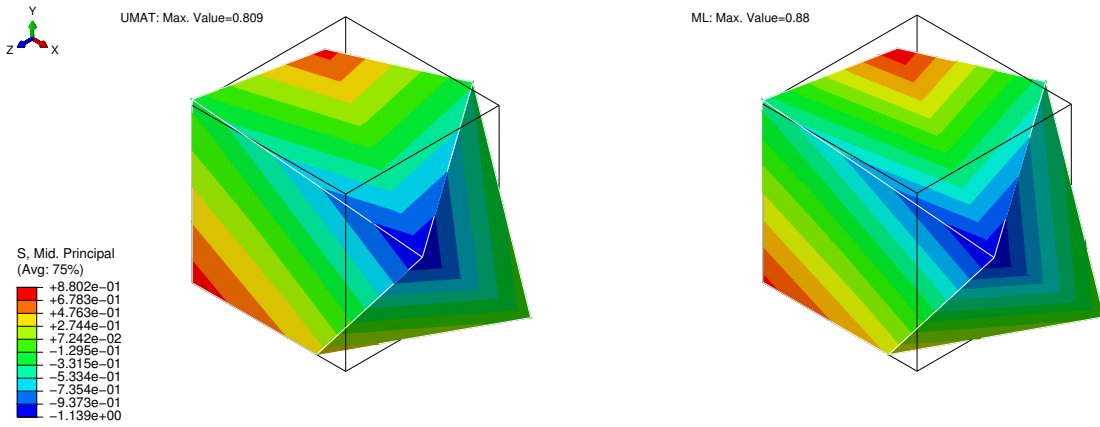


(a) UMAT (left) and ML (right)

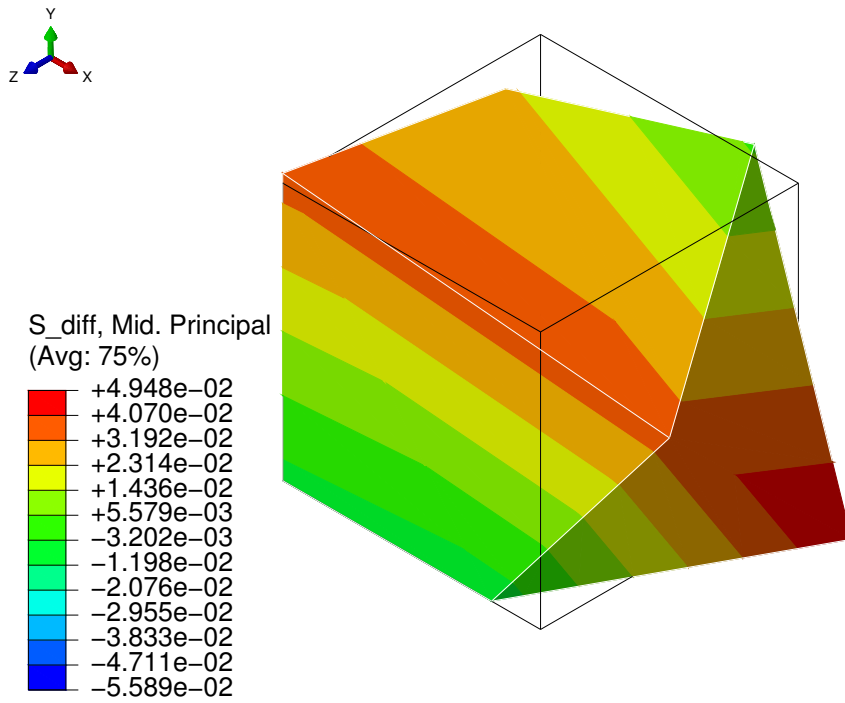


(b) Absolute difference

Figure 5.16: Maximum principal stress - Cube with random deformation

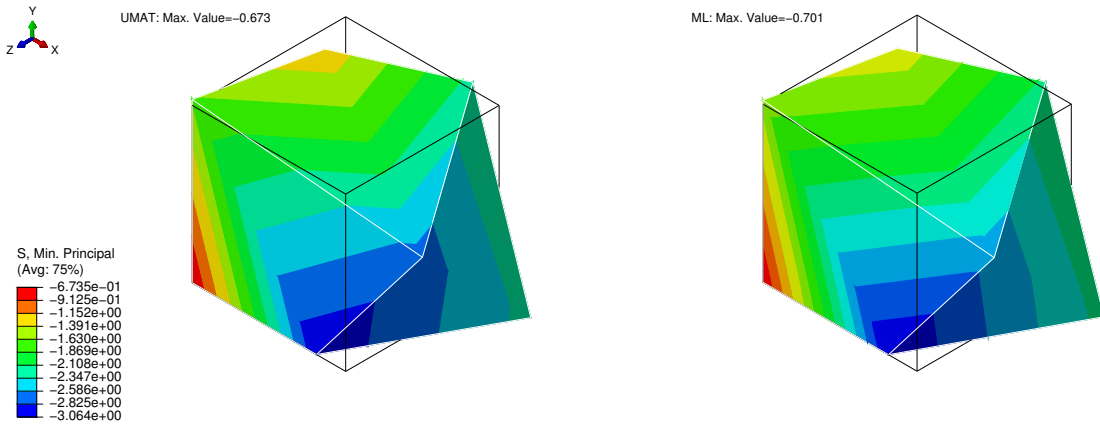


(a) UMAT (left) and ML (right)

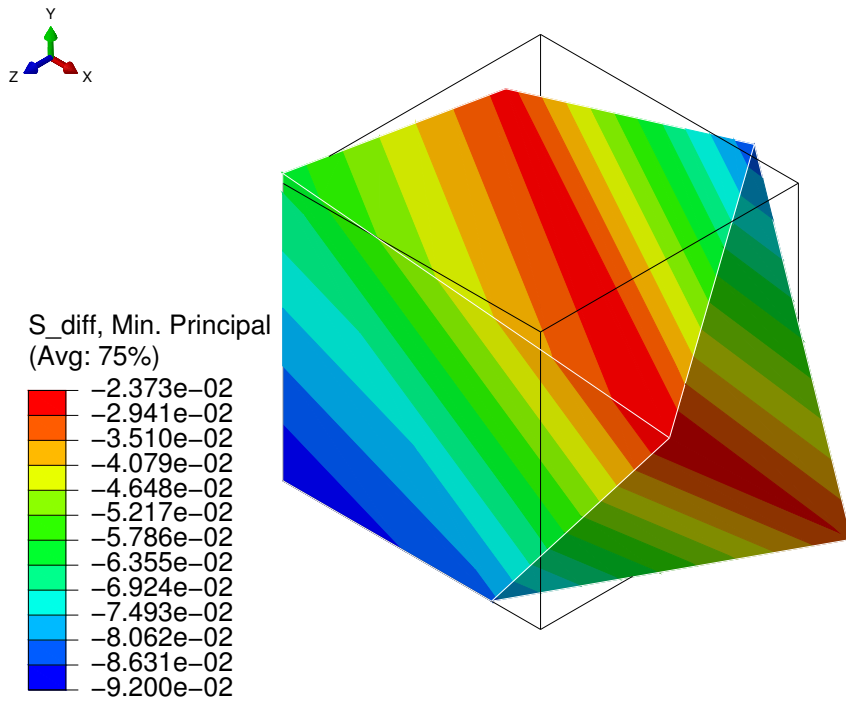


(b) Absolute difference

Figure 5.17: Middle principal stress - Cube with random deformation

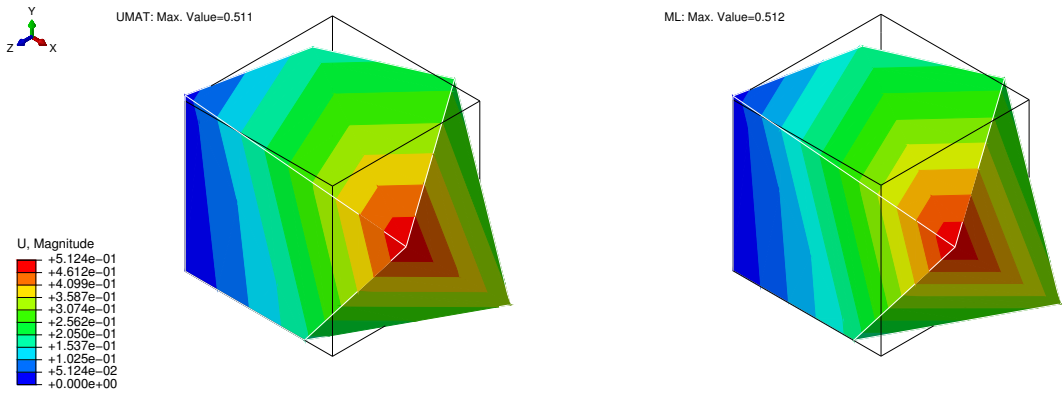


(a) UMAT (left) and ML (right)

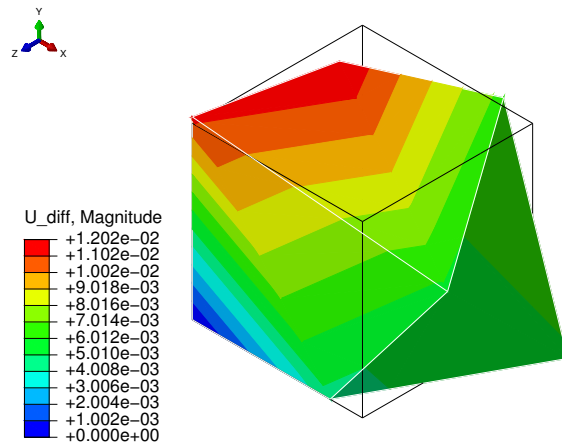


(b) Absolute difference

Figure 5.18: Minimum principal stress - Cube with random deformation

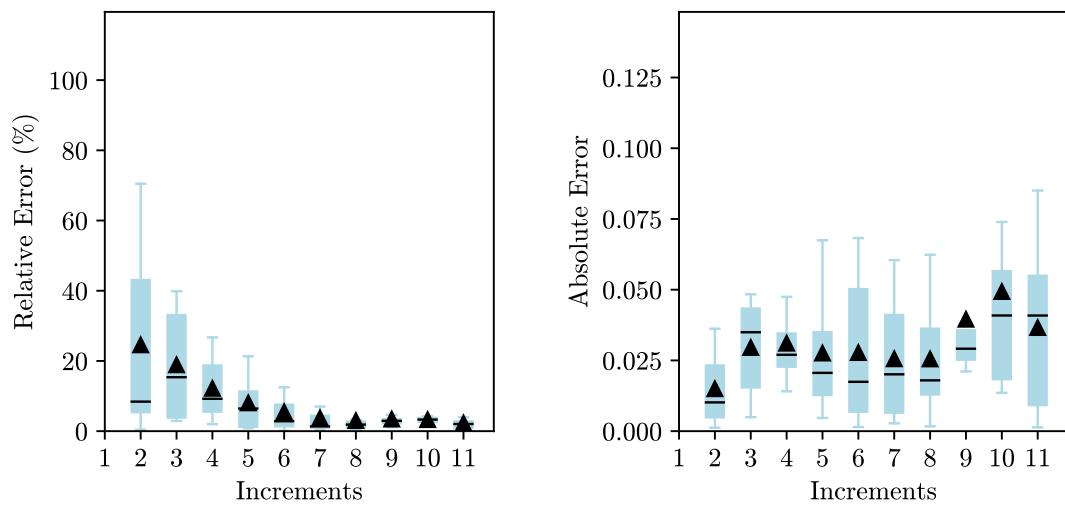


(a) UMAT (left) and ML (right)



(b) Absolute difference

Figure 5.19: Displacement magnitude - Cube with random deformation



(a) Relative error

(b) Absolute error

Figure 5.20: Error - Cube with random deformation

It is possible to observe that the contour plots of the stresses have a considerable difference. With the conventional UMAT, a homogeneous colour is obtained, indicating that the stresses are equal in all the integration points of the cube, whereas, with the ML model, the contour plot shows different colours. However, the legend shows that the results are similar, with small differences. In addition, the principal stresses were the ones chosen to be displayed, which also has some influence on the results shown because there are residual values predicted by the surrogate models for components of the stress tensor that are in fact zero (as an example, for a uniaxial loading case, the ML approach gives some residual values for the shear components when they are zero in fact).

In terms of displacement magnitude, the values obtained have only differences in the third decimal place and the contour plots are nearly equal. It was expected that the results of the displacement were nearly equal because the boundary conditions consisted of imposed displacements. The differences in the magnitude are related mainly to the computation of the tangent stiffness matrix, which has an influence on the displacements and the stresses. The relative difference between all the integration points and all the stress components shows some high values. A possible reason for such high values is that at the beginning of the analysis, the stress values are small, which means that even for a small absolute error, the relative error is high. This hypothesis is supported by a box-and-whisker plot of the absolute error that shows that, in fact, the absolute error has a small value and by the decreasing trend of the relative error as the number of increments increases.

5.2.2 Block with Pressure

In this problem, a block is subjected to a pressure load on part of its upper surface, which increases the complexity of the validation and tests the developed model in a problem with singularities.

The displacements are fixed in the x- and z-directions on the upper surface and fixed in the y-direction at the bottom. The symmetry of the block was considered, which implied some additional Dirichlet boundary conditions: fixed displacements in the x-direction due to symmetry in the x-plane and fixed displacements in the z-direction due to symmetry in z-plane. The block is loaded partially in the upper surface with a constant pressure load p , leading to a mixed Dirichlet-Neumann boundary condition [75].

The dimensions and the boundary conditions applied to one-quarter of the block are shown in Figure 5.21. A mesh of 10 x 10 x 10 hexahedral elements with 8 linear nodes was used and two analyses were performed, one with pressure and another with suction pressure. The results obtained for a suction pressure load ($p = 3$) and for a pressure load ($p = -6$) are shown in Figures 5.22 to 5.26 and Figures 5.27 to 5.31, respectively.

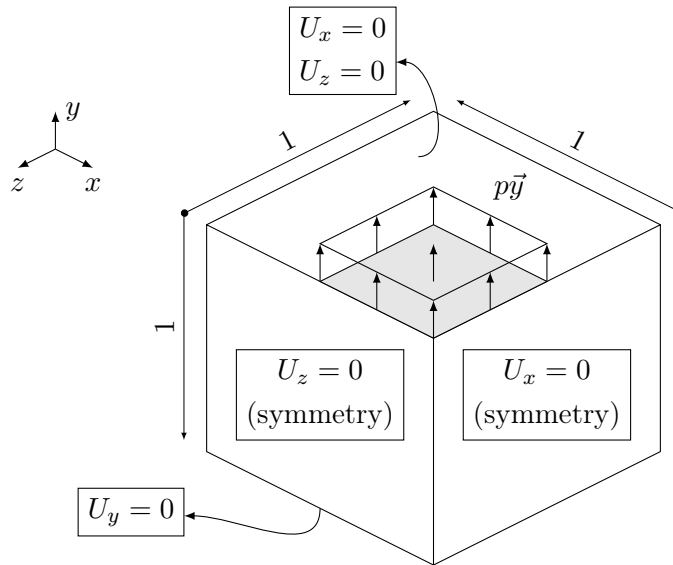
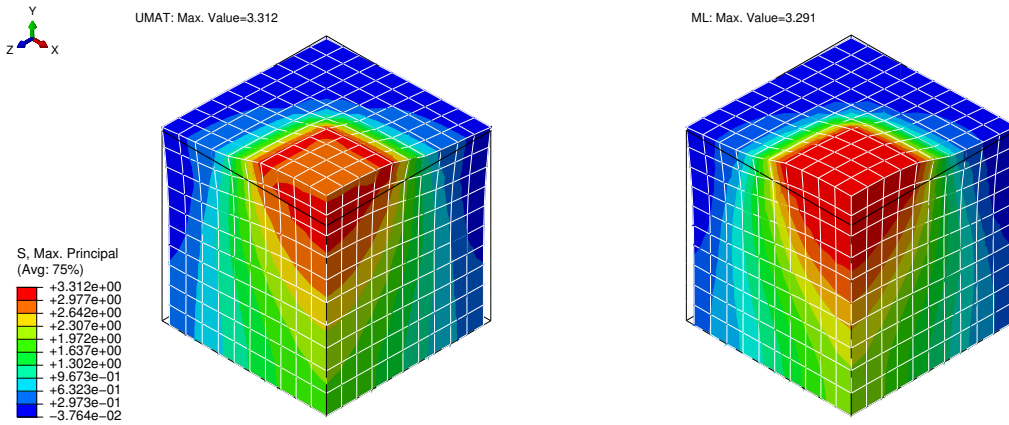
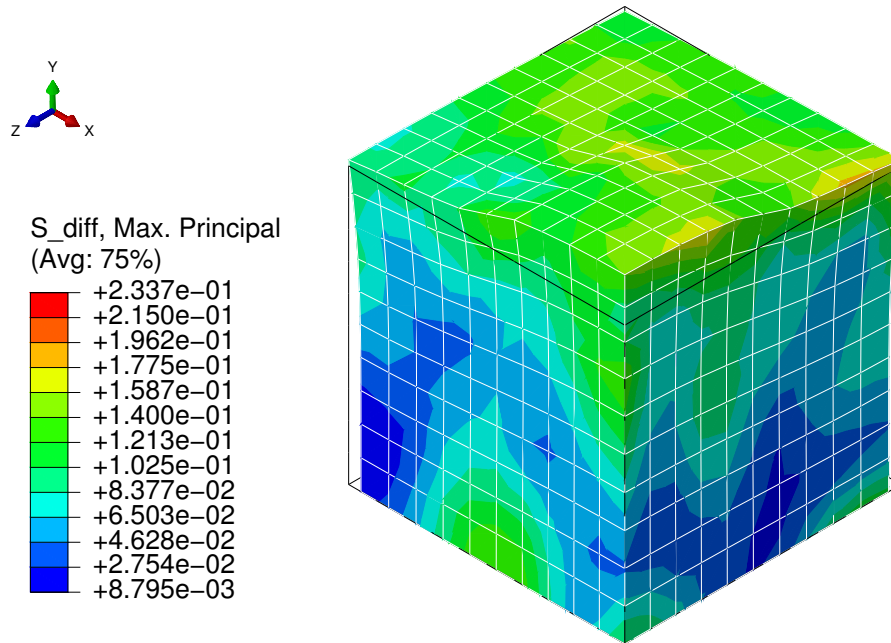


Figure 5.21: Geometry and boundary conditions - Block under partial load

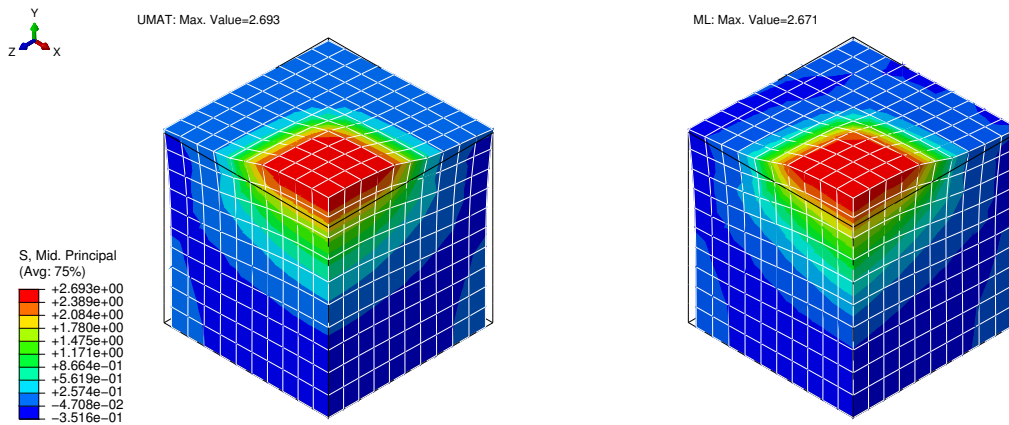


(a) UMAT (left) and ML (right)

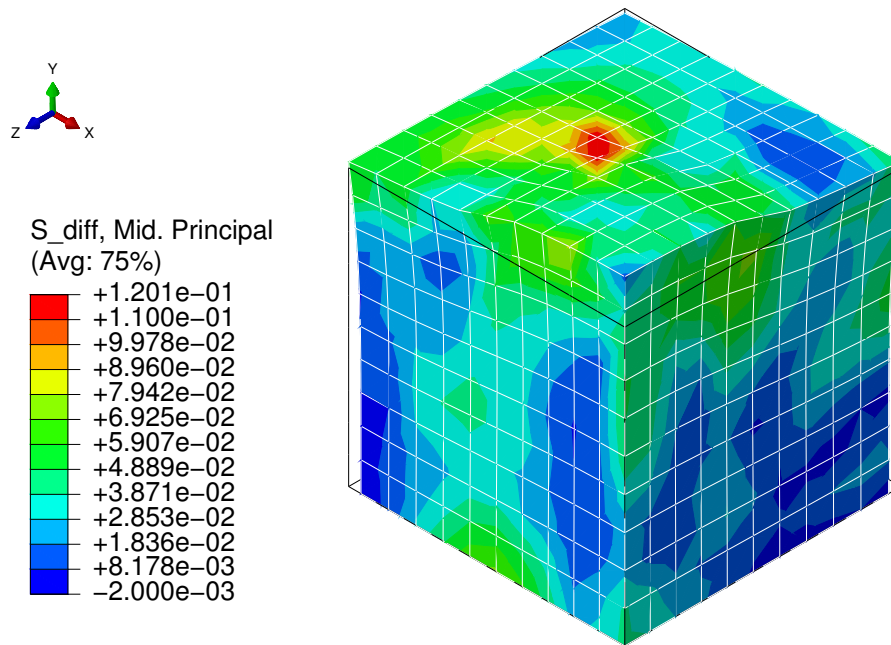


(b) Absolute difference

Figure 5.22: Maximum principal stress - Block under suction

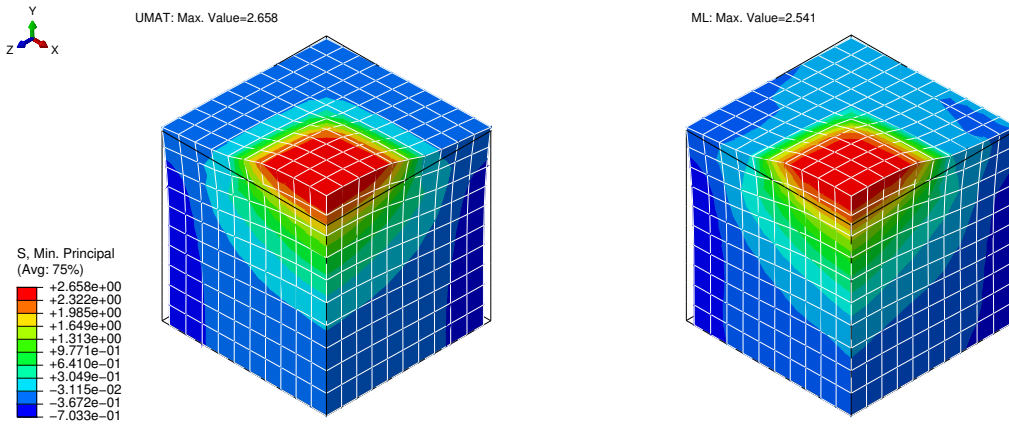


(a) UMAT (left) and ML (right)

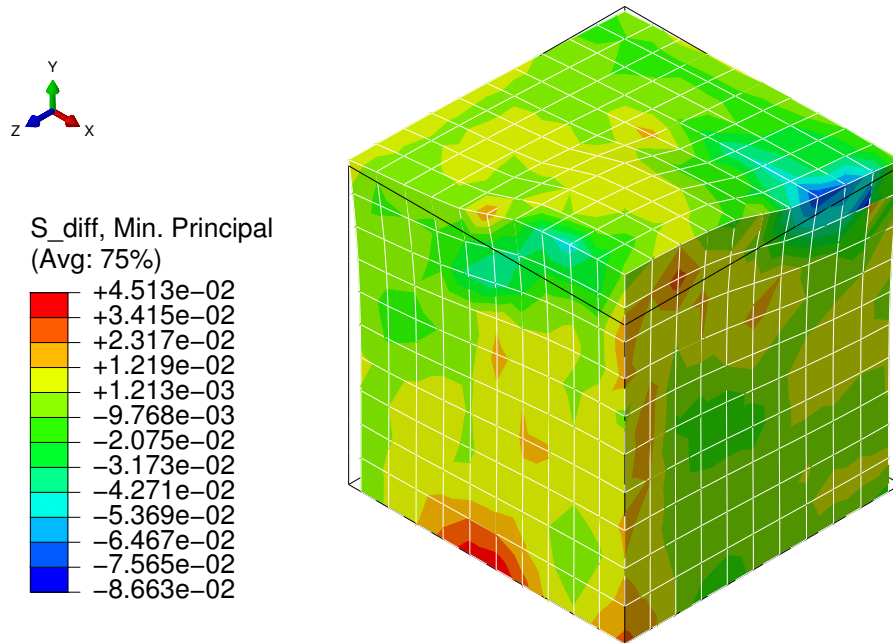


(b) Absolute difference

Figure 5.23: Middle principal stress - Block under suction

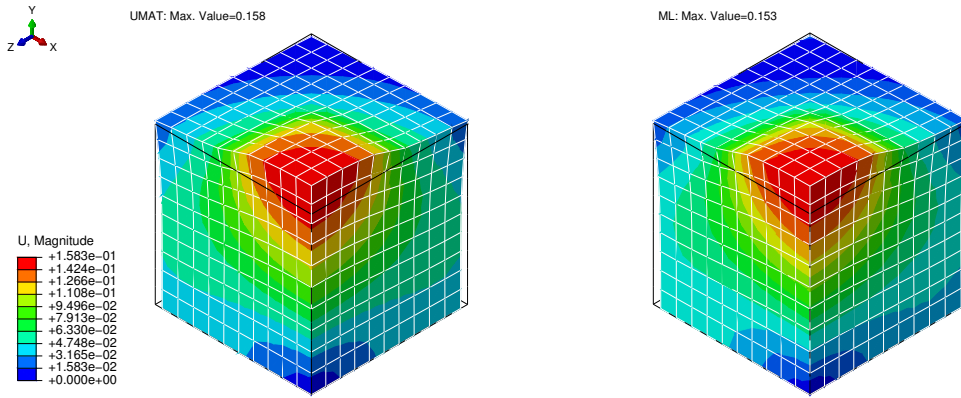


(a) UMAT (left) and ML (right)

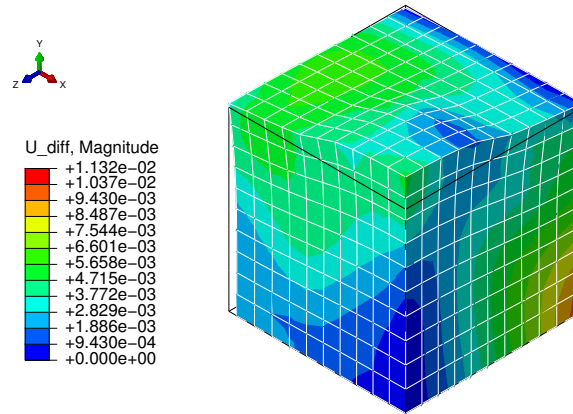


(b) Absolute difference

Figure 5.24: Minimum principal stress - Block under suction

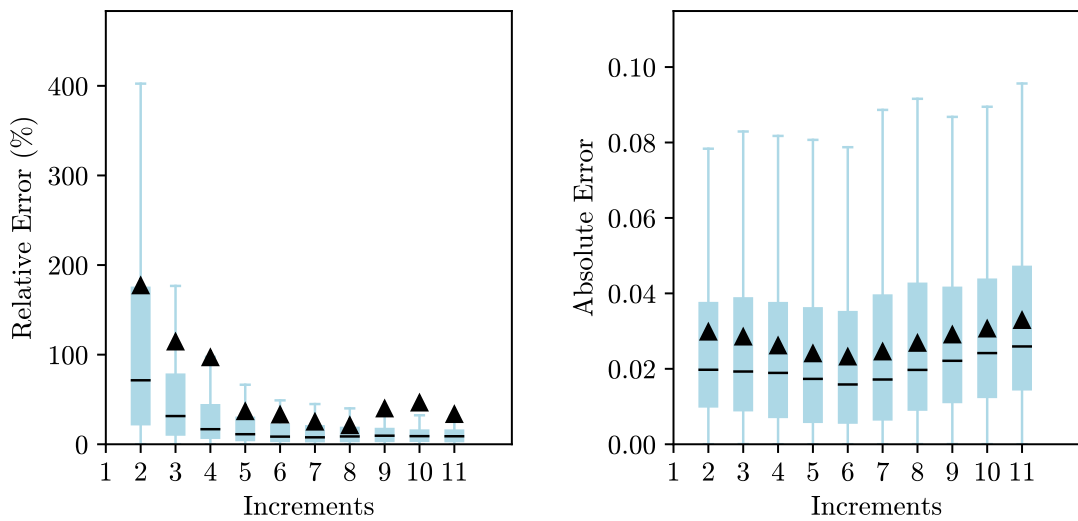


(a) UMAT (left) and ML (right)



(b) Absolute difference

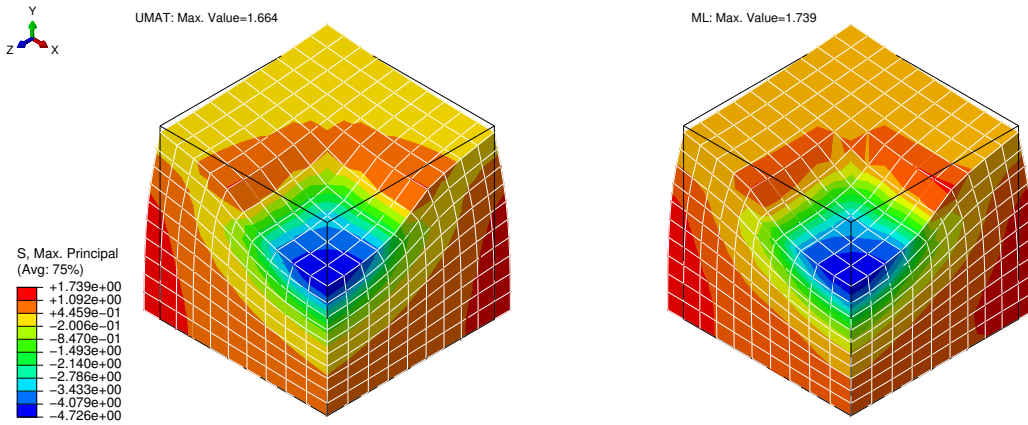
Figure 5.25: Displacement magnitude - Block under suction



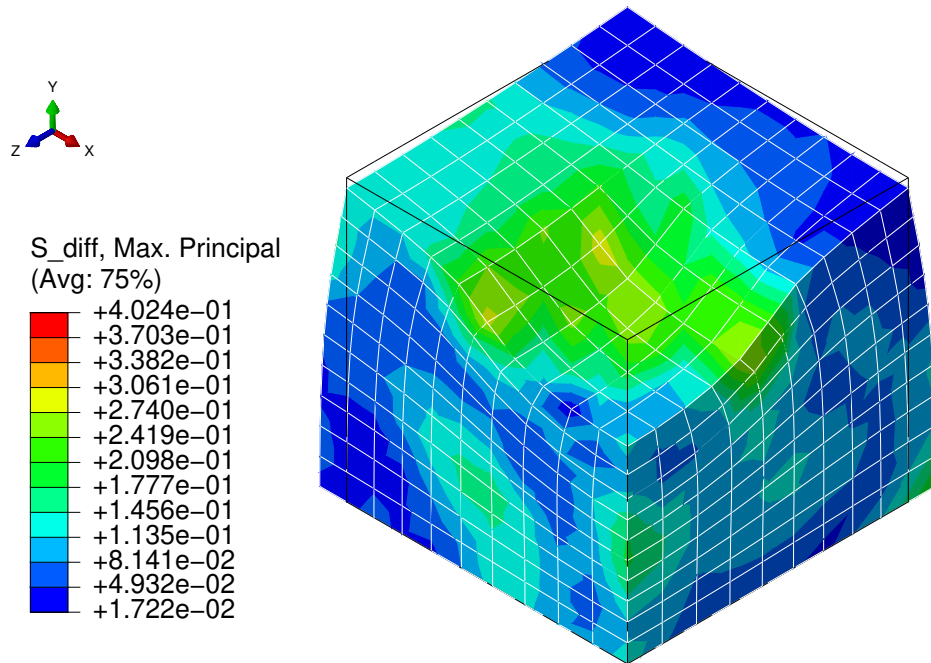
(a) Relative error

(b) Absolute error

Figure 5.26: Errors - Block under suction

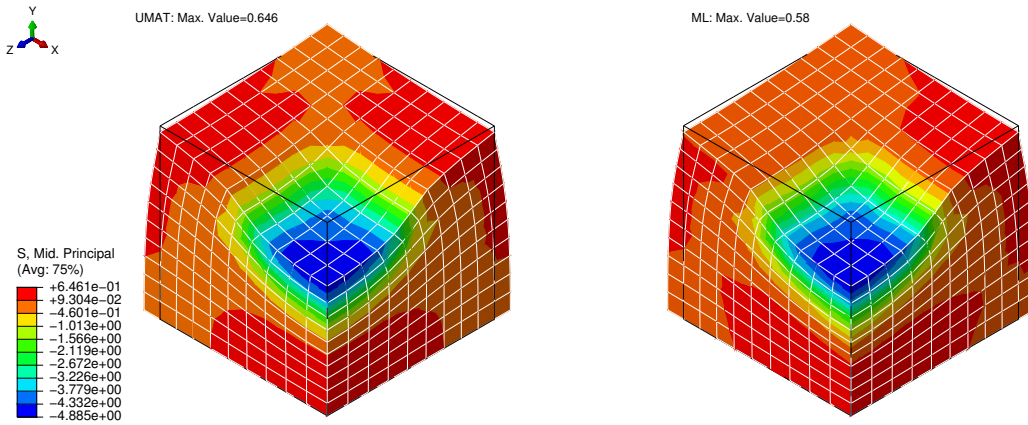


(a) UMAT (left) and ML (right)

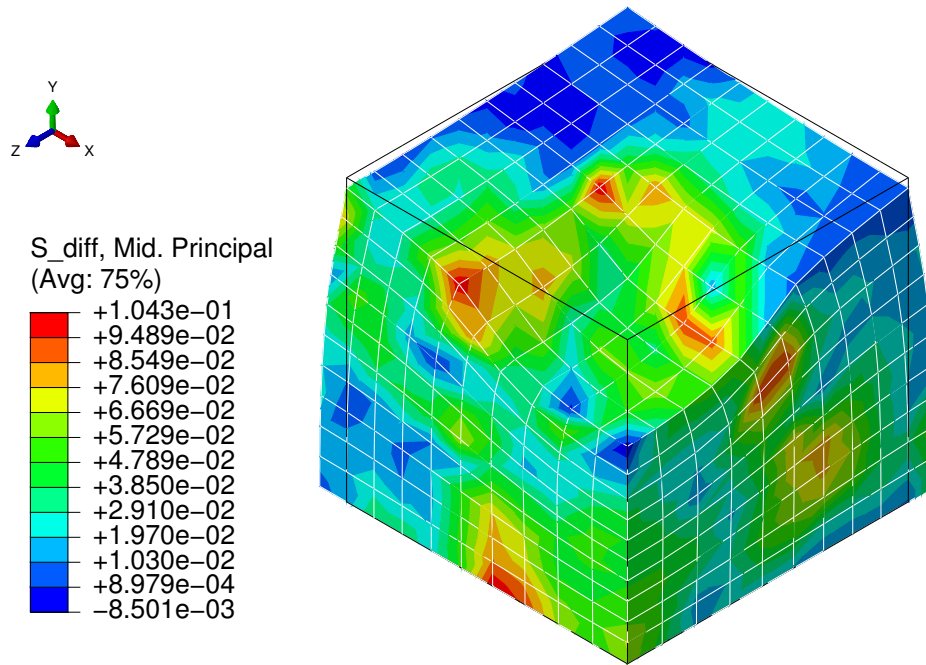


(b) Absolute difference

Figure 5.27: Maximum principal stress - Block under pressure

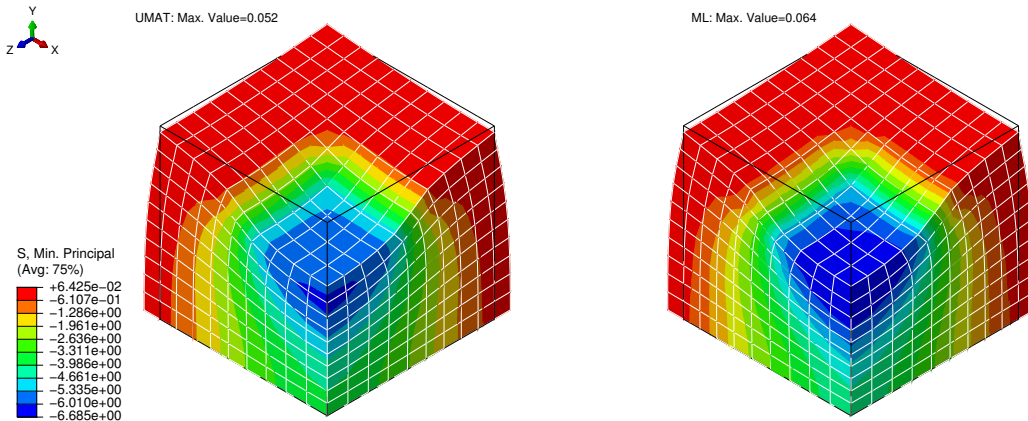


(a) UMAT (left) and ML (right)

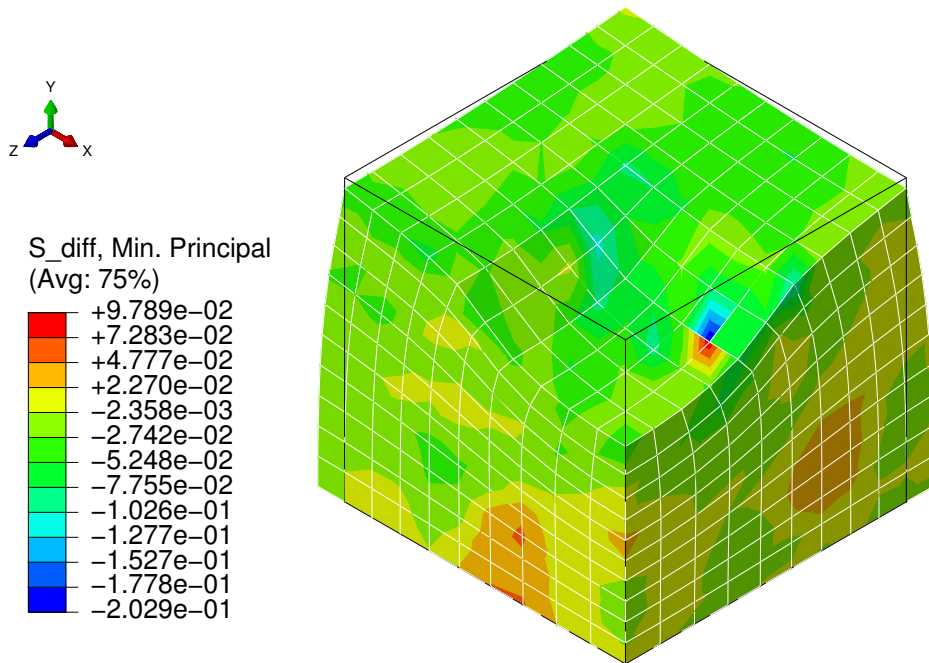


(b) Absolute difference

Figure 5.28: Middle principal stress - Block under pressure

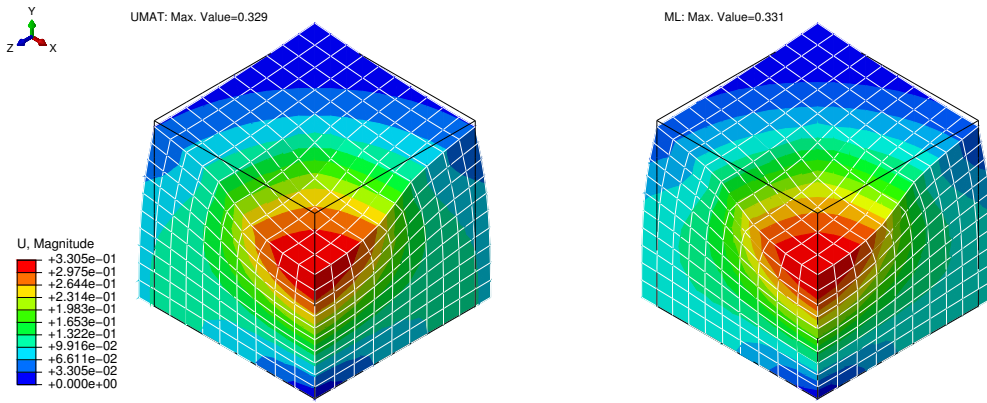


(a) UMAT (left) and ML (right)

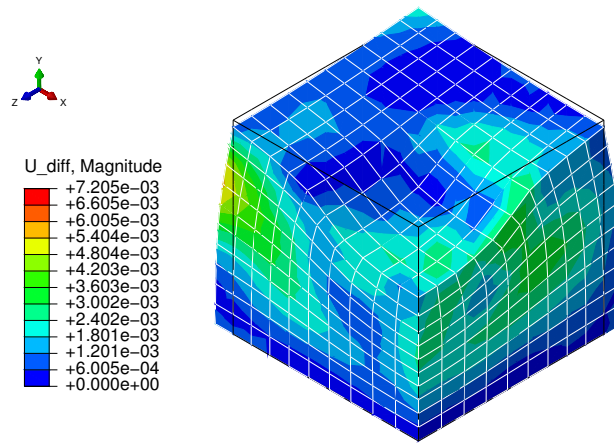


(b) Absolute difference

Figure 5.29: Minimum principal stress - Block under pressure

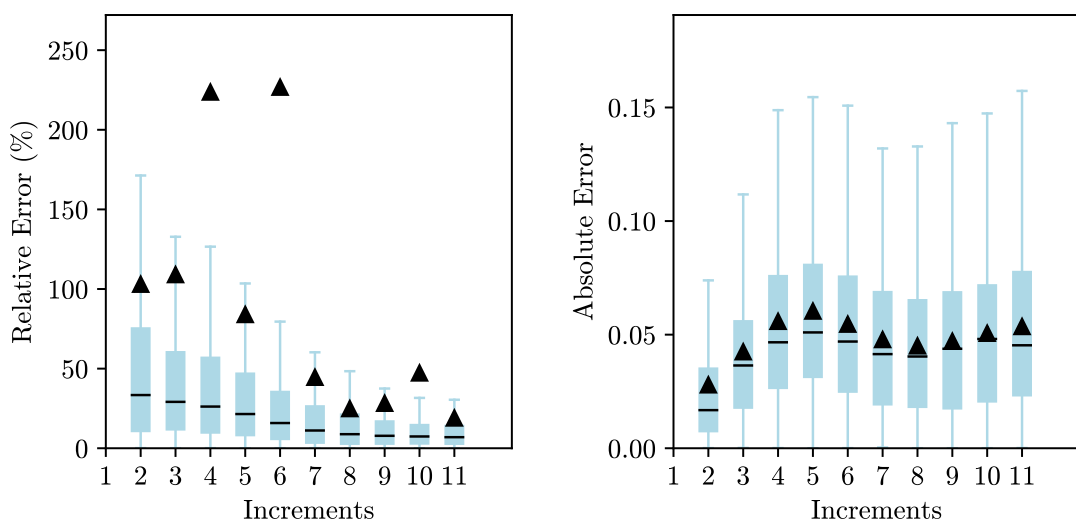


(a) UMAT (left) and ML (right)



(b) Absolute difference

Figure 5.30: Displacement magnitude - Block under pressure



(a) Relative error

(b) Absolute error

Figure 5.31: Errors - Block under pressure

The block under partial load example introduces more complexity, not only because it has more elements, but also because of the boundary conditions imposed. In these examples, the contour plots of the stresses have really similar colour maps and the maximum value obtained with the two approaches has a small relative error. Despite the fact that in these examples there is a pressure applied, the displacement magnitude is again nearly identical. Regarding the box-and-whisker plots of the errors, they have the same behaviour observed for the unitary cube: high relative error, mainly at the beginning of the analysis and with a decreasing trend and small values for the absolute error.

5.2.3 Cook's Membrane

The Cook's membrane is a typical benchmark problem in solid mechanics, which combines a bending and shearing response with moderate distortion. It was introduced by Cook [76] and it consists of a tapered cantilever with one side clamped and a constant shear load in the vertical direction applied on the opposite side. The geometry and the boundary conditions are shown in Figure 5.32 [75].

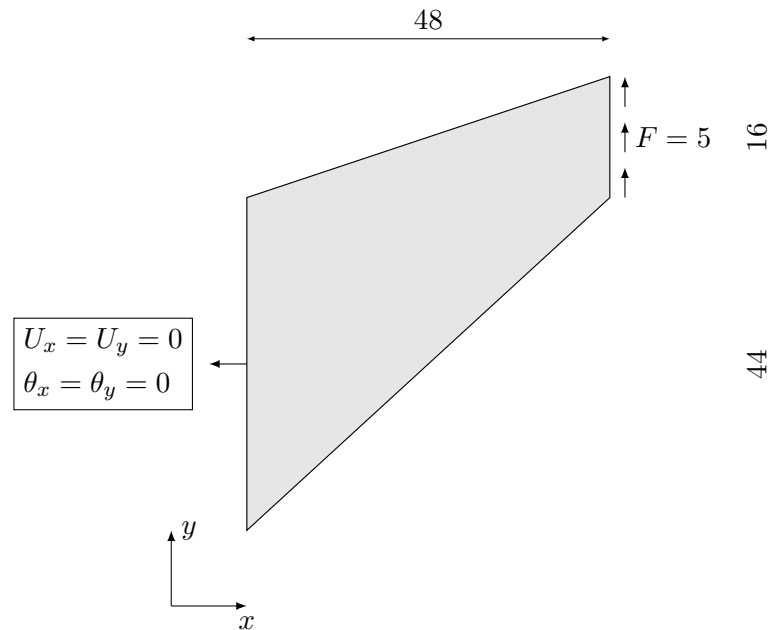


Figure 5.32: Geometry and boundary conditions - Cook's membrane with 5 of thickness

It was used a mesh of hexahedral elements with 8 linear nodes with 32 elements on each side of the membrane. Four elements through the thickness were used and a concentrated load of 5 was applied [77]. The used mesh is shown in Figure 5.33 and the results obtained are displayed in Figures 5.34 to 5.38.

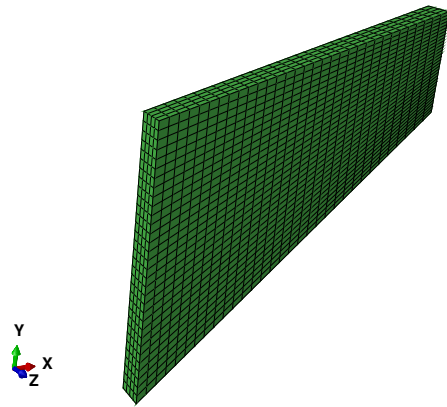
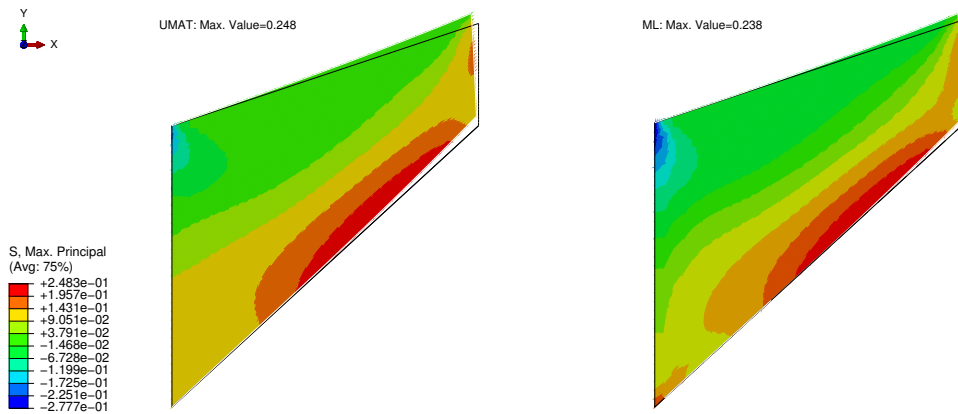
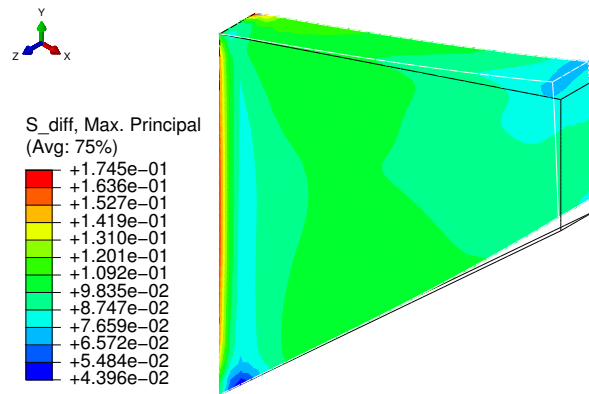


Figure 5.33: Mesh - Cook's membrane

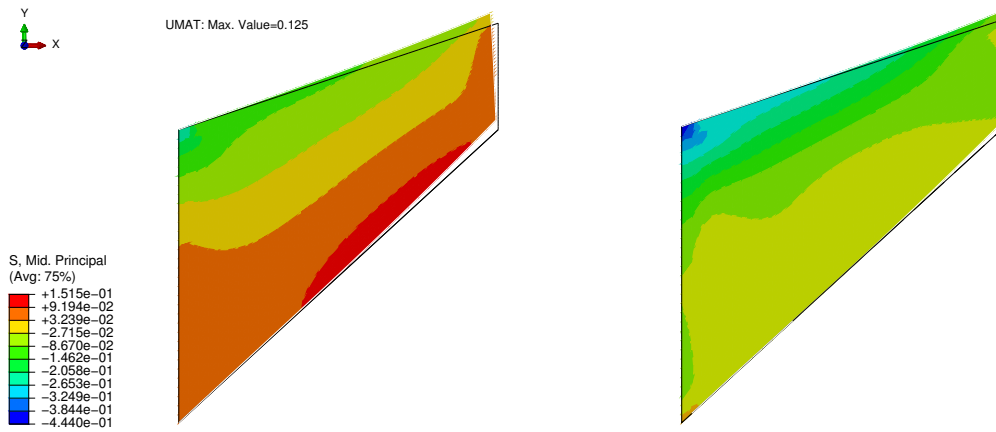


(a) UMAT (left) and ML (right)

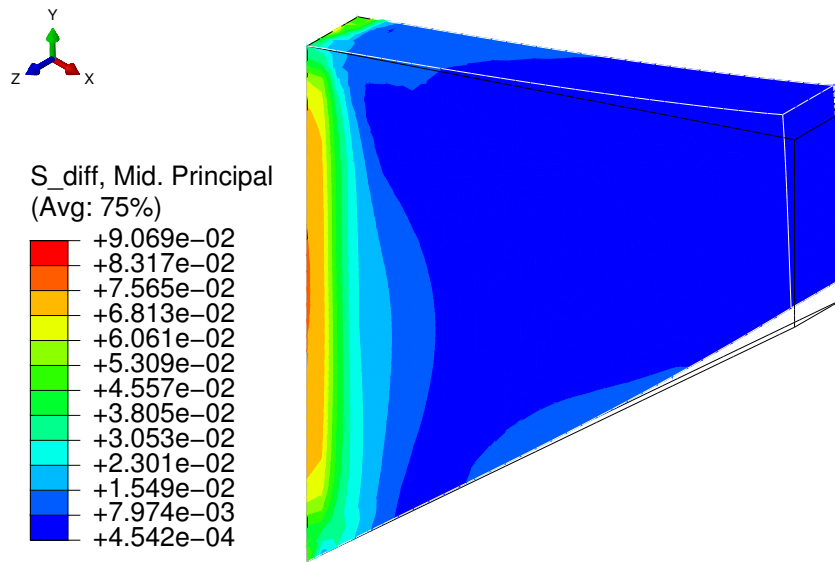


(b) Absolute difference

Figure 5.34: Maximum principal stress - Cook's membrane

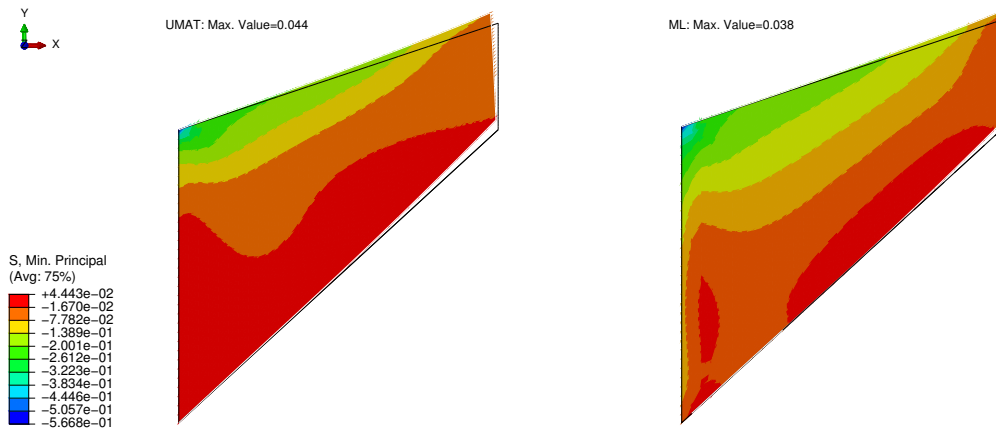


(a) UMAT (left) and ML (right)

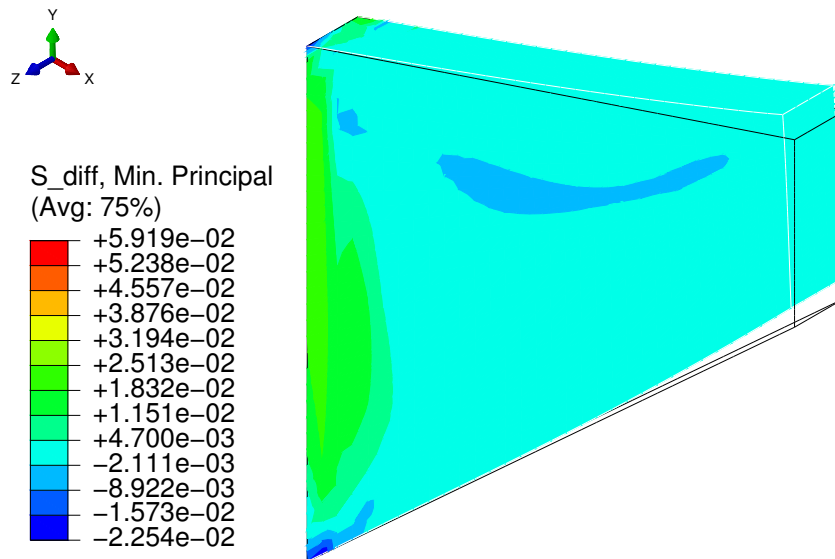


(b) Absolute difference

Figure 5.35: Middle principal stress - Cook's membrane

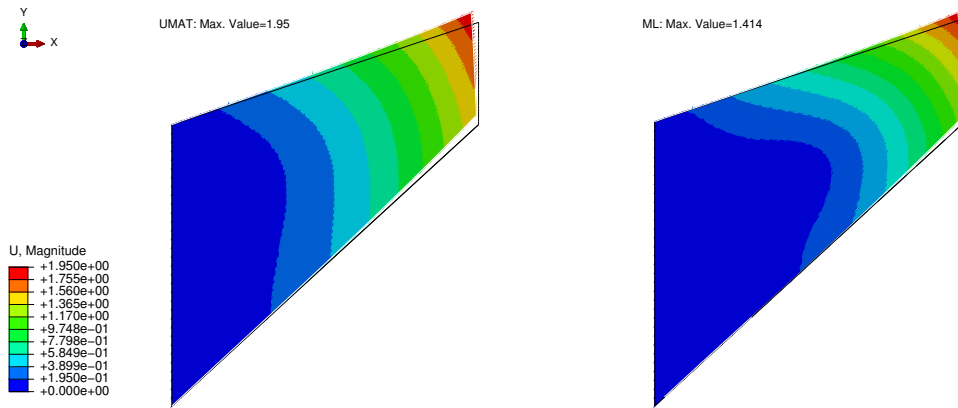


(a) UMAT (left) and ML (right)

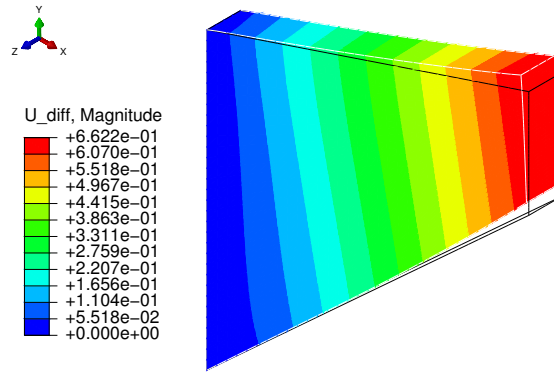


(b) Absolute difference

Figure 5.36: Minimum principal stress - Cook's membrane

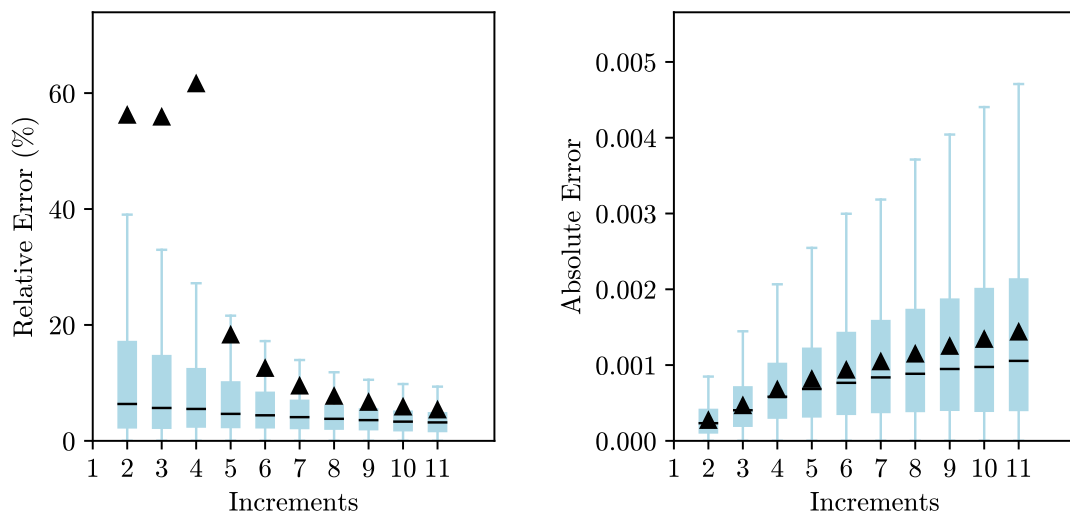


(a) UMAT (left) and ML (right)



(b) Absolute difference

Figure 5.37: Displacement magnitude - Cook's membrane



(a) Relative error

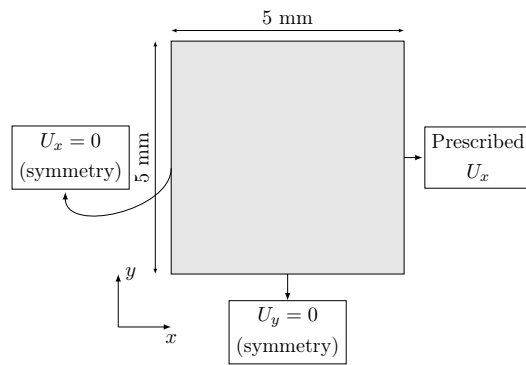
(b) Absolute error

Figure 5.38: Error - Cook's membrane

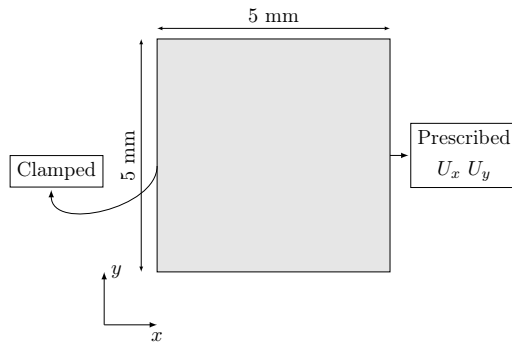
The results for Cook's membrane follow the conclusions taken for the two previous examples, with the main difference being the disparity in the displacement magnitude. In this problem, a force is applied on the right-hand side of the membrane and the displacement is not prescribed as it was the case of the examples of the unitary cube. The difference in displacement in this example is higher, possibly explained by the differences in the tangent stiffness matrix.

5.2.4 Rectangular Block - Uniaxial, Shear and Torsion

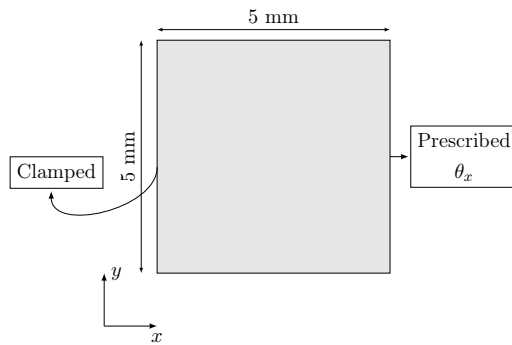
A rectangular block was subjected to three different loading conditions. The first is a uniaxial extension with a displacement in the x-direction ($U_x = 10$). The second one is a shear load, where the left side of the block is clamped and on the opposite side a displacement was applied in x- and y-direction ($U_x = U_y = 5$). The last one is a torsional load, where the left side is clamped again but on the right side, it was applied a rotation along the x-axis ($\theta_x = 1$). Hexahedral elements with 8 linear nodes were used and the geometry and the boundary conditions of the three examples are shown in Figure 5.39. The results are shown in Figures 5.40 to 5.44, in Figures 5.45 to 5.49 and in Figures 5.50 to 5.54 for the uniaxial, shear and torsional load, respectively.



(a) Uniaxial - $U_x = 10$

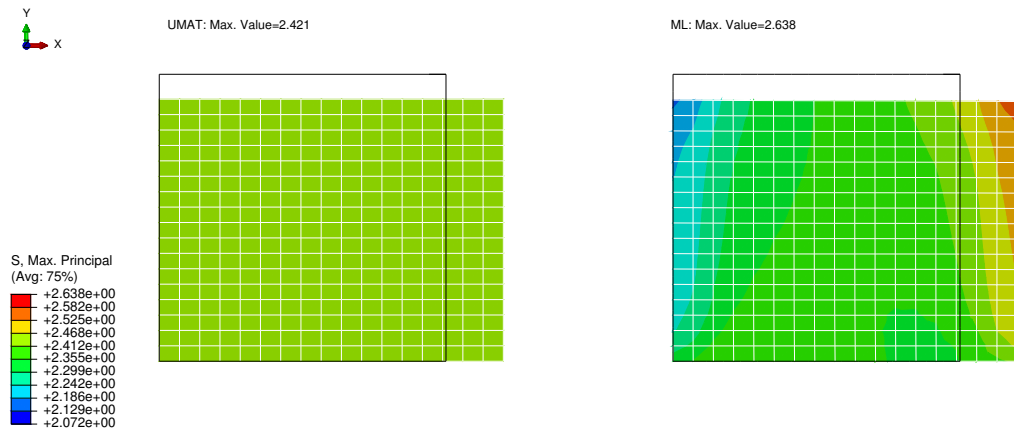


(b) Shear - $U_x = U_y = 5$

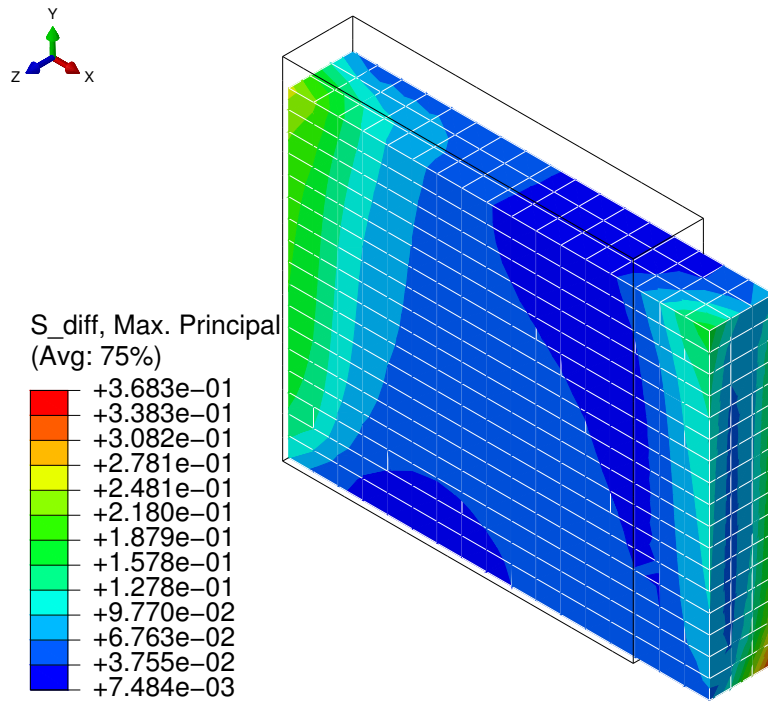


(c) Torsion - $\theta_x = 1$

Figure 5.39: Geometry and boundary conditions - Rectangular block with 1 of thickness

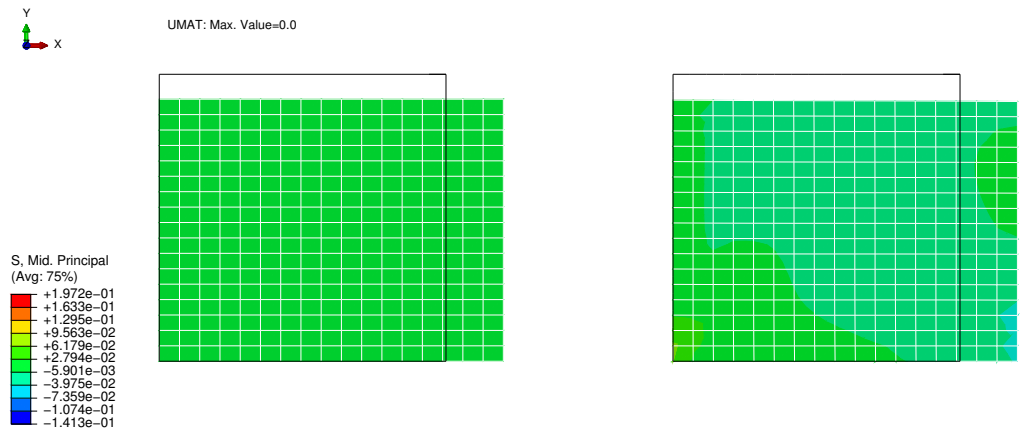


(a) UMAT (left) and ML (right)

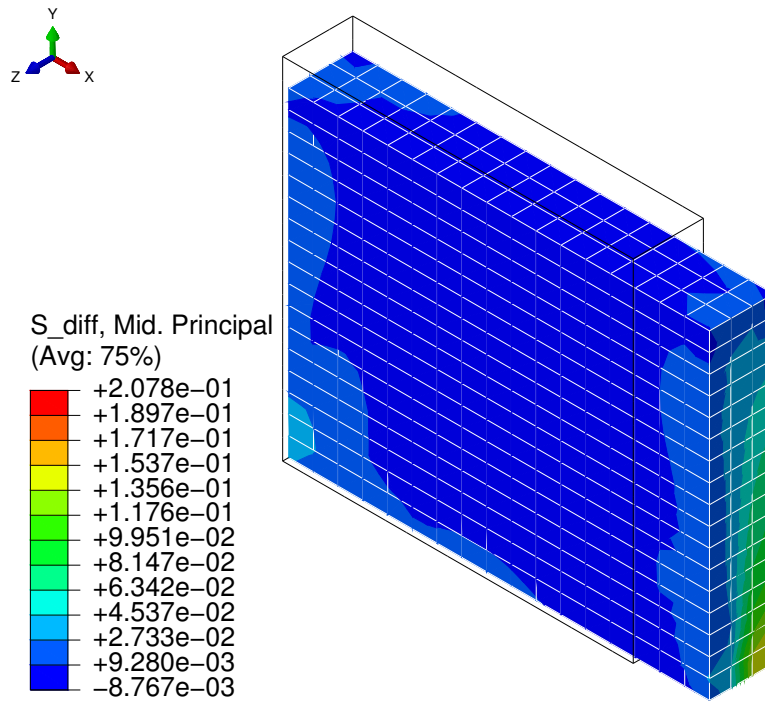


(b) Absolute difference

Figure 5.40: Maximum principal stress - Rectangular block under uniaxial load

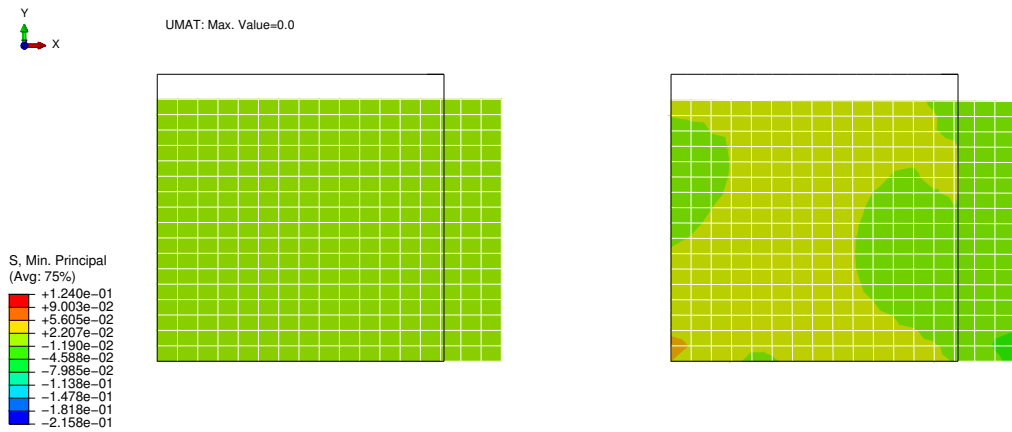


(a) UMAT (left) and ML (right)

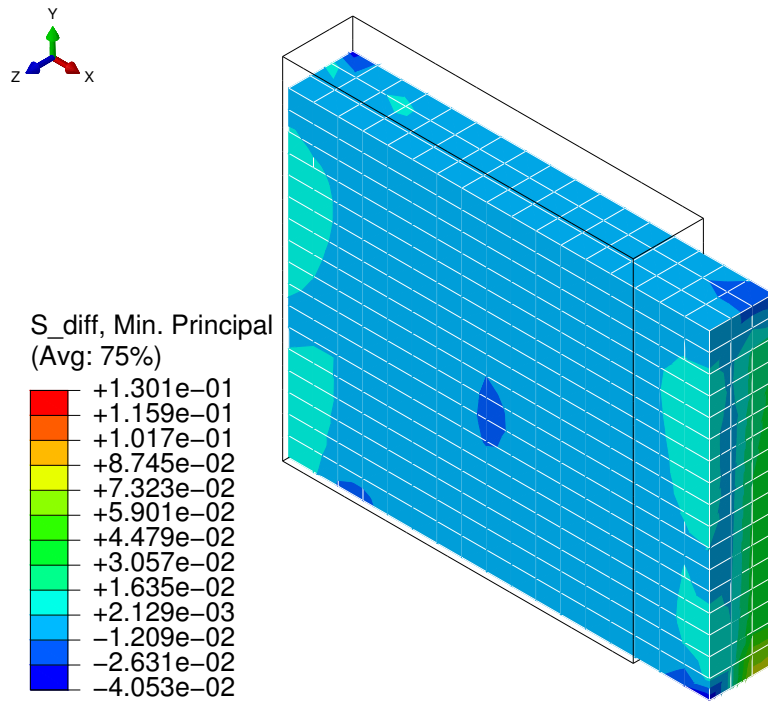


(b) Absolute difference

Figure 5.41: Middle principal stress - Rectangular block under uniaxial load



(a) UMAT (left) and ML (right)



(b) Absolute difference

Figure 5.42: Minimum principal stress - Rectangular block under uniaxial load

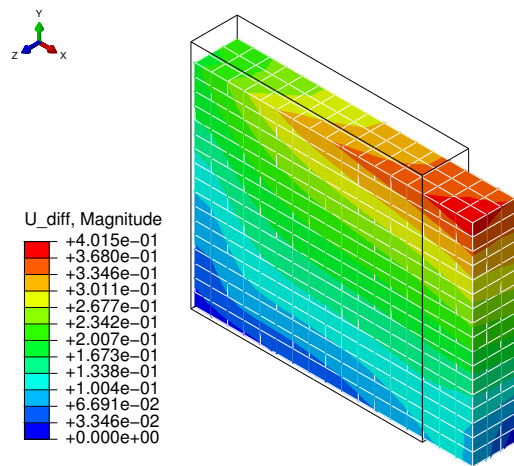
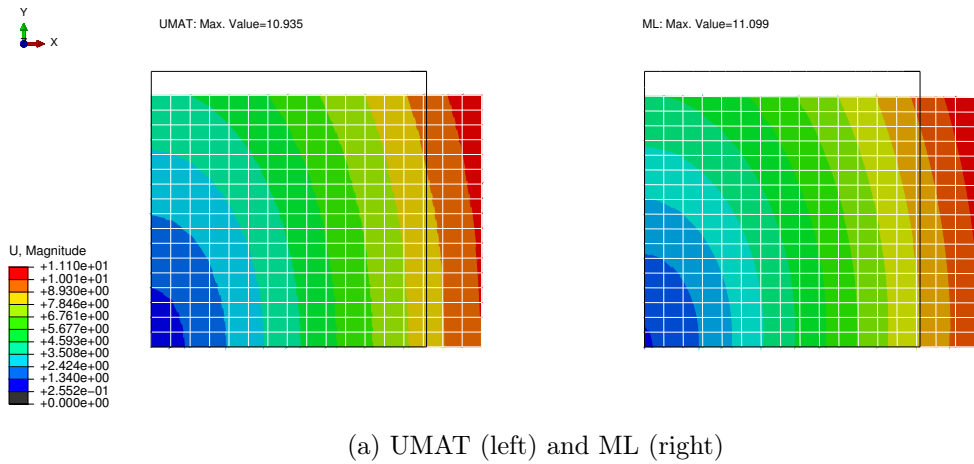
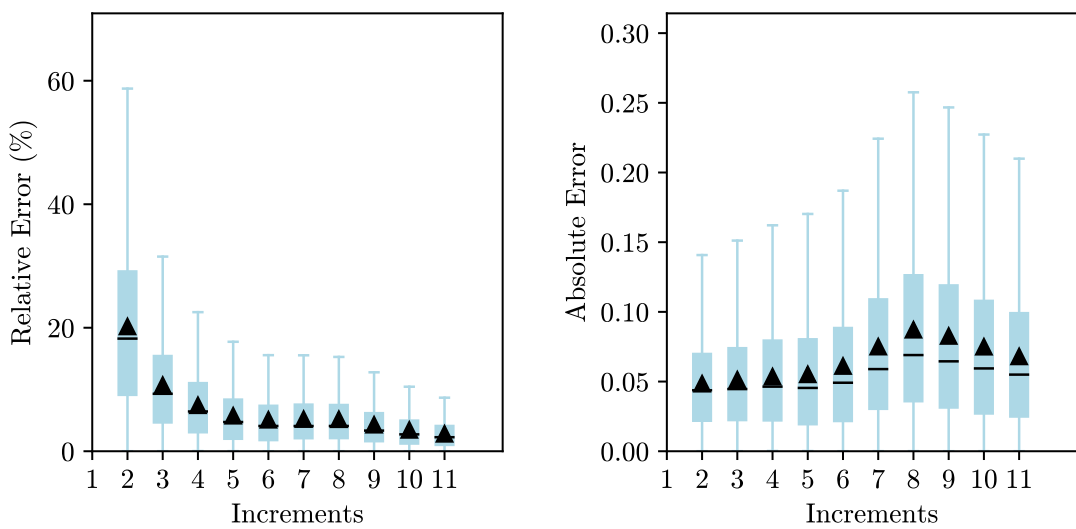


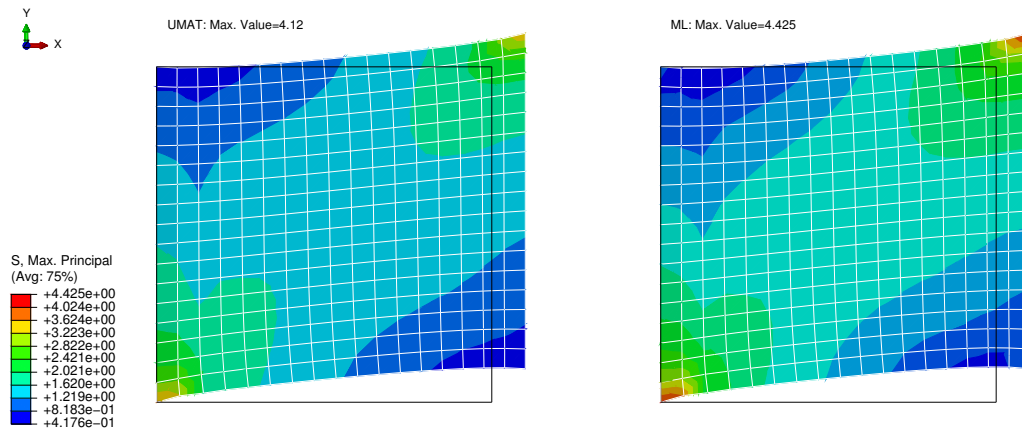
Figure 5.43: Displacement magnitude - Rectangular block under uniaxial load



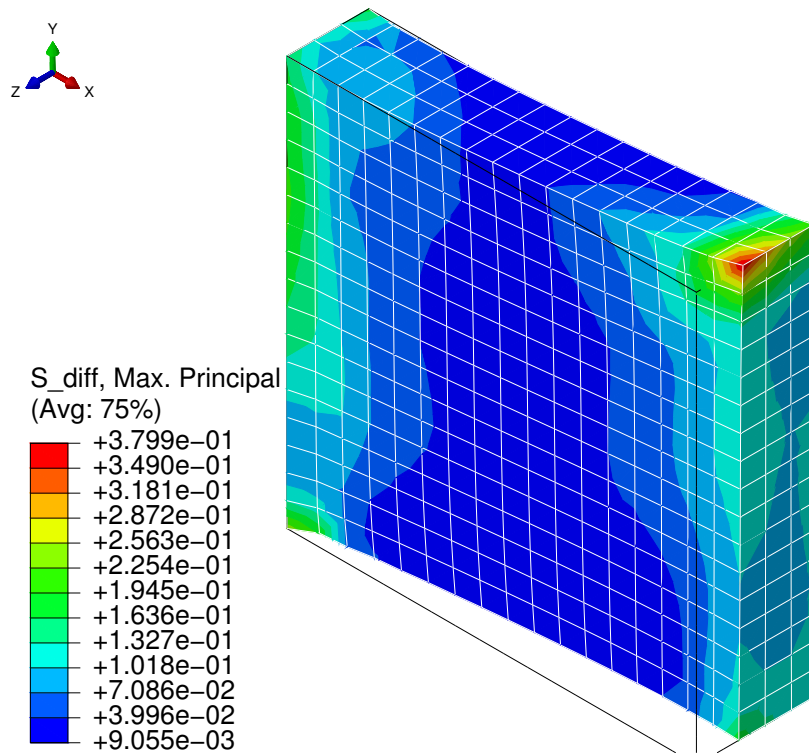
(a) Relative error

(b) Absolute error

Figure 5.44: Errors - Rectangular block under uniaxial load

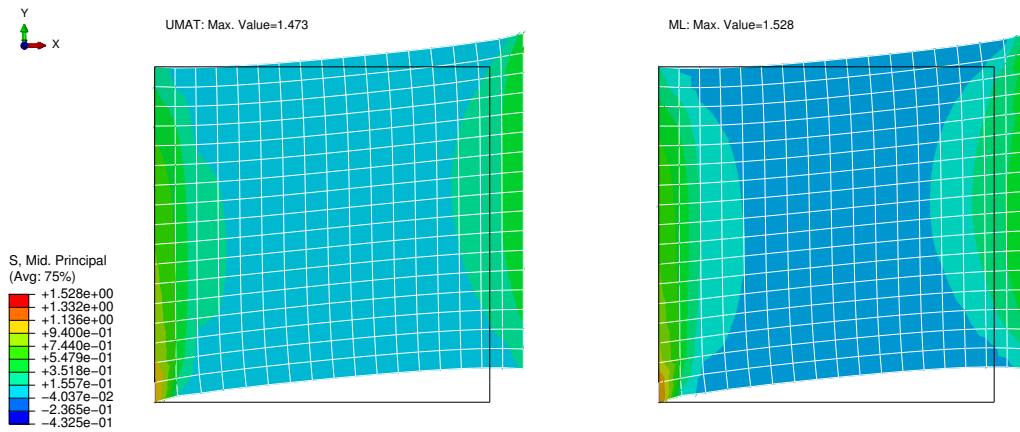


(a) UMAT (left) and ML (right)

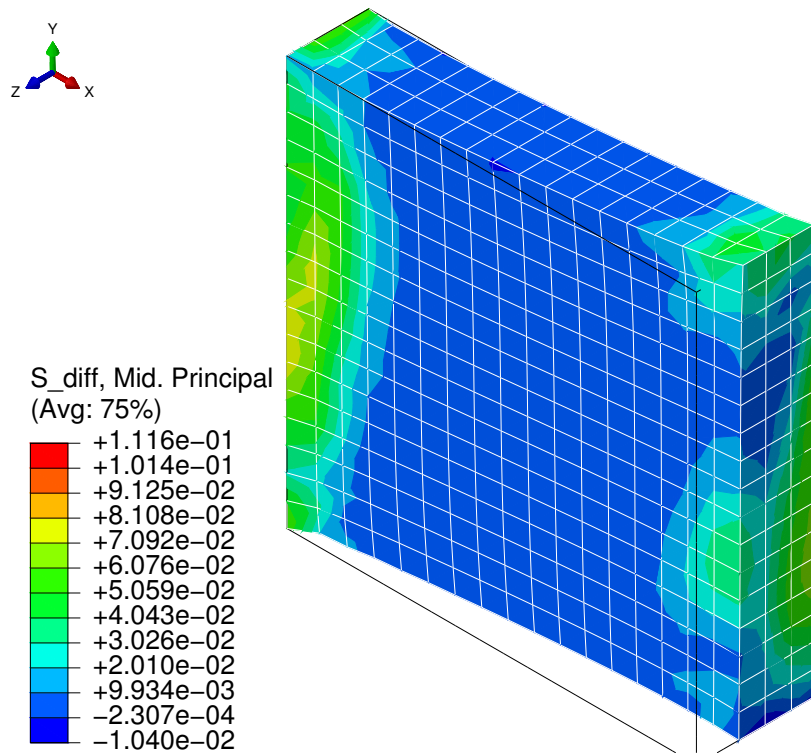


(b) Absolute difference

Figure 5.45: Maximum principal stress - Rectangular block under shear load

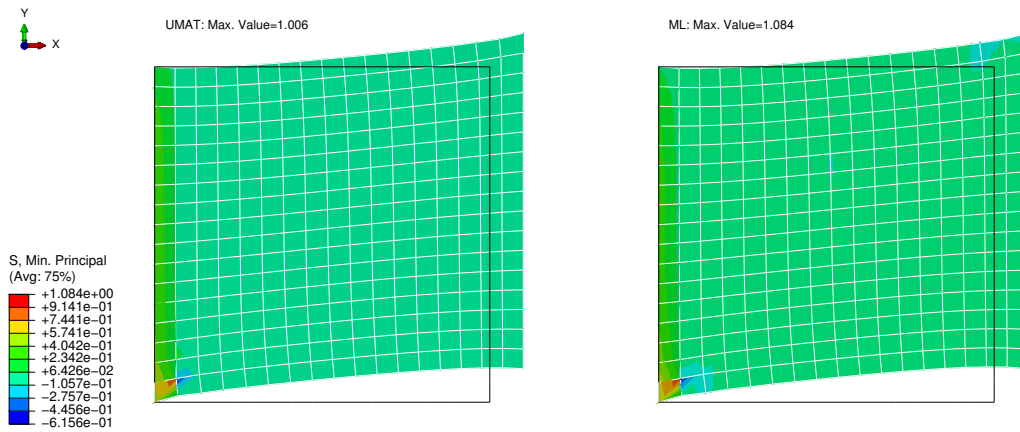


(a) UMAT (left) and ML (right)

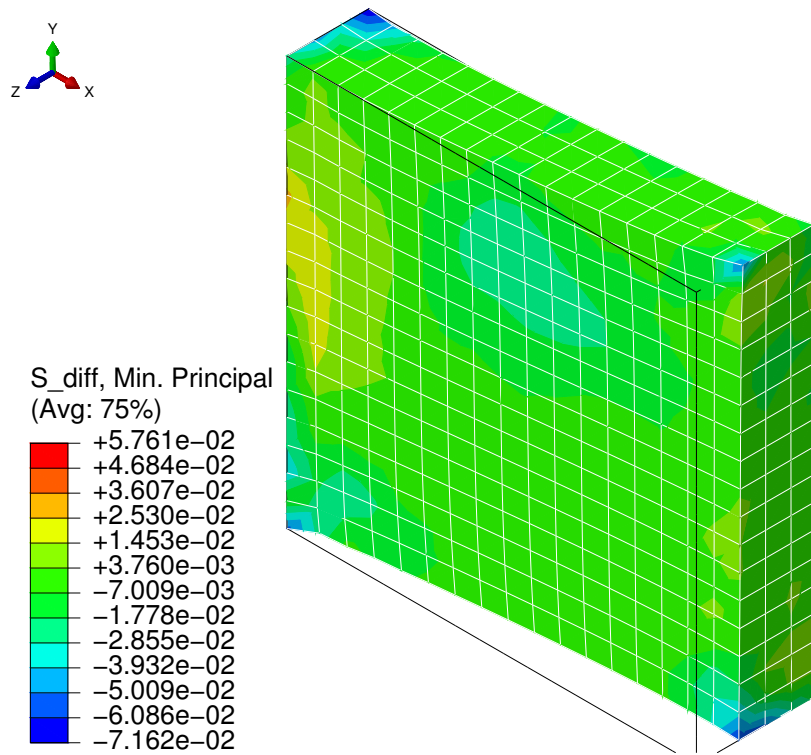


(b) Absolute difference

Figure 5.46: Middle principal stress - Rectangular block under shear load

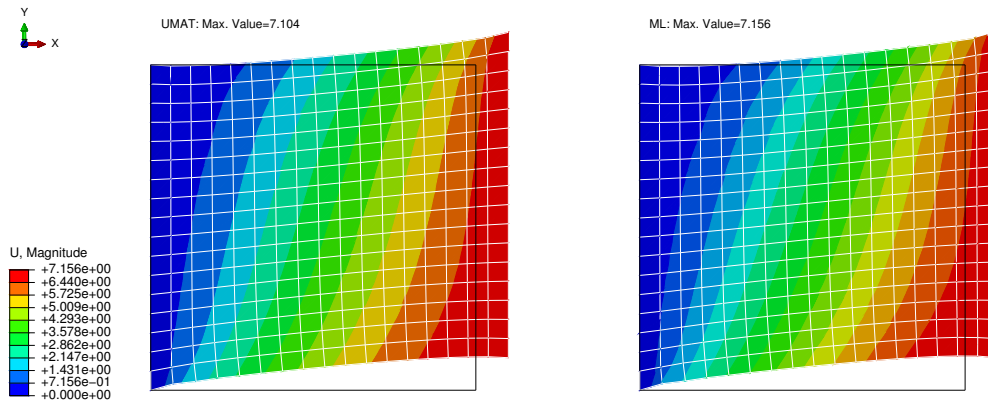


(a) UMAT (left) and ML (right)

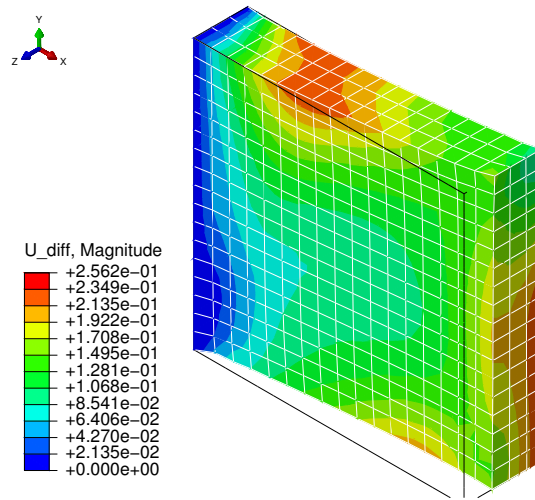


(b) Absolute difference

Figure 5.47: Minimum principal stress - Rectangular block under shear load

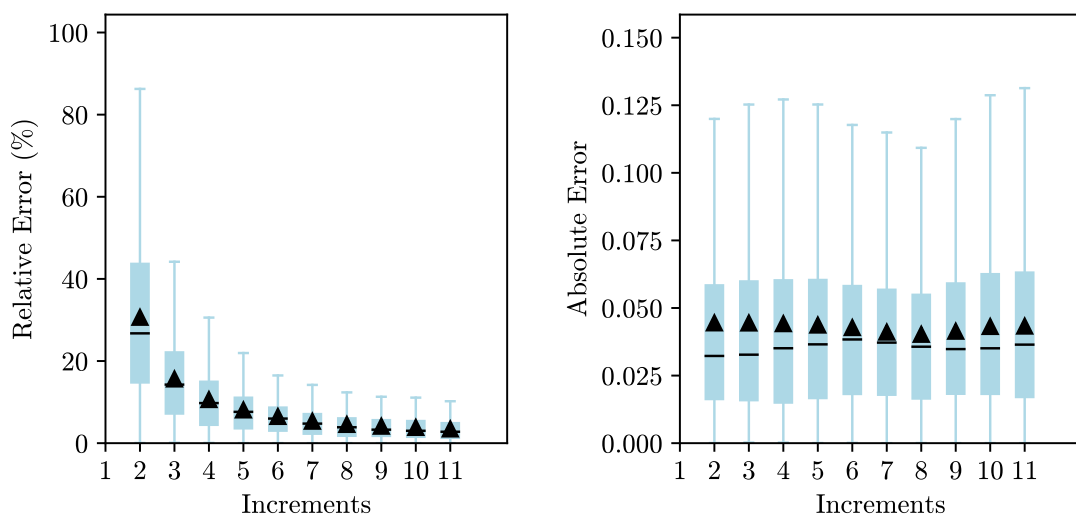


(a) UMAT (left) and ML (right)



(b) Absolute difference

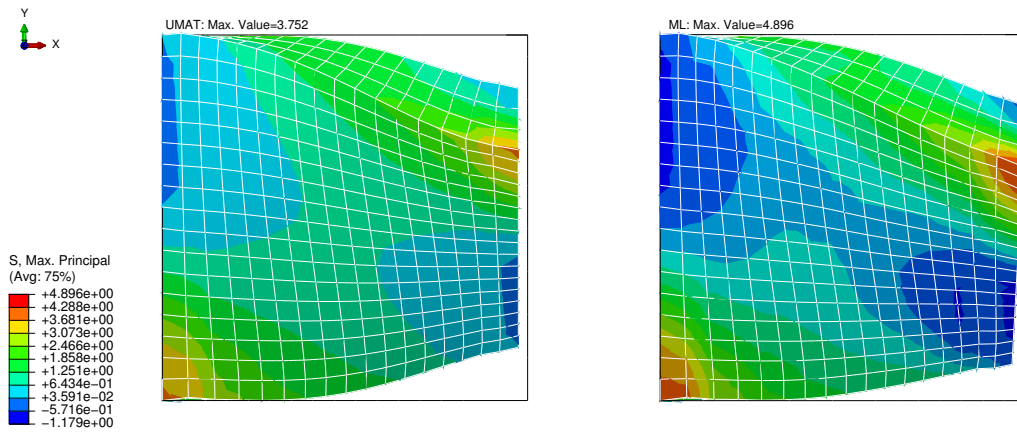
Figure 5.48: Displacement magnitude - Rectangular block under shear load



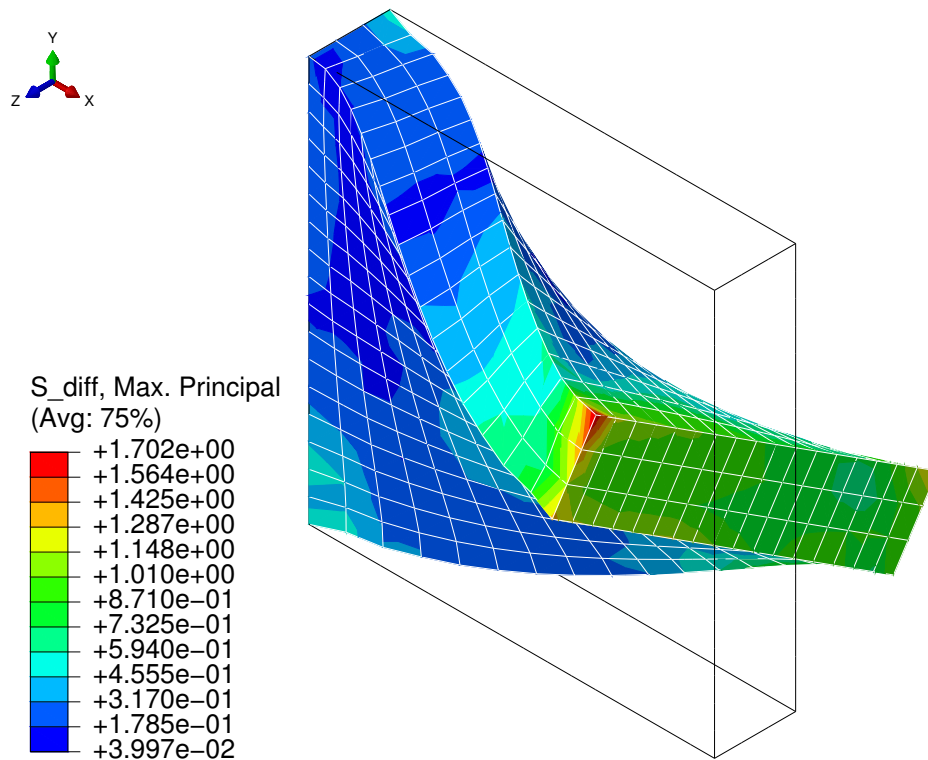
(a) Relative error

(b) Absolute error

Figure 5.49: Errors - Rectangular block under shear load

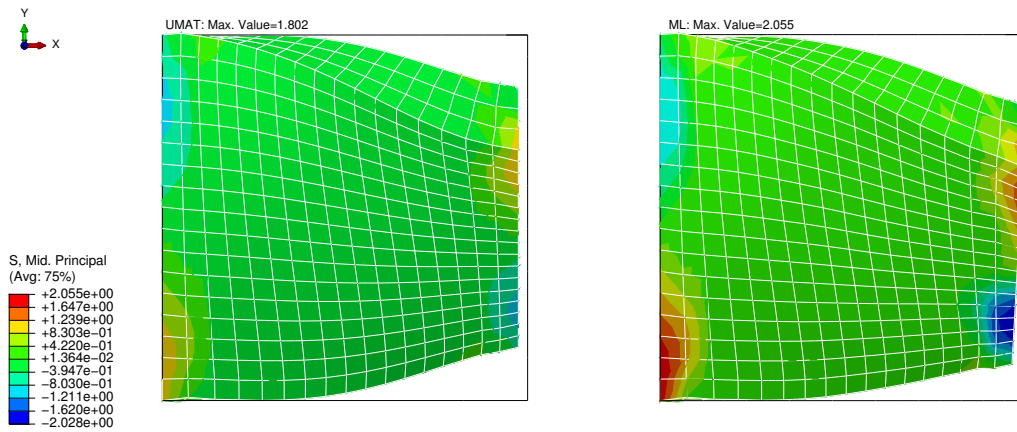


(a) UMAT (left) and ML (right)

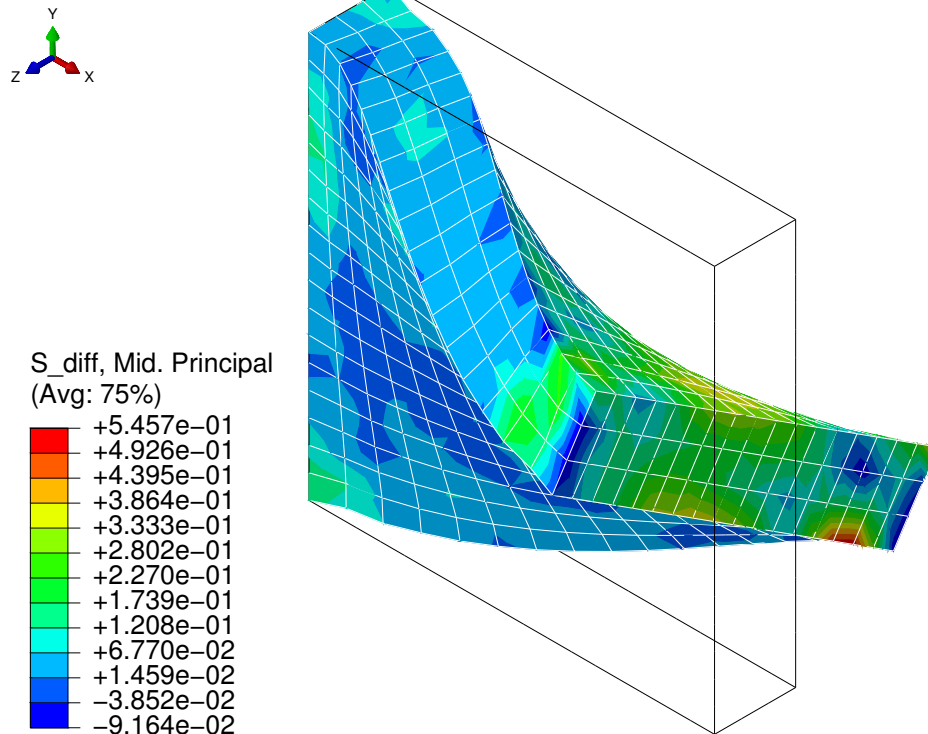


(b) Absolute difference

Figure 5.50: Maximum principal stress - Rectangular block under torsional load

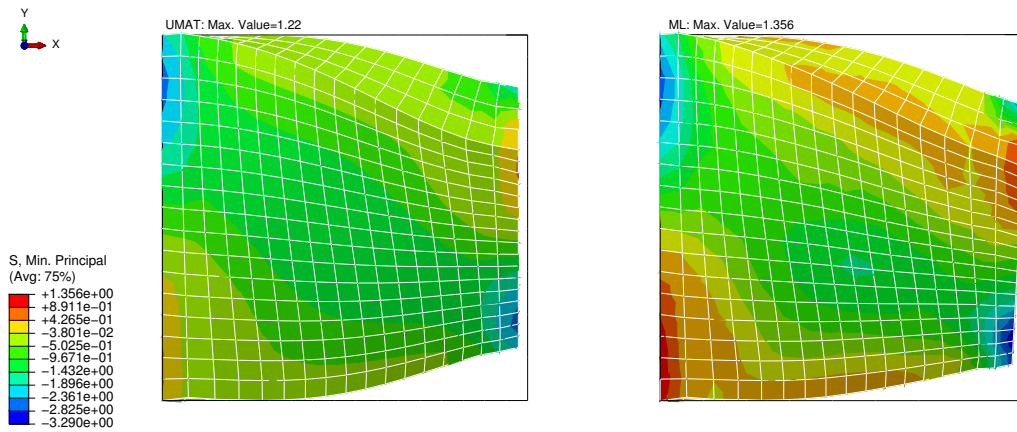


(a) UMAT (left) and ML (right)

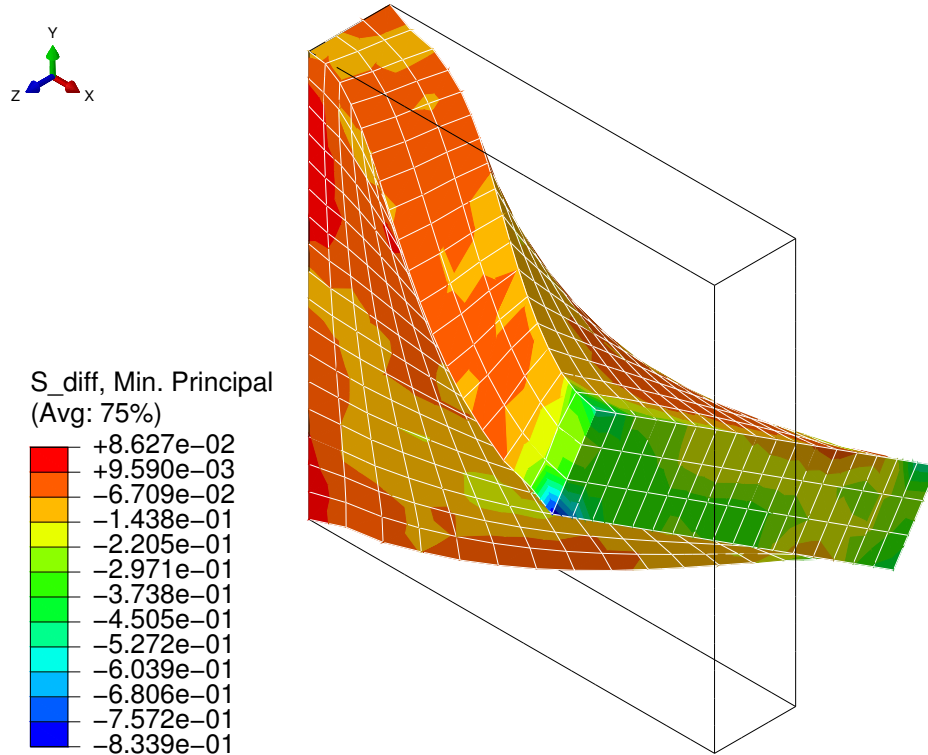


(b) Absolute difference

Figure 5.51: Middle principal stress - Rectangular block under torsional load

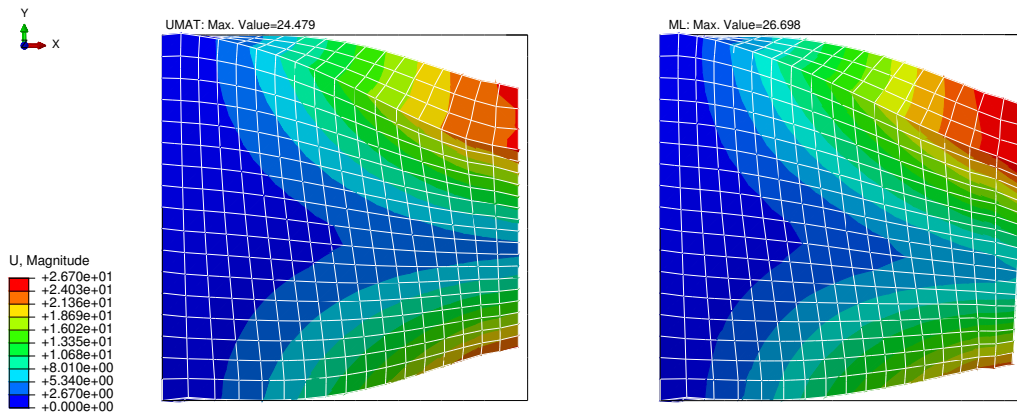


(a) UMAT (left) and ML (right)

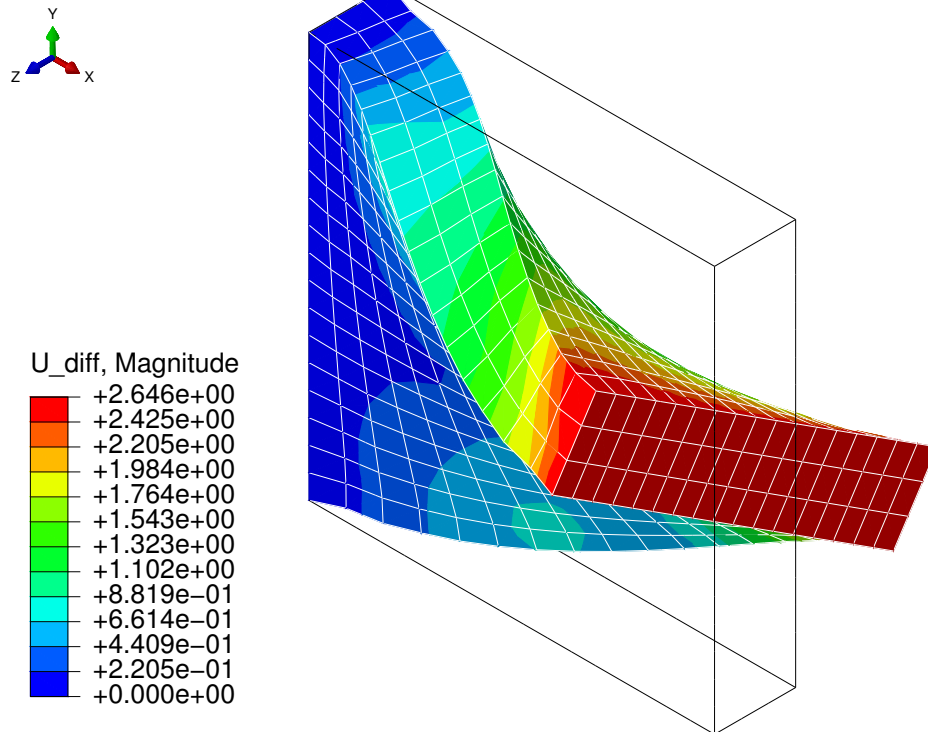


(b) Absolute difference

Figure 5.52: Minimum principal stress - Rectangular block under torsional load

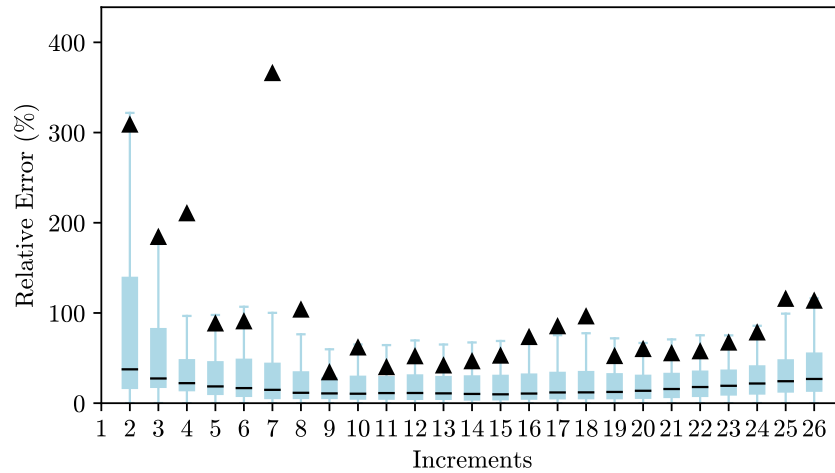


(a) UMAT (left) and ML (right)

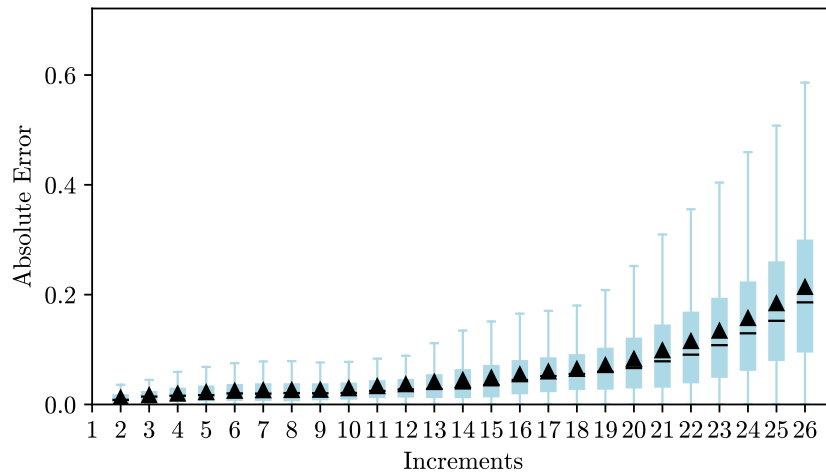


(b) Absolute difference

Figure 5.53: Displacement magnitude - Rectangular block under torsional load



(a) Relative error



(b) Absolute error

Figure 5.54: Errors - Rectangular block under torsional load

For these examples, it is possible to highlight that the main differences between the two approaches appear where the boundary conditions are applied. For example, in the maximum principal stress of the uniaxial loading scenario, it is visible that the values are identical in the middle of the rectangular block, but near the points that have a fixed displacement and near the points where the displacement is applied, the values differ a little more. However, the values are still close and the relative and absolute errors are small. The most complex example is the rectangular block under torsional load, where the results are not as good as in the previous problems. It required more time increments to complete the analysis, and the contour plots with the principal stresses show some notorious differences. These differences are not in accordance with the small absolute error shown in the box-and-whisker plot, which is explained by the fact that the absolute error was calculated for the stress components and not for the principal stresses. In addition, the absolute error tends to increase as the torsion of the rectangular block approaches

the rotation applied, which is a trend that was not observed in the previous examples. This example has significant deformations in the out-of-plane direction and is the most complex one. Despite this, the ML-based approach still gave acceptable results that show its potential in complex problems.

5.2.5 Convergency

It is also important to compare the number of iterations and the CPU time required to complete the analysis. Since the tangent stiffness matrix is also obtained with a ML model, it was expected that it would require more iterations and more CPU time when compared to an analysis that used the analytically calculated tangent stiffness matrix. In Table 5.1 it is shown the total number of iterations and the required CPU time for each problem with each approach. For almost all examples, the total number of iterations for the ML approach is double the total obtained with the UMAT. The difference is mainly in the first-time increments, where the ML approach requires some more iterations. Despite these differences, the ML approach still converges and the analysis is completed successfully, which means that the tangent stiffness matrix predicted is accurate enough.

To further study the differences, all the numerical problems were also analysed with a hybrid UMAT, where the stress state was calculated with a surrogate model and the tangent stiffness matrix was calculated analytically (column name Stress ML). In terms of the total number of iterations, the relative difference in relation to the conventional UMAT is nearly the same as the relative difference between the ML-based approach and the conventional UMAT. This may be indicative that the Jaumann rate has a big influence on the convergence because the Jaumann rate for the Stress ML approach was calculated with the ML-predicted stress, whereas the volumetric and isochoric components of the spatial elasticity tensor were calculated analytically.

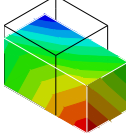
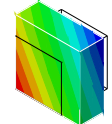
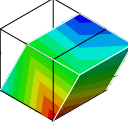
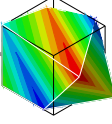
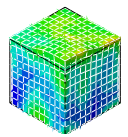
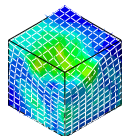
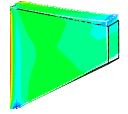
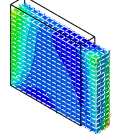
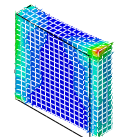
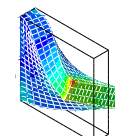
			Stress ML		ML		
			UMAT	Value	Diff.(%)	Value	Diff.(%)
	Uniaxial		17	19	+11.76	21	+23.53
			0.12 s	0.14 s	+16.67	0.13 s	+8.33
One Element Cube	Biaxial		21	20	-4.76	19	-9.52
			0.14 s	0.12 s	-14.29	0.14 s	+0.0
	Shear		11	14	+27.27	14	+27.27
			0.10 s	0.11 s	+10.00	0.12 s	+20.00
Random		12	18	+50.00	18	+50.00	
		0.11 s	0.12 s	+9.09	0.13 s	+18.18	
Block	Traction		21	28	+33.33	28	+33.33
			7.81 s	11.91 s	+52.50	19.88 s	+154.55
	Compression		21	33	+57.14	33	+57.14
			8.01 s	13.33 s	+66.42	22.56 s	+181.65
Cook's Membrane		11	23	+109.09	24	+118.18	
		22.98 s	45.38 s	+97.48	75.64 s	+229.16	
	Uniaxial		12	23	+91.67	24	+100.00
			4.75 s	8.81 s	+85.47	15.67 s	+229.90
Rect. Block	Shear		12	24	+100.00	24	+100.00
			4.82 s	9.17 s	+90.25	15.66 s	+224.90
	Torsion		100	183	+83.00	188	+88.00
			26.90 s	82.28 s	+205.87	95.90 s	+256.51

Table 5.1: Comparison - Total number of iterations and required CPU time

5.3 HGO Numerical Examples

The trained model to predict the Cauchy stress tensor for the HGO model was integrated in Abaqus and the FE method was applied. For the HGO model, it was not trained a model to predict the isochoric part of the elasticity tensor and the analytical tangent stiffness matrix was used to run the analysis. The ML-based approach for the HGO model was only tested in uniaxial loading cases because the NNs were only trained with deformation gradients representative of such loading scenarios. The example considered was a cube with one hexahedral element with 8 linear nodes, stretched in the x direction with different fibre orientations. The results are shown in Figures 5.55 to 5.58 for the fibre in the x-direction, in Figures 5.60 to 5.63 for the fibre in the y-direction and in Figures 5.65 to 5.68 for the fibre in the z-direction. Additionally, for the fibres in the x-direction, a compressive load was also considered and the respective results are shown in Figures 5.70 to 5.73.

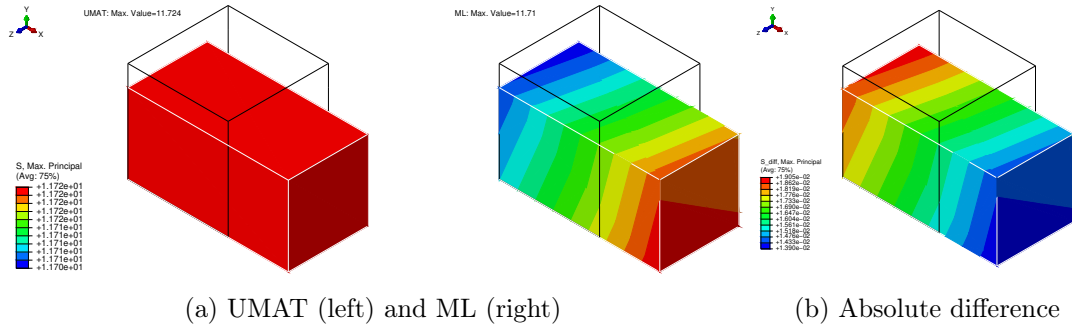


Figure 5.55: Maximum principal stress - Cube with uniaxial load and fibres in x-direction

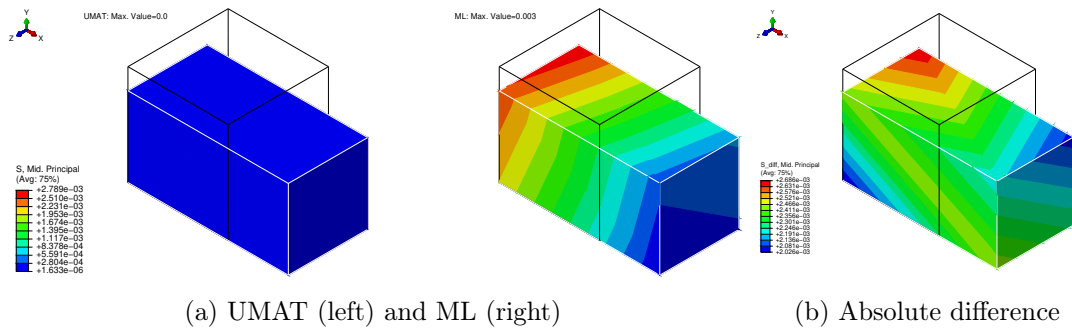


Figure 5.56: Middle principal stress - Cube with uniaxial load and fibres in x-direction

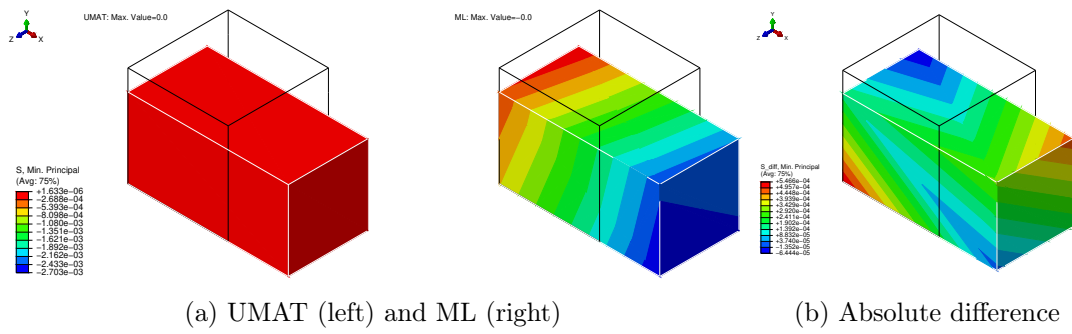


Figure 5.57: Minimum principal stress - Cube with uniaxial load and fibres in x-direction

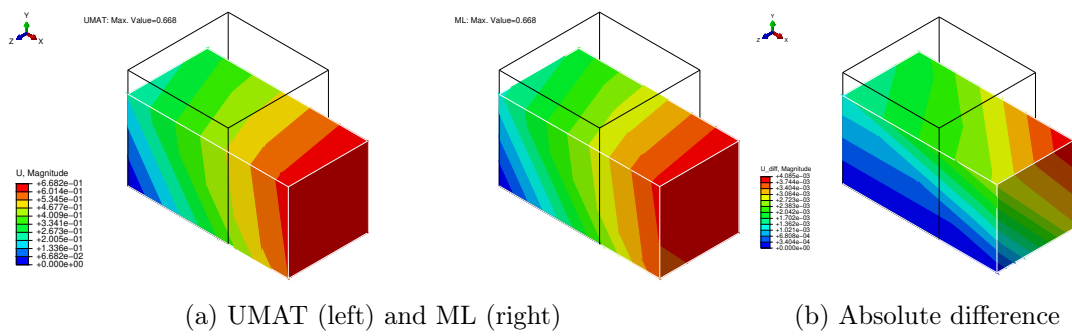


Figure 5.58: Displacement magnitude - Cube with uniaxial load and fibres in x-direction

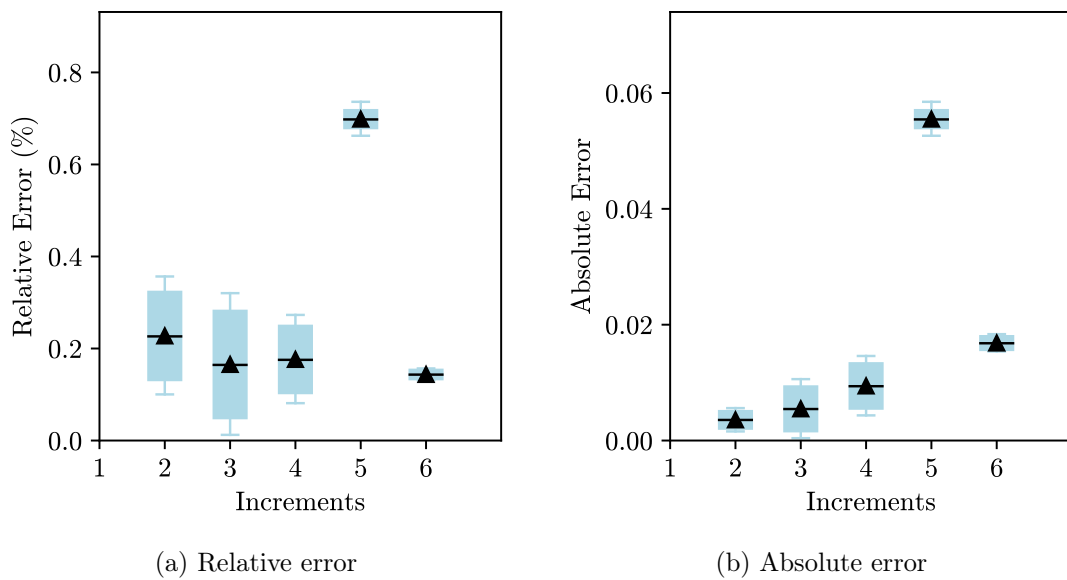


Figure 5.59: Errors - Cube with uniaxial load and fibres in x-direction

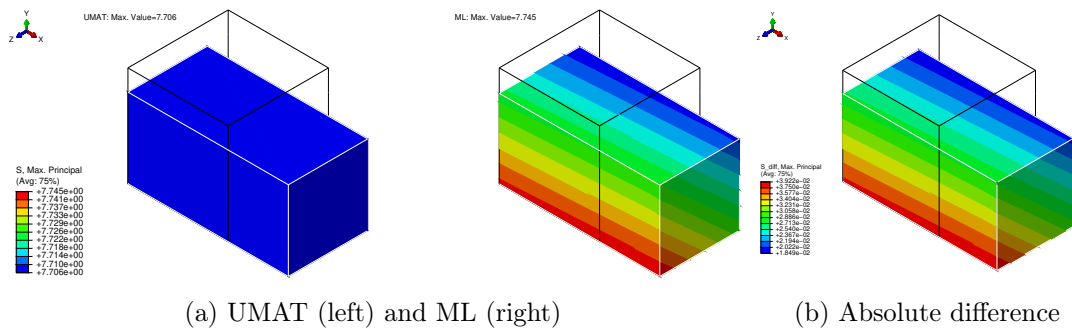


Figure 5.60: Maximum principal stress - Cube with uniaxial load and fibres in y-direction

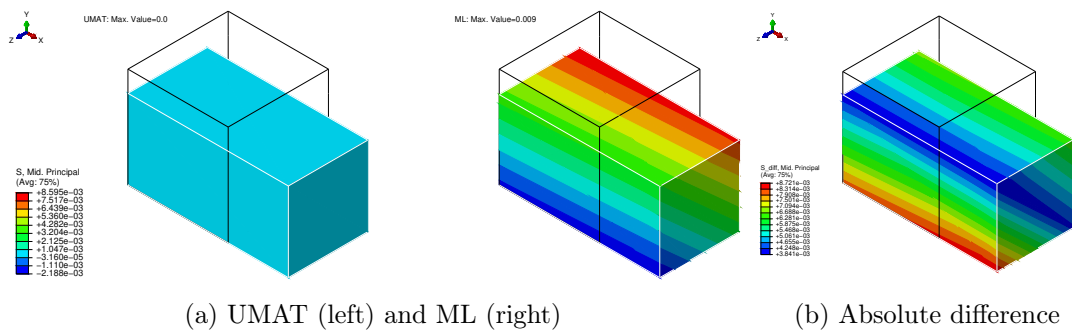


Figure 5.61: Middle principal stress - Cube with uniaxial load and fibres in y-direction

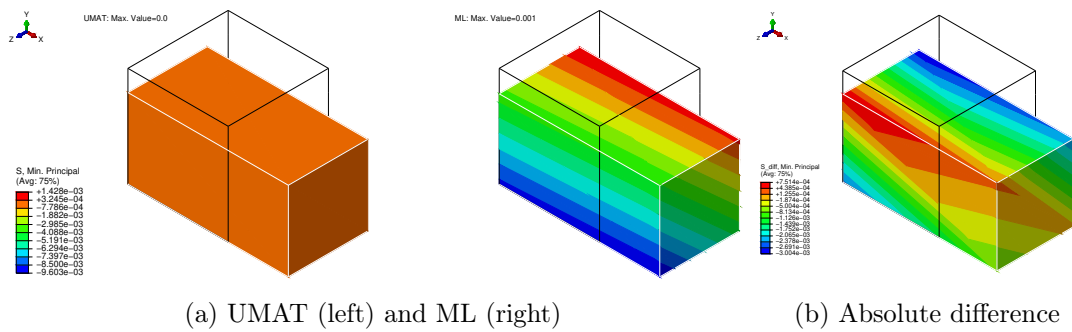


Figure 5.62: Minimum principal stress - Cube with uniaxial load and fibres in y-direction

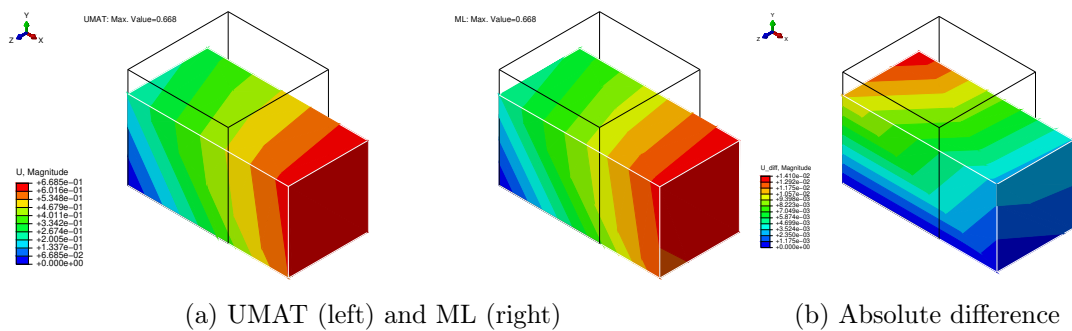


Figure 5.63: Displacement magnitude - Cube with uniaxial load and fibres in y-direction

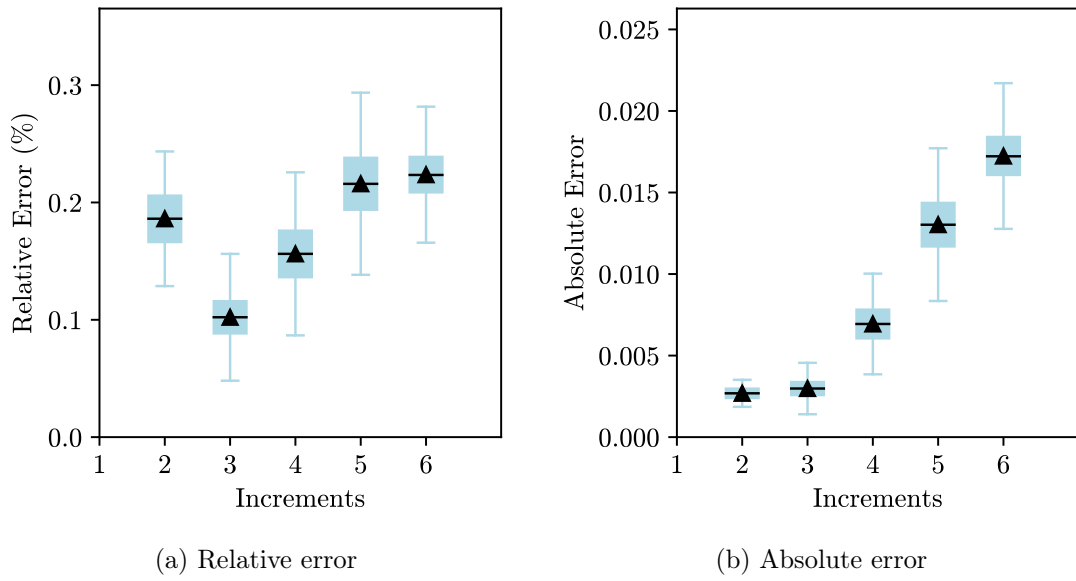


Figure 5.64: Errors - Cube with uniaxial load and fibres in y-direction

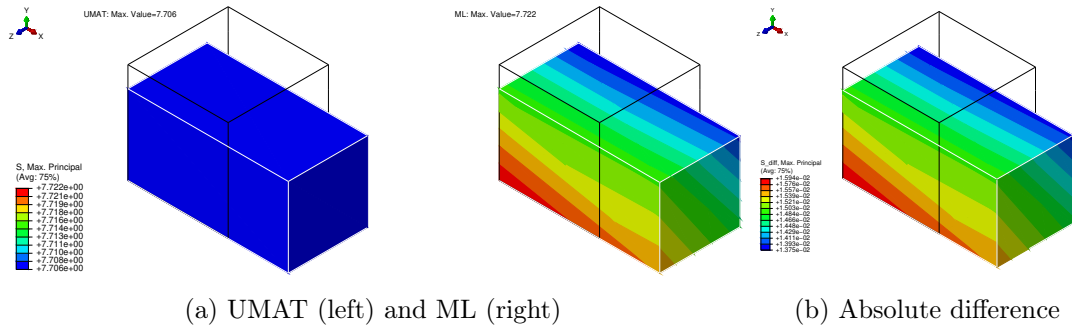


Figure 5.65: Maximum principal stress - Cube with uniaxial load and fibres in z-direction

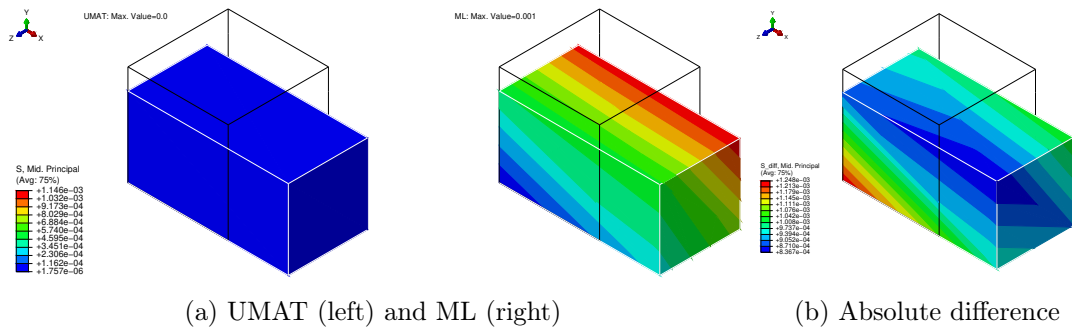


Figure 5.66: Middle principal stress - Cube with uniaxial load and fibres in z-direction

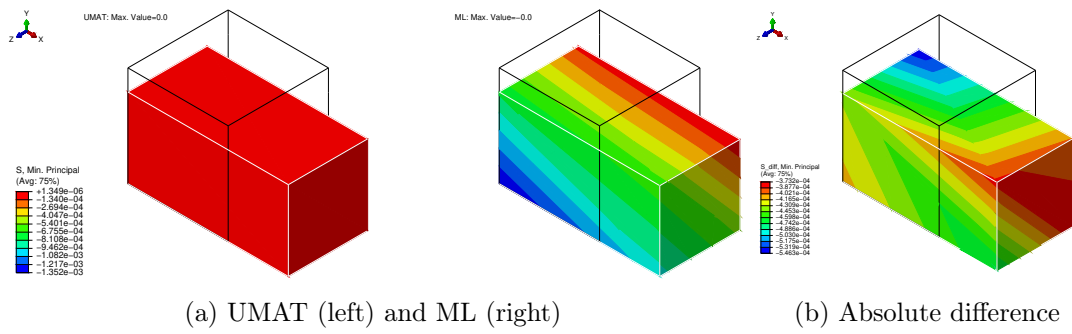


Figure 5.67: Minimum principal stress - Cube with uniaxial load and fibres in z-direction

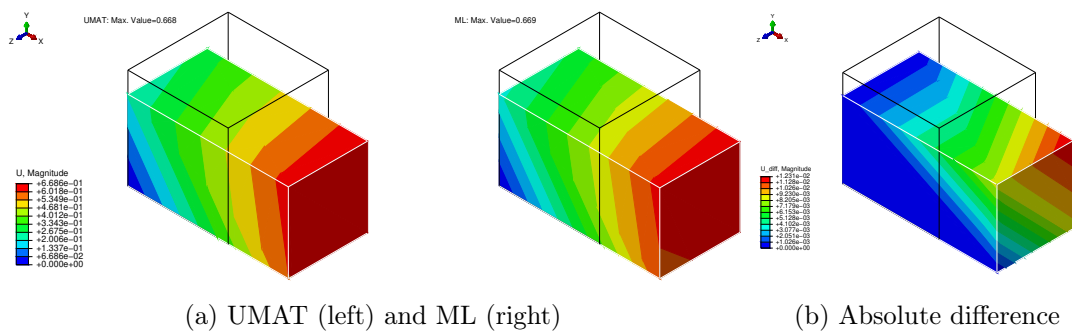


Figure 5.68: Displacement magnitude - Cube with uniaxial load and fibres in z-direction

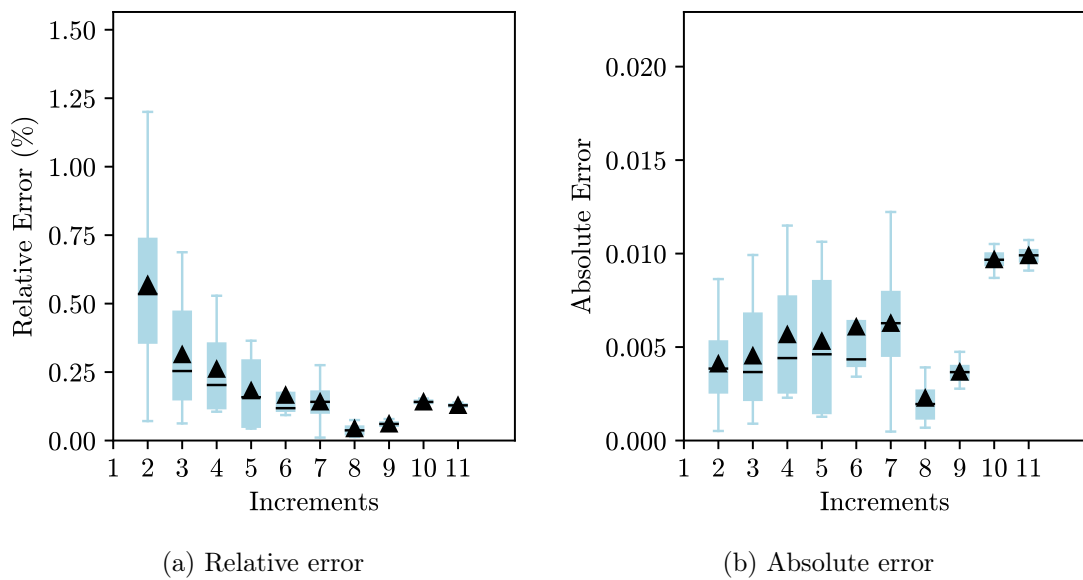


Figure 5.69: Errors - Cube with uniaxial load and fibres in z-direction

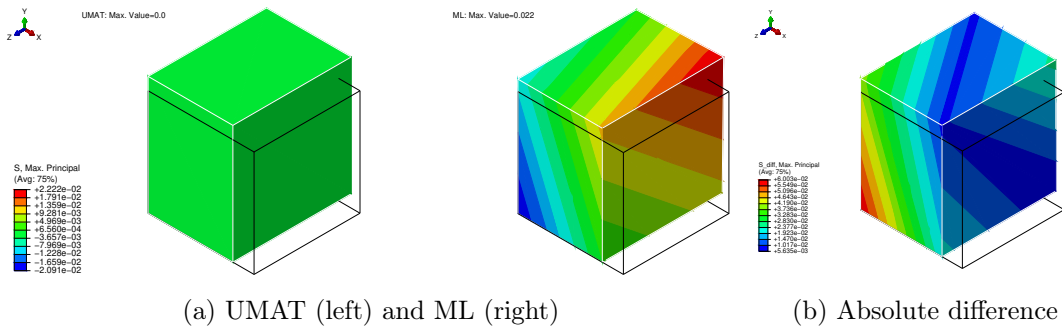


Figure 5.70: Maximum principal stress - Cube with compressive load and fibres in x-direction

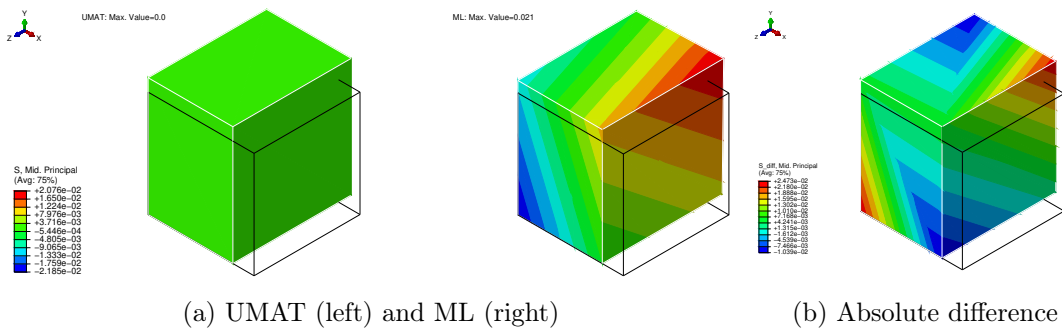


Figure 5.71: Middle principal stress - Cube with compressive load and fibres in x-direction

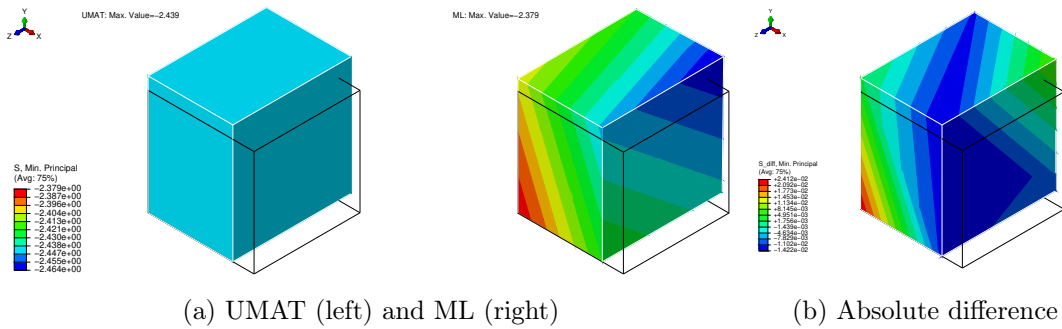


Figure 5.72: Minimum principal stress - Cube with compressive load and fibres in x-direction

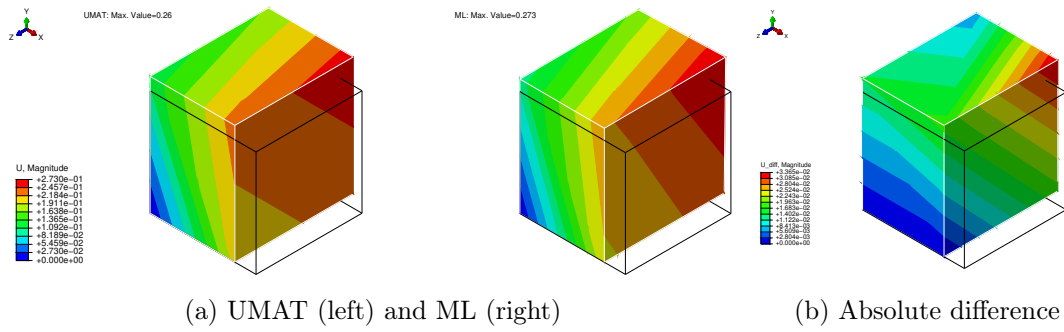


Figure 5.73: Displacement magnitude - Cube with compressive load and fibres in x-direction

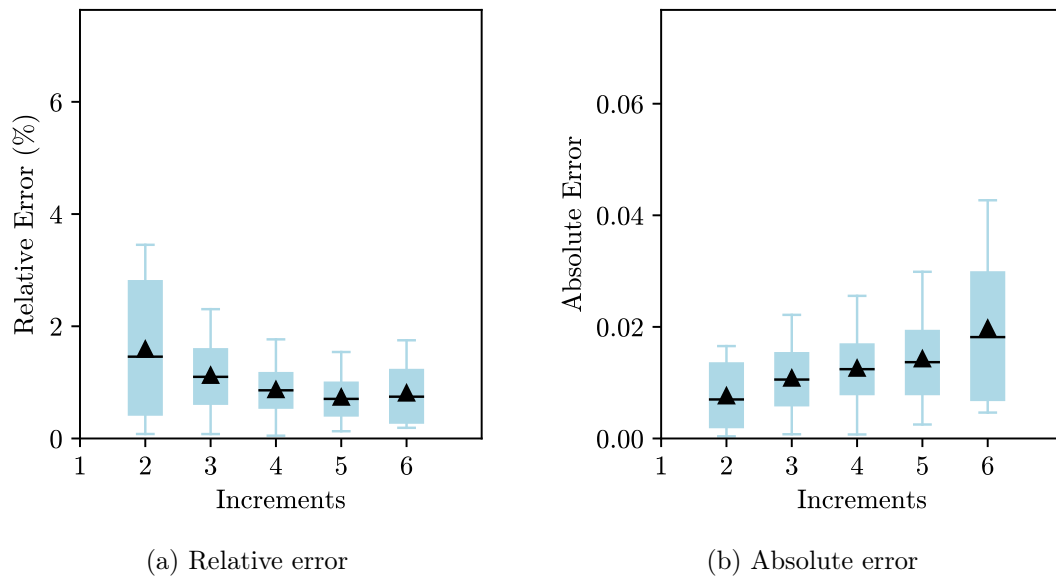


Figure 5.74: Errors - Cube with compressive load and fibres in x-direction

The results obtained for the principal stresses with the two approaches are similar for every fibre direction, and the relative and absolute errors are small. The displacement magnitude is nearly the same, which was expected since a displacement was applied. Here, as it was with the Neo-Hookean model, the contour plot for the ML-based approach does not give a uniform colour contour and the reasons for this are the same. The stresses in all the integration points are not equal and have in fact some residual differences that give a colour contour that seems much different from the one obtained with a conventional UMAT. However, if a closer look is taken at the legend, it is possible to see that the differences between values for each colour are minimal. Regarding the example with compression, the relative and absolute errors are a little bit higher but still really close to the ones obtained with a conventional UMAT. The example with the fibres in the y and z-directions and the one of the compressive load are particularly relevant because the fibres are in compression and the proposed HGO model refers that in such cases, the contribution of the fibres is

ignored and the material behaves as an isotropic one, in this case coinciding with a Neo-Hookean material. This is a complex and important aspect that the NN was able to learn since the results are close.

It is important to mention that using the analytical tangent stiffness matrix helps to reach such good results. If a surrogate was trained to predict it for the HGO model, the results may have been worse because the tangent stiffness matrix also has an influence on the calculated stresses. However, these examples still show the capability to express the stress-strain relationship with a data-driven approach.

5.4 Concluding Remarks

In a general way, the ML-based approach yielded similar results to the ones obtained with a conventional UMAT. The relative errors were high in some numerical examples, especially for the first increments, but as it was mentioned, a plausible reason for such high values is that at the beginning of the analysis, the stress values are small, which means that even for a small absolute error, the relative error is high. This hypothesis is supported by the box-and-whisker plot of the absolute error that shows that in fact, in most of the numerical examples, the absolute error is small. Additionally, the relative error tends to decrease as the number of increments increases, further supporting this hypothesis.

Additionally, the contour plots are really identical for most of the variables considered and for most of the examples considered. This is supported by the contour plots that display the absolute differences between the results, where the values are small.

Regarding the convergence, the ML-based approach took more iterations and more CPU time to complete the analysts, but they were still completed. In addition, the tangent stiffness matrix also has an influence on the stresses and displacements and since the results are similar in general, it is possible to conclude that the tangent stiffness matrix is accurately predicted. It is also important to highlight again the fact that the number of iterations obtained with a hybrid approach (Stress ML) shows that the Jaumann rate has the biggest influence on the convergence of the analysed problems.

Chapter 6

Conclusion

6.1 Final Remarks

In this work, a data-driven approach for the constitutive modelling of hyperelastic materials is proposed as an alternative to the conventional way of modelling the mechanical behaviour of such materials. Surrogate models were trained to describe the behaviour of hyperelastic materials and they were then integrated in Abaqus in order to employ the FE method.

Firstly, it was considered a Neo-Hookean material and a NN was trained to predict the stress for some homogeneous deformations and for a general deformation. The test losses obtained for the trained models were all small and indicative that it is possible to consider a data-driven approach to model the constitutive behaviour of hyperelastic materials. However, the main goal of this dissertation was to test and integrate the trained surrogates with the FE method, in this case with Abaqus. Thus, another NN was trained to predict the spatial elasticity tensor that would be used to compute the tangent stiffness matrix used in Abaqus.

The NN to predict the isochoric part of the stress and of the spatial elasticity tensor for a random deformation gradient were used to write a UMAT to describe the constitutive behaviour of the material in analysis. To do so, the weights and biases of the trained models were used to write the forward pass equations in Fortran. After this step, the results from some numerical examples with the ML-based UMAT and a conventional one were compared. The ML-based approach gave similar results, even for some complex problems. The relative errors between the components of the stress tensor were high for some cases, but the absolute errors showed that the differences were small. Additionally, the contour plots of the principal stresses and the magnitude of the displacement showed that the ML-based approach yielded accurate results. Regarding the convergence, the ML-based approach took more iterations and more time for the most part of the examples, which was expected because the tangent stiffness matrix was approximated and this has a direct influence on the convergence. However, the analysis with the ML-based approach was still completed.

Then, a more complex model that accounts for a preferred direction was considered. It was restricted to uniaxial loading scenarios, and the tangent stiffness matrix was calculated analytically. However, this was still a hard model for the NN to learn because, with a different fibre direction, the material behaves completely differently. Hence, for the HGO model, the trained surrogate model trained is able to predict the stress for a material with fixed material parameters and for uniaxial loading cases but with different material orientations. Additionally, the proposed HGO model states that if the fibres are in compression, they do not contribute to the SEF and the material is regarded as a Neo-Hookean material in this case. This is another complexity of the HGO model that the NN was able to learn, as it was proved with the uniaxial compressive load and for the fibres in the y and z-direction.

Data-driven constitutive modelling is gaining more interest in recent years, and there are already some different approaches proposed in the literature. However, most of them use the invariants of the Cauchy-Green deformation tensors as inputs of the NN and the strain energy and its derivatives as outputs. This solution provides more flexibility but still requires some mathematical treatment to compute the stress and the tangent stiffness matrix required by the UMAT. In this work, the UMAT has a model to predict the stresses and a model to predict the spatial elasticity tensor, and there is no need to express the mathematical equations that describe the different models used. Only the volumetric part is calculated with mathematical equations because they depend on the boundary conditions of the problem in analysis. Therefore, in this work, there is no need to derive equations to describe the behaviour of the material in analysis and it is only necessary to train a surrogate model and pass the parameters of the trained model to Fortran.

In general, the different numerical examples analysed showed that the ML-based approach was able to model the constitutive behaviour of hyperelastic materials since the results obtained were similar to the ones obtained with a conventional UMAT. The analyses took longer to converge, which was expected. However, all the analyses were completed, requiring only more iterations to do so when compared with a standard UMAT.

6.2 Future Work

For future work, it would be important to train a surrogate to predict the spatial elasticity tensor for the HGO model to get a UMAT based only on ML that does not need the development of mathematical equations. This was accomplished for the Neo-Hookean model but for the HGO model, only the stress was predicted with a ML surrogate. Additionally, it would also be interesting to generalize the ML surrogate to predict the Cauchy stress tensor for the HGO model for a general deformation, and not only for a uniaxial loading scenario. Validating the ML-based approach for the HGO model for some more complex problems is also relevant to future work.

Additionally, incorporating the material parameters of the SEF as inputs of the NNs would also be a valuable consideration, since the developed and trained models assume values for the material parameters. However, generalizing the trained surrogates for any material parameter would make the training process much harder. It would be necessary much more data and to use regularization and other techniques to obtain better results. This gives rise to another difficulty, which is to pass a model with dropout, for example, from Python to Fortran.

Another interesting future work would be to explore the effect of the splits of the dataset. During the training process of the NNs, the splits of the dataset used were fixed, but they can have an influence on the results obtained and tuning these hyperparameters, namely the size of the training, validation and test data, can improve the accuracy of the predictions. In addition, in the numerical examples analysed, the mesh used was fixed and this can also have an impact on the results obtained. Thus, doing a mesh convergence study could also be relevant in the future.

Finally, considering different inputs and outputs for the NNs could also be relevant to future work. For example, using the right Cauchy-Green deformation tensor as input of the deformation gradient would be interesting because this tensor is symmetric, which means that it would be possible to decrease from 9 inputs to 6. Then, if the output is the second Piola-Kirchhoff stress tensor, the derivatives of the outputs in relation to the inputs, which are easily obtained for a trained model with backpropagation, could be used to compute the tangent stiffness matrix. This would be of great importance because it would not be necessary to train a model for the spatial elasticity tensor. Then, the Cauchy stress tensor required for Abaqus could be easily obtained from the second Piola-Kirchhoff stress tensor. With this approach and in the presence of enough experimental data for a given material without a proposed model in the literature that describes its constitutive behaviour properly, it would be possible to use this approach to simulate a problem involving such material instead of developing a SEF that describes the material and to derive the correspondent material parameters.

Bibliography

- [1] J. D. Bronzino, *Chapter 1 - Biomedical Engineering: A Historical Perspective*. Boston: Academic Press, 2012, pp. 1–33. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780123749796000010>
- [2] J. Humphrey and S. L. O'Rourke, *An Introduction to Biomechanics: Solids and Fluids, Analysis and Design*, 2nd ed. Springer New York, NY, 2015.
- [3] M. Moayedi, A. R. Arshi, M. Salehi, M. Akrami, and R. Naemi, "Associations between changes in loading pattern, deformity, and internal stresses at the foot with hammer toe during walking; a finite element approach," *Computers in Biology and Medicine*, vol. 135, p. 104598, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482521003929>
- [4] S. S. Sajjadinia, M. Haghpanahi, and M. Razi, "Computational simulation of the multiphasic degeneration of the bone-cartilage unit during osteoarthritis via indentation and unconfined compression tests," *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 233, no. 9, pp. 871–882, 2019. [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1177/0954411919854011>
- [5] L. Zhang, S.-K. Choi, T. Xie, P. Jiang, J. Hu, and J. Koo, "Multi-fidelity surrogate model-assisted fatigue analysis of welded joints," *Structural and Multidisciplinary Optimization*, vol. 63, no. 6, pp. 2771–2787, 2021. [Online]. Available: <https://doi.org/10.1007/s00158-020-02840-9>
- [6] A. Chakraborty, P. Datta, S. Majumder, S. C. Mondal, and A. Roychowdhury, "Finite element and experimental analysis to select patient's bone condition specific porous dental implant, fabricated using additive manufacturing," *Computers in Biology and Medicine*, vol. 124, p. 103839, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482520302006>
- [7] L.-R. Chang, Y.-P. Hou, and T.-S. Lin, "Is perpendicular double two-hole plates fixation superior to single four-hole plate fixation to treat mandibular symphysis fracture?—a finite element study," *Applied Sciences*, vol. 11, no. 18, p. 8629, 2021. [Online]. Available: <https://www.mdpi.com/2076-3417/11/18/8629>

- [8] G. Chen, W. Fan, S. Mishra, A. El-Atem, M. A. Schuetz, and Y. Xiao, “Tooth fracture risk analysis based on a new finite element dental structure models using micro-ct data,” *Computers in Biology and Medicine*, vol. 42, no. 10, pp. 957–963, 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482512001163>
- [9] I. S. Mohd Moideen, C. T. Lim, R. C. H. Yeow, and D. Y. R. Chong, “Polka dot cementless talar component in enhancing total ankle replacement fixation: A parametric study using the finite element analysis approach,” *Computers in Biology and Medicine*, vol. 141, p. 105142, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482521009367>
- [10] Y. Peng, D. W.-C. Wong, T. L.-W. Chen, Y. Wang, G. Zhang, F. Yan, and M. Zhang, “Influence of arch support heights on the internal foot mechanics of flatfoot during walking: A muscle-driven finite element analysis,” *Computers in Biology and Medicine*, vol. 132, p. 104355, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482521001499>
- [11] S. S. Sajjadinia and M. Haghpanahi, “A parametric study on the mechanical role of fibrillar rotations in an articular cartilage finite element model,” *Scientia Iranica*, vol. 28, no. 2, pp. 830–836, 2021, 1026-3098. [Online]. Available: http://scientiairanica.sharif.edu/article_21815_50ac8f385a0c2db62e2d6fd440288051.pdf
- [12] S. S. Sajjadinia, B. Carpentieri, D. Shriram, and G. A. Holzapfel, “Multi-fidelity surrogate modeling through hybrid machine learning for biomechanical and finite element analysis of soft tissues,” *Computers in Biology and Medicine*, vol. 148, p. 105699, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482522004796>
- [13] M. Freutel, H. Schmidt, L. Dürselen, A. Ignatius, and F. Galbusera, “Finite element modeling of soft tissues: material models, tissue interaction and challenges,” *Clin Biomech (Bristol, Avon)*, vol. 29, no. 4, pp. 363–72, 2014.
- [14] M. Liu, L. Liang, and W. Sun, “A generic physics-informed neural network-based constitutive model for soft biological tissues,” *Computer Methods in Applied Mechanics and Engineering*, vol. 372, p. 113402, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782520305879>
- [15] L. R. G. Treloar, “The Elasticity of a Network of Long-Chain Molecules. II,” *Rubber Chemistry and Technology*, vol. 17, no. 2, pp. 296–302, 06 1944. [Online]. Available: <https://doi.org/10.5254/1.3546653>
- [16] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, “Deep speech: Scaling

- up end-to-end speech recognition,” p. arXiv:1412.5567, December 01, 2014 2014. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/2014arXiv1412.5567H>
- [17] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, Conference Proceedings, pp. 1026–1034.
- [18] K. He, X. Zhang, and S. Ren, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, Conference Proceedings, pp. 770–778.
- [19] X. He, S. Avril, and J. Lu, “Machine learning prediction of tissue strength and local rupture risk in ascending thoracic aortic aneurysms,” *Molecular & Cellular Biomechanics*, vol. 16, no. Suppl.2, pp. 50–52, 2019. [Online]. Available: <http://www.techscience.com/mcb/v16nSuppl.2/35182>
- [20] I. Kokkinos, “Pushing the boundaries of boundary detection using deep learning,” p. arXiv:1511.07386, November 01, 2015 2015. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/2015arXiv151107386K>
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, p. 84–90, 2017. [Online]. Available: <https://doi.org/10.1145/3065386>
- [22] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: <https://doi.org/10.1038/nature14539>
- [23] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, Conference Proceedings, pp. 1701–1708.
- [24] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” p. arXiv:1609.08144, September 01, 2016 2016. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/2016arXiv160908144W>
- [25] Y. Dabiri, A. Van der Velden, K. L. Sack, J. S. Choy, G. S. Kassab, and J. M. Guccione, “Prediction of left ventricular mechanics using machine learning,” *Frontiers in Physics*, vol. 7, 2019. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fphy.2019.00117>

- [26] M. Cilla, I. Pérez-Rey, M. A. Martínez, E. Peña, and J. Martínez, “On the use of machine learning techniques for the mechanical characterization of soft biological tissues,” *International Journal for Numerical Methods in Biomedical Engineering*, vol. 34, no. 10, p. e3121, 2018. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.3121>
- [27] H. N. Do, A. Ijaz, H. Gharahi, B. Zambrano, J. Choi, W. Lee, and S. Baek, “Prediction of abdominal aortic aneurysm growth using dynamical gaussian process implicit surface,” *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 3, pp. 609–622, 2019.
- [28] Z. Jiang, H. Do, J. Choi, W. Lee, and S. Baek, “A deep learning approach to predict abdominal aortic aneurysm expansion using longitudinal data,” *Frontiers in Physics*, vol. 7, p. 235, 2020.
- [29] Y. Luo, Z. Fan, S. Baek, and J. Lu, “Machine learning–aided exploration of relationship between strength and elastic properties in ascending thoracic aneurysm,” *International Journal for Numerical Methods in Biomedical Engineering*, vol. 34, no. 6, p. e2977, 2018. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.2977>
- [30] G. R. Joldes, K. Miller, A. Wittek, R. O. Forsythe, D. E. Newby, and B. J. Doyle, “Bioparr: A software system for estimating the rupture potential index for abdominal aortic aneurysms,” *Scientific Reports*, vol. 7, no. 1, p. 4641, 2017. [Online]. Available: <https://doi.org/10.1038/s41598-017-04699-1>
- [31] Z. Liu, C. T. Wu, and M. Koishi, “A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials,” *Computer Methods in Applied Mechanics and Engineering*, vol. 345, pp. 1138–1168, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782518304729>
- [32] L. Liang, M. Liu, and W. Sun, “A deep learning approach to estimate chemically-treated collagenous tissue nonlinear anisotropic stress-strain responses from microscopy images,” *Acta Biomaterialia*, vol. 63, pp. 227–235, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1742706117305883>
- [33] J. Ghaboussi, J. H. Garrett, and X. Wu, “Knowledge-based modeling of material behavior with neural networks,” *Journal of Engineering Mechanics*, vol. 117, no. 1, pp. 132–153, 1991. [Online]. Available: <https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9399%281991%29117%3A1%28132%29>
- [34] X. Wu, J. G. J. H, and J. Ghaboussi, “Representation of material behavior: neural network-based models,” in *1990 IJCNN International Joint Conference on Neural Networks*, 1990, Conference Proceedings, pp. 229–234 vol.1.

- [35] Y. M. A. Hashash, S. Jung, and J. Ghaboussi, “Numerical implementation of a neural network based material model in finite element analysis,” *International Journal for Numerical Methods in Engineering*, vol. 59, no. 7, pp. 989–1005, 2004. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.905>
- [36] K. Linka, M. Hillgärtner, K. P. Abdolazizi, R. C. Aydin, M. Itskov, and C. J. Cyron, “Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning,” *Journal of Computational Physics*, vol. 429, p. 110010, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0021999120307841>
- [37] H. Dal, F. A. Denli, A. K. Aan, and M. Kaliske, “Data-driven hyperelasticity–part i: A canonical isotropic formulation for rubberlike materials,” *Available at SSRN 4386964*, 2023. [Online]. Available: <https://ssrn.com/abstract=4386964>
- [38] F. As’ad, P. Avery, and C. Farhat, “A mechanics-informed artificial neural network approach in data-driven constitutive modeling,” *International Journal for Numerical Methods in Engineering*, vol. 123, no. 12, pp. 2738–2759, 2022. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6957>
- [39] K. A. Kalina, L. Linden, J. Brummund, P. Metsch, and M. Kästner, “Automated constitutive modeling of isotropic hyperelasticity based on artificial neural networks,” *Computational Mechanics*, vol. 69, no. 1, pp. 213–232, 2022. [Online]. Available: <https://doi.org/10.1007/s00466-021-02090-6>
- [40] V. Tac, K. Linka, F. Sahli-Costabal, E. Kuhl, and A. Buganza Tepole, “Benchmarks for physics-informed data-driven hyperelasticity,” p. arXiv:2301.10714, January 01, 2023 2023. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/2023arXiv230110714T>
- [41] K. Linka and E. Kuhl, “A new family of constitutive artificial neural networks towards automated model discovery,” *Computer Methods in Applied Mechanics and Engineering*, vol. 403, p. 115731, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0045782522006867>
- [42] V. Tac, V. D. Sree, M. K. Rausch, and A. B. Tepole, “Data-driven modeling of the mechanical behavior of anisotropic soft biological tissue,” *Engineering with Computers*, vol. 38, no. 5, pp. 4167–4182, 2022. [Online]. Available: <https://doi.org/10.1007/s00366-022-01733-3>
- [43] J. N. Fuhg, N. Bouklas, and R. E. Jones, “Learning hyperelastic anisotropy from data via a tensor basis neural network,” *Journal of the Mechanics and Physics of Solids*, vol. 168, p. 105022, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0022509622002058>

- [44] J. Kudela and R. Matousek, “Recent advances and applications of surrogate models for finite element method computations: a review,” *Soft Computing*, vol. 26, no. 24, pp. 13 709–13 733, 2022. [Online]. Available: <https://doi.org/10.1007/s00500-022-07362-8>
- [45] A. Madani, A. Bakhaty, J. Kim, Y. Mubarak, and M. R. K. Mofrad, “Bridging finite element and machine learning modeling: Stress prediction of arterial walls in atherosclerosis,” *Journal of Biomechanical Engineering*, vol. 141, no. 8, 2019. [Online]. Available: <https://doi.org/10.1115/1.4043290>
- [46] D. Lorente, F. Martínez-Martínez, M. J. Rupérez, M. A. Lago, M. Martínez-Sober, P. Escandell-Montero, J. M. Martínez-Martínez, S. Martínez-Sanchis, A. J. Serrano-López, C. Monserrat, and J. D. Martín-Guerrero, “A framework for modelling the biomechanical behaviour of the human liver during breathing in real time using machine learning,” *Expert Systems with Applications*, vol. 71, pp. 342–357, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417416306728>
- [47] F. Martínez-Martínez, M. J. Rupérez-Moreno, M. Martínez-Sober, J. A. Solves-Llorens, D. Lorente, A. J. Serrano-López, S. Martínez-Sanchis, C. Monserrat, and J. D. Martín-Guerrero, “A finite element-based machine learning approach for modeling the mechanical behavior of the breast tissues under compression in real-time,” *Computers in Biology and Medicine*, vol. 90, pp. 116–124, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0010482517303177>
- [48] R. I. Mukhamediev, Y. Popova, Y. Kuchin, E. Zaitseva, A. Kalimoldayev, A. Symagulov, V. Levashenko, F. Abdoldina, V. Gopejenko, K. Yakunin, E. Muhamedijeva, and M. Yelis, “Review of artificial intelligence and machine learning technologies: Classification, restrictions, opportunities and challenges,” *Mathematics*, vol. 10, no. 15, 2022. [Online]. Available: <https://www.mdpi.com/2227-7390/10/15/2552>
- [49] J. Bell, *Machine Learning: Hands-On for Developers and Technical Professionals*. John Wiley & Sons, Inc., 2014.
- [50] J. D. Kelleher, *Deep learning*. MIT press, 2019.
- [51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [52] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.
- [53] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter optimization for machine learning models based on bayesian optimizationb,” *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1674862X19300047>

- [54] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in Neural Information Processing Systems*, vol. 4, p. 2951 – 2959, 2012, cited by: 3739. [Online]. Available: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84869201485&partnerID=40&md5=14aa83df115308a2cf91468d634a36aa>
- [55] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 2623 – 2631, 2019, cited by: 970.
- [56] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
- [57] G. A. Holzapfel, *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. England: John Wiley & Sons, Ltd, 2000.
- [58] J. Bonet and R. Wood, *Nonlinear Continuum Mechanics for Finite Element Analysis*. USA: Cambridge University Press, 2008.
- [59] G. C. Johnson and D. J. Bammann, “A discussion of stress rates in finite deformation problems,” *International Journal of Solids and Structures*, vol. 20, no. 8, pp. 725–737, 1984. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0020768384900611>
- [60] C. Truesdell, “The simplest rate theory of pure elasticity,” *Communications on Pure and Applied Mathematics*, vol. 8, no. 1, pp. 123–132, 1955. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160080109>
- [61] J. G. Oldroyd and A. H. Wilson, “On the formulation of rheological equations of state,” *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 200, no. 1063, pp. 523–541, 1950. [Online]. Available: <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1950.0035>
- [62] A. E. Green and P. M. Naghdi, “A general theory of an elastic-plastic continuum,” *Archive for Rational Mechanics and Analysis*, vol. 18, pp. 251–281, 1965.
- [63] W. PRAGER, “An elementary discussion of definitions of stress rate,” *Quarterly of Applied Mathematics*, vol. 18, no. 4, pp. 403–407, 1961. [Online]. Available: <http://www.jstor.org/stable/43634819>

- [64] M. Mooney, “A theory of large elastic deformation,” *Journal of Applied Physics*, vol. 11, no. 9, pp. 582–592, 1940. [Online]. Available: <https://doi.org/10.1063/1.1712836>
- [65] R. S. Rivlin and E. K. Rideal, “Large elastic deformations of isotropic materials iv. further developments of the general theory,” *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 241, no. 835, pp. 379–397, 1948. [Online]. Available: <https://royalsocietypublishing.org/doi/abs/10.1098/rsta.1948.0024>
- [66] G. Marckmann and E. Verron, “Comparison of hyperelastic models for rubber-like materials,” *Rubber Chemistry and Technology*, vol. 79, pp. 835–858, 2006.
- [67] R. W. Ogden and R. Hill, “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*, vol. 326, no. 1567, pp. 565–584, 1972. [Online]. Available: <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1972.0026>
- [68] L. R. G. Treloar, “Stress-strain data for vulcanised rubber under various types of deformation,” *Transactions of The Faraday Society*, vol. 40, pp. 59–70, 1944.
- [69] J. D. Humphrey and F. C. P. Yin, “On Constitutive Relations and Finite Deformations of Passive Cardiac Tissue: I. A Pseudostrain-Energy Function,” *Journal of Biomechanical Engineering*, vol. 109, no. 4, pp. 298–304, 11 1987. [Online]. Available: <https://doi.org/10.1115/1.3138684>
- [70] J. Martins, E. Pires, R. Salvado, and P. Dinis, “A numerical model of passive and active behavior of skeletal muscles,” *Computer Methods in Applied Mechanics and Engineering*, 1998. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S004578259700162X>
- [71] G. Holzapfel, T. Gasser, and R. Ogden, “A new constitutive framework for arterial wall mechanics and a comparative study of material models,” *Journal of Elasticity*, vol. 61, no. 1-3, pp. 1–48, 2000.
- [72] T. C. Gasser, R. W. Ogden, and G. A. Holzapfel, “Hyperelastic modelling of arterial layers with distributed collagen fibre orientations,” *Journal of The Royal Society Interface*, vol. 3, no. 6, pp. 15–35, 2006. [Online]. Available: <https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2005.0073>
- [73] W. Falcon and T. P. L. team, “Pytorch lightning,” Mar. 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7786025>
- [74] K. Sawarkar, *Deep Learning with PyTorch Lightning*. Packt Publishing Ltd., 2022.

- [75] J. Schröder, T. Wick, S. Reese, P. Wriggers, R. Müller, S. Kollmannsberger, M. Kästner, A. Schwarz, M. Igelbüscher, N. Viebahn, H. R. Bayat, S. Wulfinghoff, K. Mang, E. Rank, T. Bog, D. D’Angella, M. Elhaddad, P. Hennig, A. Düster, W. Garhuom, S. Hubrich, M. Walloth, W. Wollner, C. Kuhn, and T. Heister, “A selection of benchmark problems in solid mechanics and applied mathematics,” *Archives of Computational Methods in Engineering*, vol. 28, no. 2, pp. 713–751, Mar 2021. [Online]. Available: <https://doi.org/10.1007/s11831-020-09477-3>
- [76] R. D. Cook, “Improved two-dimensional finite element,” *Journal of the Structural Division*, vol. 100, pp. 1851–1863, 1974.
- [77] Dassault Systèmes Simulia Corp., *Abaqus Analysis User’s Manual, Volume I*, Providence, RI, 2011.