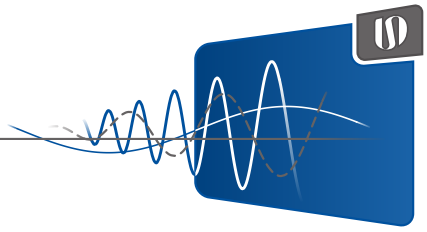
Institute of Mechanics,
Structural Analysis and Dynamics

Univ.-Prof. Dr.-Ing. Tim Ricken



Master-Thesis

# A data driven efficient multiscale computational tool using hybrid AI modelling framework for short fiber reinforced thermoplastics

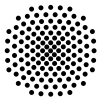
Pavan Bhat Keelanje Srinivas

May 15, 2022

Report No. 106

Editor: Univ.-Prof. Dr.-Ing. Tim Ricken
Supervisor: Dipl-Phys. Andre Mielke (ISD)
Dr.-Ing Fabian Welschinger,MSc Argha Protim Dey (RB GmbH)

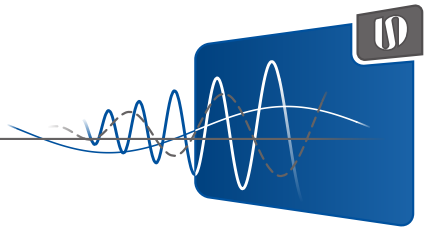
Report of the Institute of Mechanics, Structural Analysis and Dynamics



**University of Stuttgart**
Germany

Institute of Mechanics,
Structural Analysis and Dynamics

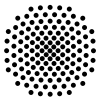
Univ.-Prof. Dr.-Ing. Tim Ricken



Master-Thesis

# A data driven efficient multiscale computational tool using hybrid AI modelling framework for short fiber reinforced thermoplastics

| | |
|---|---|
| Author: | Pavan Bhat Keelanje Srinivas |
| Report: | No. 106 |
| Start Date: | November 15, 2021 |
| Submitted: | May 15, 2022 |
| Editor: | Univ.-Prof. Dr.-Ing. Tim Ricken |
| Supervisor: | Dipl-Phys. Andre Mielke (ISD) |
| | Dr.-Ing Fabian Welschinger,MSc Argha Protim Dey (RB GmbH) |
| Institute: | Institute of Mechanics,<br>Structural Analysis and Dynamics<br>Pfaffenwaldring 27<br>70569 Stuttgart<br>University of Stuttgart |
| Company: | Robert Bosch GmbH Corporate Research (RB GmbH)<br>Robert-Bosch-Campus 1<br>71272 Renningen |



**University of Stuttgart**
Germany

# Task Definition

## for Pavan Bhat Keelanje Srinivas

**Title:** A data driven efficient multiscale computational tool using hybrid AI modelling framework for short fiber reinforced thermoplastics

**Topic:** Conducting complex experiments on fiber reinforced thermoplastics is very time and cost intensive. An alternative solution is to create accurate digital twins of the material based on the results of micro-CT scans till the component level. Multiscale simulations hold good promise in modelling the material behaviour accurately. State of art mathematical methods used to bridge the scales are resource intensive, so an alternative reliable hybrid machine learning solution method is proposed. An extension to the machine learning method is to be implemented to generalize the solver to component level microstructure variations. The steps involved in identifying such a solver for a chosen material is to be realized. A data science platform to automate and ease the solver identification and validation process is to be created. The platform is to be proved for a chosen material for quasi-static loading scenario within engineering accuracy of industrial use. The validation procedure for long term loading case is to be benchmarked.

**Supervisor:** Dipl-Phys. Andre Mielke (ISD)
Dr.-Ing Fabian Welschinger,MSc Argha Protim Dey (RB GmbH)

**Editor:** Univ.-Prof. Dr.-Ing. Tim Ricken

**Start Date:** November 15, 2021

**Submitted:** May 15, 2022

**Institute:** Institute of Mechanics,
Structural Analysis and Dynamics
Pfaffenwaldring 27
70569 Stuttgart

**Company:** Robert Bosch GmbH Corporate Research (RB GmbH)
Robert-Bosch-Campus 1
71272 Renningen

# Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe, dass ich keine anderen als die angegebenen Quellen benutzt habe und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist, dass ich die Arbeit weder vollständig noch in Teilen bereits veröffentlicht habe und dass das elektronische Exemplar mit den anderen Exemplaren übereinstimmt.

---

Place, Date Pavan Bhat Keelanje Srinivas

# Abstract

During product development, there is a need to characterize the change in material behaviour to evaluate component reliability. Experimental evaluation of material behavior is complex and time consuming. An alternative is to build accurate multiscale simulation models to replace experiments and create virtual experimental data. Since these changes are at the microscale, translating this information into a macroscale component level simulation is time intensive. In the case of a two phase fiber reinforced composites, both phases of a representative volume element(RVE) considered at the microscale is homogenized into a single phase using traditional full field solvers on microstructure CT-images. This work dicusses a hybrid machine learning modelling approach as an alternative to traditional full-field simulations at the microscale. In this work, an extension to the state of the art hybrid machine learning model is performed to generalize the solver to different microstructure configurations. The steps for identifying such a model is presented and a framework for automating such a process in the high performance computing (HPC) environment is realized. The framework NumProDeep is a new paradigm of a datascience platform analogous to FE software architecture used to identify such a model. The working of the framework is proved using the quasi-static loading case for plasticity, as a case study. The gains in speeds in data generation and processing is seen to be at least three folds. A deterministic algorithm developed is found to reduce 98% of the validation experiments for the particular case study. The best model was found to preserve accuracy within 10.91% maximum error compared to traditional full field solver. The same model was validated for long term loading case on few microstructure configurations without the aid of the framework. The resulting evaluation was found to have high errors due to high nonlinearity of the problem. Future direction for long term loading case were benchmarked and final results were presented in this work.

**Keywords:** Multiscale simulation, numerical methods, machine learning, homogenization, hyper parameter tuning

# Contents

**Task Definition** i

**Eidesstattliche Erklärung** iii

**Abstract** v

**List of Figures** ix

**List of Tables** xi

**1 Introduction** 1
1.1 Motivation . . . 1

**2 Theoretical background** 3
2.1 Short fiber reinforced thermoplastics (SFRTs) . . . 3
2.1.1 What are SFRTs? . . . 3
2.1.2 Fiber orientation tensor . . . 3
2.1.3 X-ray micro-computed tomography ($\mu$CT) . . . 6
2.1.4 Micro-structure generation and parameters . . . 8
2.2 Multiscale modelling and homogenization . . . 9
2.2.1 Analytical homogenization methods . . . 11
2.2.2 Cell problem and FFT numerical homogenization methods . . . 11
2.3 Plasticity and creep coupled model . . . 12
2.3.1 Kinematics . . . 13
2.3.2 Free energy function definition, driving forces and stresses . . . 13
2.3.3 Dissipation function . . . 14
2.3.4 Parameter identification . . . 15
2.4 Data driven methods and hybrid modelling . . . 16
2.4.1 Hybrid modelling and its need . . . 17
2.5 Deep material networks . . . 19
2.5.1 Offline training-direct deep material network . . . 19
2.5.2 Offline training- deep material network with interpolation . . . 22
2.5.3 Online evaluation . . . 24

**3 Methodology** 27
3.1 NumProDeep-A Framework to identify Machine learning models . . . 27
3.1.1 Discretization and FE software architecture . . . 28
3.1.2 Element and bisection algorithm . . . 30
3.1.3 Node-linear . . . 32

3.1.4 Node-nonlinear . . . 34
3.1.5 Microstructure parameters . . . 35
3.1.6 Material sampling and sampling size . . . 36
3.1.7 Offline phase implementation . . . 37
3.1.8 Online evaluation and methods . . . 40
3.1.9 Selective distillation algorithm . . . 46

**4 Numerical investigation and results 49**
4.1 Training and validation data generation . . . 49
4.2 Ground truth data generation . . . 49
4.3 Training Framework . . . 50
4.4 Online evaluation using selective distillation algorithm . . . 50
4.5 Hyper-parameters and numerical experiments for plasticity . . . 51
4.6 Creep or long term loading . . . 53
4.7 Identified model in the context of multiscale simulations . . . 56

**5 Conclusion and outlook 57**
5.1 Conclusions . . . 57
5.2 Outlook . . . 58

**6 Individual plots 61**

**Bibliography 65**

# List of Figures


2.1 Some examples of parts manufactured using SFRT . . . . . 4
2.2 Skin core strcuture . . . . . 4
2.3 A vector representation of fiber [2] . . . . . 5
2.4 **(*a*)** shows the Fiber Orientation Triangle. The bounding vertices of the triangle are shown in the images as **(*b*)** Uni-directional,**(*c*)** Planar-Isotropic,**(*d*)** Fully-Isotropic . . . . . 7
2.5 $\mu-$CT scan of microstructure. . . . . 8
2.6 Dogbone sample testing [52] . . . . . 10
2.7 The two scale homogenization approach [48] . . . . . 10
2.8 Framing a peridically microstructured material to a homogenized material [46] . . . . . 12
2.9 Classical Approach[35] . . . . . 16
2.10 Machine Learning Approach [35] . . . . . 17
2.11 Approaches to perform Homogenization . . . . . 18
2.12 The figure shows an intuitive understanding of the offline phase . . . 20
2.13 The figure shows a two Phase Deep Material Network . . . . . 21
2.14 Approach to calculate the direction of lamination for a single 2 phase laminate of an interpolated DMN . . . . . 24

3.1 Overview of the steps involved in the model identification workflow . 28
3.2 The figure shows different levels of sampling on the FOT triangle . . 29
3.3 A basic class diagram showing the components of the NumProDeep framework . . . . . 30
3.4 Folder structure tracked by Element discretization object . . . . . 31
3.5 D10 discretization having four dummy Element objects . . . . . 32
3.6 Bisection algorithm outputs . . . . . 32
3.7 Folder structure tracked by Node class object . . . . . 33
3.8 Node naming of D10 discretization . . . . . 34
3.9 Folder structure tracked by Nonlinear Node class object . . . . . 35
3.10 Boundary condition for plasticity [4] . . . . . 42
3.11 Boundary condition for creep . . . . . 43
3.12 Online evaluation . . . . . 44
3.13 Error plot of all six loading cases of plasticity over the entire epoch range. This work is more concerned with the maximum of all the errors. The verticle line points out the epoch state at which the DMN has the lowest error in the entire range. . . . . 45
3.14 The image shows the major stages in the framework . . . . . 46

4.1 Aggregated plots of both Maximum and mean error shown along with the training curve . . . . . 52
4.2 Plots showing the D4 creep evaluation comparision of DMN and full nonlinear FFT . . . . . 55

6.1 Plots showing distribution for E11 load case . . . . . 62
6.2 Plots showing distribution for E22 load case . . . . . 62
6.3 Plots showing distribution for E33 load case . . . . . 62
6.4 Plots showing distribution for E12 load case . . . . . 63
6.5 Plots showing distribution for E23 load case . . . . . 63
6.6 Plots showing distribution for E13 load case . . . . . 63

# List of Tables


2.1 Material parameters for PBT GF30. . . . . 16
2.2 Shape function used for interpolating the angles over the FOT triangle 23

3.1 Naming convention for Nodes . . . . . 34
3.2 Chosen microstructure parameters. . . . . 36
3.3 Sampling size . . . . . 37
3.4 Hyper parameter chosen . . . . . 41
3.5 Load levels chosen for D4 discretization . . . . . 43
3.6 Time taken for full evaluation . . . . . 46

4.1 Time expense comparision for ground truth generation. . . . . 50
4.2 Timeline for model training . . . . . 50
4.3 Time comparision for full evaluation . . . . . 51
4.4 Values achieved in online evaluation phase of each loadcase . . . . . 52
4.5 Comparision of long term loading on complex and DMN microstructure 54
4.6 Comparision of simulation runtime in long term loading on complex and DMN microstructure . . . . . 56
4.7 Comparision of compute time for multiscale simulation of plasticity . 56

# 1 Introduction

## 1.1 Motivation

Modern products see an increased use of advanced light weight engineered materials. During the product development stage, material selection plays a central role for the design of the components due to the different loading conditions and environment the product is exposed to. Given that most of these materials are increasingly being used in high performance applications, there is a need to understand the safety and reliability of these manufactured components. At an early stage of development, making prototypes and conducting complex experiments is very time consuming when compared to the project timelines. This calls for a need for accurate simulations to evaluate reliability. A particular focus of this work is performance in predicting the quasi-static deformation due to monotonic loading and the creep deformation due to long-term constant force loading which are highly nonlinear load cases.

One material of interest is short fiber reinforced thermoplastics (SFRTs). These materials are made of the mixture of glass fibers which have high strength with polymer matrix which are relatively weaker. Although the composite material now has a higher strength, there exists a high variability in its properties. This is mainly attributed to varying topology and complexities imposed by manufacturing methods. Characterizing the microscale topological level behaviour under varied loading scenarios, into the macroscale simulation is a major challenge for these materials. These aspects are very important because processes like injection molding and compression molding might cause changes in its properties which are difficult to characterize only with experiments. This is where the multiscale approach to structural simulations hold considerable promise. The technology shows the way to inversely identify material model parameters by matching simulations with sparsely available experimental results. Once the material models are identified many variations of the testings can be performed virtually, thus reducing the need for experiments. One could imagine a sparsely filled material database with only few experimental testing results. Such a database can be filled by virtual experimental data on demand using these accurate multiscale simulations at every stage of the processing.

The traditional numerical methods available for bridging the scales are highly accurate, but time and cost intensive. The state of art methods available in literature, aim to provide alternative fast methods to make multiscale simulations a reality. Most

of these methods range from purely data-driven methods to techniques that reduces the model order of physics. One such alternative which holds good promise is a hybrid machine learning modelling solution which forms the focus of this work. Given that data is very limited in product engineering, combining the strengths of machine learning and rich knowledge of first principle physics opens-up unique opportunity and increases the modelling capabilities to understand material behaviour. Identifying such a model for given material is very complex and ambiguous process. This work presents the steps involved in identifying such a model along with the challenges involved in validating such a model. In the process it proposes a highly customizable automation framework called NumProDeep, with inbuilt algorithms that make the hybrid machine learning model identification process easier.

# 2 Theoretical background

## 2.1 Short fiber reinforced thermoplastics (SFRTs)

### 2.1.1 What are SFRTs?

Thermoplastics are the class of polymers that under higher temperature becomes a viscous moldable fluid and remains solid at normal room temperature. Due to its properties and ease of manufacturing, this class of polymers is used to produce variety of goods from biological prosthetics, bike parts to spacecrafts. One of the attempts to improve the mechanical properties of the thermoplastics is SFRT where the fibers having a higher stiffness are spatially distributed into the polymer matrix to obtain a high performance composite material [45]. Figure 2.1 shows some common high performance plastics that are manufactured using short fiber reinforced thermoplastics.

As can be seen in Liu [1] injection molding and compression molding are two major manufacturing methods for short and discontinuous fiber reinforced composites. Molding allows for large volume of production of parts and good control over the quality of the parts produced. Like any manufacturing process injection molding comes with its own disadvantages. Costantino, Rosales and Pettarin [32] points out many key problems associated to molding. Having multiple injection points or several mold flow fronts causes weld lines. Another defect that was pointed out was the skin core effect where in the morphology of fiber distribution are different in the boundaries and along the thickness of the part. Lutz et al.[31] gives a description of how these defects cause a reduction in mechanical properties and ultimately act as potential failure points for a given loading condition. Thus one can see that capturing these abnormalities becomes extremely important while one develops accurate simulation models resulting in the importance of characterization of the microstructure.

### 2.1.2 Fiber orientation tensor

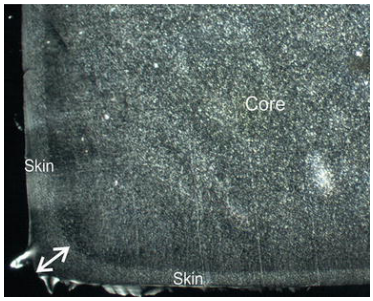
The spatial arrangement and distribution of fibers in the matrix has a significant effect on the properties of the composite material [45]. Thus emerges a need for quantifying fiber allignment in the matrix. So mathematically this can be achieved through fiber

**(a)** A spinal implant made of fiber-plastic

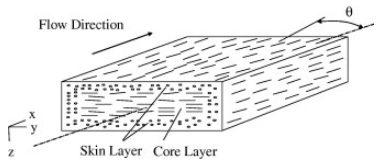
**(b)** A high pressure air hose clamp

**(c)** Structural plastic handels

**Figure 2.1:** Some examples of parts manufactured using SFRT




**(a)** Skin-core microstructure [32]




**(b)** Skin core structure schematic [31]

**Figure 2.2:** Skin core strcuture

orientation tensor (FOT). FOT was discussed in the paper by Advani and Tucker [2]. In its simplest form a fiber can be represented as a rigid body vector $\boldsymbol{p}$ with a length and diameter oriented in 3D space with angles $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$. The schematics can be seen in figure 2.3. The spatial components of $\boldsymbol{p}$ can be resolved as shown in equation 2.1,

$$\begin{aligned} p_x &= \sin(\theta)\sin(\varphi) \\ p_y &= \sin(\theta)\cos(\varphi) \\ p_z &= \cos(\theta). \end{aligned} \tag{2.1}$$

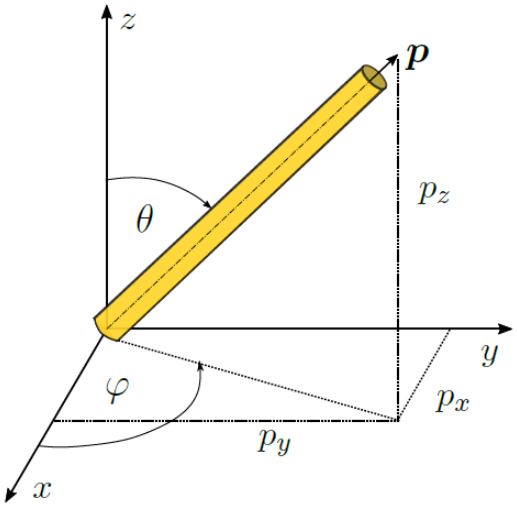



**Figure 2.3:** A vector representation of fiber [2]

When it comes to quantifying fibers distributed in a matrix then probabilities of fibers along the fiber directions are considered. This is denoted as probability distribution function (PDF) $\psi(\boldsymbol{p})$. The sum of probability over all the directions are given by the equation

$$\oint \psi(\boldsymbol{p})\mathrm{d}\boldsymbol{p} = 1. \tag{2.2}$$

Since the PDF is computationally expensive, a Fourier expansion is performed as per Advani and Tucker [2] and n-th order coefficient of the expansion are chosen, which are nothing but the FOT. These tensors are the moments of the $\psi(\boldsymbol{p})$. The FOT is a more efficient way to represent fiber orientation information in simulations than PDF [24]. Also higher the order of these moments more is the information they hold. Further because the PDF is always positive, they always occur in even order. As per Müller and Böhlke [39] second order fiber orientation tensor is seen to be fair enough approximation for material properties calculations. This work chooses the second order fiber orientation tensor $\boldsymbol{A}_2$ as shown in equation 2.3. For higher order FOT

one can refer to the work by Montgomery-smith et al.[37].

$$\begin{aligned} \boldsymbol{A}_2 &= \int p_i \otimes p_j \psi(\boldsymbol{p}) \mathrm{d}\boldsymbol{p} \\ \boldsymbol{A}_4 &= \int p_i \otimes p_j \otimes p_k \otimes p_l \psi(\boldsymbol{p}) \mathrm{d}\boldsymbol{p} \end{aligned} \tag{2.3}$$

The tensor $\boldsymbol{A}_2$ is positive definite and under decomposition the diagonal matrix has an unit trace. Gajek et al. [10] explains the decomposition of the $\boldsymbol{A}_2$ as given by equation 2.4, which shows five independent parameters. The matrix $\boldsymbol{Q} \in SO(3)$ is orthogonal and specifies the principle directions under decomposition. The diagonal eigen values show relationship $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. This relationship allows for any given microstructure to be identified by just two parameters $\lambda_1$, $\lambda_2$.

$$\boldsymbol{A}_2 = \boldsymbol{Q} \operatorname{diag}(\lambda_1, \lambda_2, \lambda_3) \boldsymbol{Q}^T \tag{2.4}$$

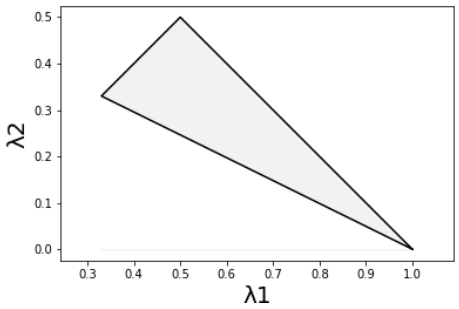
$$\frac{1}{3} \leq \lambda_1 \leq 1 \quad \text{and} \quad 1 - 2\lambda_1 \leq \lambda_2 \leq \lambda_1 \tag{2.5}$$

Owing to the constraints imposed by these parameters the entire space of possible configuration of microstructure can be reduced to inequalities as shown in equation 2.5. This defines a triangular domain called the Fiber Orientation Triangle bounded by three microstructure configuration unidirectional, isotropic and planar isotropic arrangement of fibers in polymer matrix. The figure 2.4 shows the Triangle and the microstructure configurations of the corner vertices.

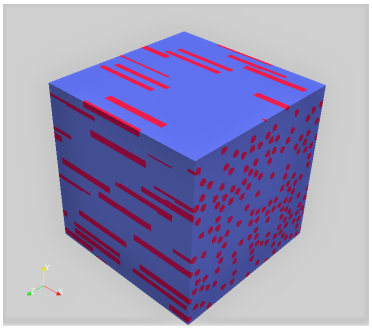
### 2.1.3 X-ray micro-computed tomography ($\mu$CT)

The last section mainly dealt with the mathematical tools that were developed for quantifying a material microstructure of the fiber reinforced composites. In this section one deals with the tools that allow us to see the topology of the microstructure itself. As pointed out in the section 2.1.1 the injection molding process introduces a variety of heterogeneity that makes it essential to study the microstructural evolution of such processes. One such tools that helps us see this is the X-ray micro-computed tomography.

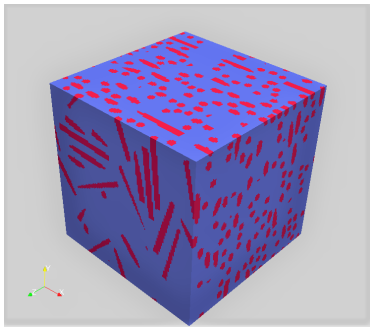
Micro-computed tomography is a non destrcutive methodology for such applications where one needs to obtain structural information of an inhomogeneous material. It can be employed when the multiple phases within the material of interest have widely different absorption rate for X-rays. One such experiment was performed at Robert Bosch GmbH. The evaluation of fiber length distribution, fiber orientation and fiber volume content were done using methods discussed in Hessman et al. [18]. Further



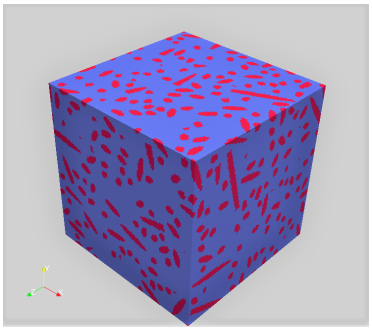

**(a)** Fiber Orientation Tensor Triangle



**(b)** $\lambda_1 = 1.0, \lambda_2 = 0.0$



**(c)** $\lambda_1 = 0.5, \lambda_2 = 0.5$



**(d)** $\lambda_1 = 0.33, \lambda_2 = 0.33$

**Figure 2.4:** $(a)$ shows the Fiber Orientation Triangle.
The bounding vertices of the triangle are shown in the images as $(b)$ Uni-directional,$(c)$ Planar-Isotropic,$(d)$ Fully-Isotropic

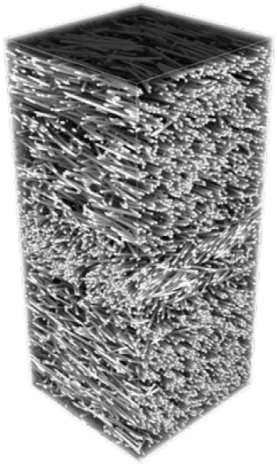


**Figure 2.5:** $\mu-$CT scan of microstructure.

Garcea et al. [12] provide an in-depth discussion on the various features that can be measured, loading scenarios that can be studied and associated hurdles that are faced while conducting the study [12].

The material that was chosen for performing $\mu$CT scans is 30% glass fiber reinforced polybutylenterephthalate (PBT GF30). The specimen is milled out of a sheet of dimensions 120 mm x 80 mm x 2 mm that is injection molded.The resulting digital image of the $\mu$CT can be seen in the Fig. 2.5. The structural information is extracted from the high resolution volume photos as shown in Fig. 2.5 using layer wise calculation as described in Robb et al .[44].

In this context, microstructure image segmentation is the process of marking a voxel as fiber or matrix.To segment 3D $\mu$CT image data, a commercial software application called ScanIP [51] is available.The software thresholds the voxels based on their gray scale and marks the voxel into matrix or fiber.The fibers that appear in contact are also distinctly segmented by subtracting the interface voxels. Using ScanIP [51] one is able to compute the second and fourth order fiber orientation tensor, length-weighted mean fiber length and volume fraction which are useful for synthetic microstructure generation.

### 2.1.4 Micro-structure generation and parameters

Along with the fiber orientation tensor which captures the spatial and strcutural information of the microstructure, there are few other parameters relevant to define a microstructure in its entirety. Once these parameters are defined, in order to perform numerical analysis on the microstructure one also needs to generate a synthetic microstructure.

### Volume fraction

The term describes the amount of material that is contained in the composite as a fraction of the total material contained. It is usually referred as a percentage of fibers contained in the material and is given by the equation 2.6. Here $w_f$ is the weight fraction of fibers, $\rho_f$ is the density of fibers and $\rho_m$ is the density of matrix.

$$v_f = \frac{w_f/\rho_f}{w_f/\rho_f + (1 - w_f)/\rho_m} \tag{2.6}$$

### Microstructure generation

As discussed in section 2.1.3 and also in section 2.1.2 a microstructure composed of the SFRT, comes with huge amount of variability in its topology. This makes performing analysis on the digital image extremely challenging. Few of the variables are fiber distribution, fiber proximity, varying levels of fiber length/diameter, the shape of the fiber ends, contact between the fibers and many more. It is necessary to consider assumptions and simplifications of the topology so that there is a possibility to perform computations on a statistically similar microstructure.

So this work uses FiberGenerator tool available within robert-bosch for plotting digital twins of the microstructure. The tool is based on sequential addition and migration algorithm for generating representative volume element (RVE) proposed by Schneider [47]. Given variable like fiber orientation tensor, volume fraction and fiber aspect ratio, the method systematically adds and removes fibers to ensure that fibers do not overlap.

## 2.2 Multiscale modelling and homogenization

The field of computational mechanics of materials deal with modelling the behaviour of different materials under various boundary conditions. The subject deals with hollistic, purely phenomenological approach to the modelling of material reponses. The word phenomenological means that one performs experiments, watches the behaviour of the material during experiments and infers a model based on observations [22]. A typical tensile testing of a dogbone sample can be seen in figure 2.6.

These modelling frameworks are built by first solving universal balance laws based on thermodynamical equations. And then, in order to completely model the behaviour of the material so called constitutive equations are required which govern the individual material response [33]. These constitutive equations are derived by material experts who derive these based on their experimental inference of the material behaviour.

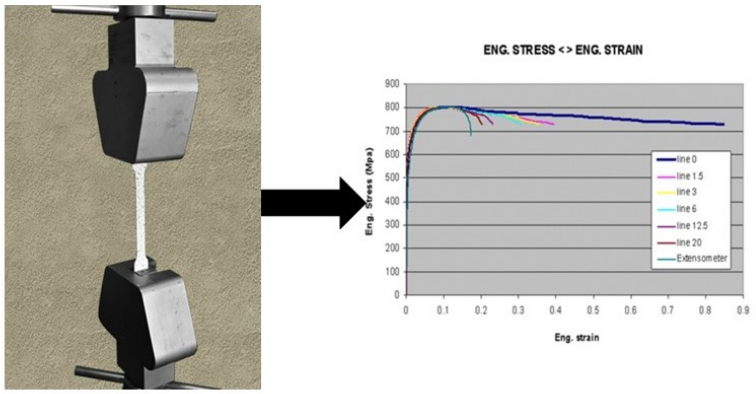



**Figure 2.6:** Dogbone sample testing [52]

For simple loading scenarios and single phase homogeneous materials these equations capture the macroscopic behaviour of the material response. For a multiphase inhomogeneous composite material like fiber reinforced plastics where one of the material is much stiffer than the matrix the properties vary based on the distribution of these fibers in the matrix. Moreover if one was tasked at running an FEM at this scale, the resulting meshing has to be very fine resulting in large simulation times [55].

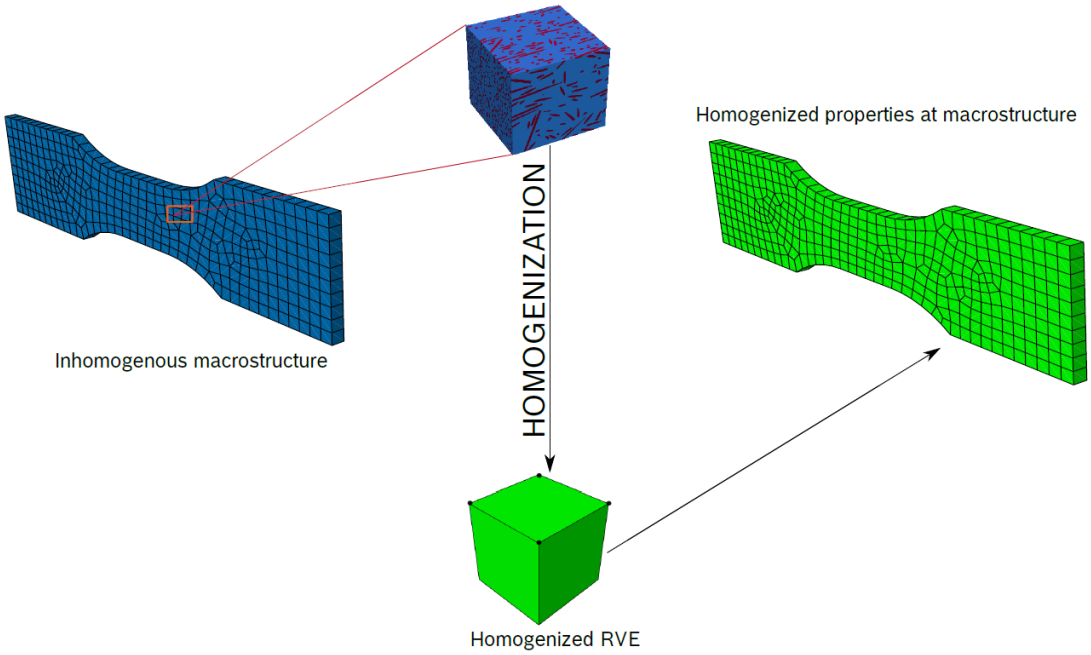



**Figure 2.7:** The two scale homogenization approach [48]

$$\bar{\varepsilon} = \frac{1}{V}\int_V \varepsilon(x)dV \quad , \quad \bar{\sigma} = \frac{1}{V}\int_V \sigma(x)dV \tag{2.7}$$

To solve this Zohdi et al. [55] proposes a two scale framework, one on the microscale and another on the macroscale. The work proposes considering a small volume of the structure and averaging the properties in that volume called the representative volume element (RVE) which was originally proposed by Hill et al. [19]. Then the properties from RVE are used in finite element simulation as a single homogeneous material. So homogenization is a method to frame an inhomogeneous continuum into a homogeneous material in order to derived so called effective material model of the inhomogeneous RVE. The equation 2.7 shows averaging over microscopic local stress field $\sigma(x)$ and microscopic local strain field $\varepsilon(x)$ to macroscopic stress $\bar{\sigma}$ and macroscopic strain $\bar{\varepsilon}$ in the RVE. Siddique [48] gives a very good pictorial dipiction in the context of the current setup at Bosch as shown in figure 2.7.

The realm of homogenization methods to derive the effective material model can be broadly classified as numerical homogenization and analytical homogenization techniques.

### 2.2.1 Analytical homogenization methods

Alouges et al. [3] give an introductory account of the analytical methods that are used in multiscale problems. Some of these methods that are discussed are multicale expansion methods, two scale convergence and energy convergence methods for deriving solutions for the cell problem.

### 2.2.2 Cell problem and FFT numerical homogenization methods

SFRT is an inhomogeneous material where-in one has fibers in sizes of microns spatially distributed in components of sizes of few centimeter to meters. Homogenization provides a framework to represent the inhomogenity in the microscale material response into the macroscopic component behaviour under specified boundary conditions.This is done by solving a coupled system of equations on the macroscale and the microscale.

A thorough introductory reference can be found in the course 'Computational Homogenization on Digital Image Data' offered by Jun-Prof. Dr. Matti Schneider at KIT [46]. Consider a periodically microstructured material domain $\Omega$ with periodic cell Y of length L as depicted in figure 2.8. Here $\eta$ is a scalar value and is taken as a quotient of microscopic and macroscopic scale. Here the aim is to study the homogenized material behaviour for $\eta \to 0$.

Taking a simple case of linear elastic scenario one obtains a coupled system of equations as shown in equation 2.8. The first equation indicates the cell problem solved at the microscale and second equation is the effective elastic equation solved as an average values over the cell Y.

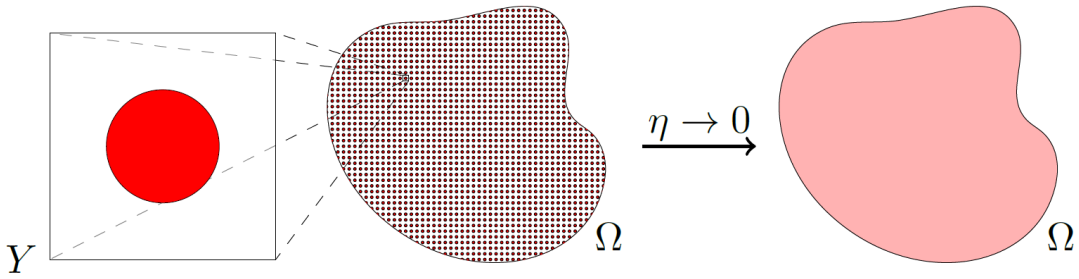



**Figure 2.8:** Framing a peridically microstructured material to a homogenized material [46]

$$\begin{aligned} \text{micro}: \ 0 &= \text{div}_y \left[ \mathbb{C} : \left( \nabla_x^s u + \nabla_y^s v \right) \right] \\ \text{macro}: \ -f &= \text{div}_x \left\langle \mathbb{C} : \left( \nabla_x^s u + \nabla_y^s v \right) \right\rangle_{\text{Y}} \end{aligned} \tag{2.8}$$

The cell problem is obtained by solving the balance of linear momentum in the microscale of $y$. The effective elastic equation is obtained by solving the balance of linear momentum in the macroscale as an average over the cell Y, shown in equation 2.8. Here the variable $x$ is the macroscopic parameter such that $x \in \Omega$.$\left(\nabla_x^s u + \nabla_y^s v\right)$ denotes the symmetric gradient of the periodic displacement ansatz $u_\eta$, under first order asymptotic expansion [46]. Two french mechanical Engineers Moulinec and Suquet [38] developed a Fast Fourier Transform (FFT) based algorithm that can solve the cell problem directly on the images with reasonable accuracy and speed, thus replacing the FE (Finite element) based numerical methods. This work uses a FFT based solver FeelMath [13] to perform linear and nonlinear computation.

## 2.3 Plasticity and creep coupled model

In Section 2.2 one touches briefly about constitutive equations with a small overview of the process involved in material modelling. The material of interest here is PBT GF30. In the context of SFRT there are two phases each exibiting their own material behaviour. The glass fibers are considered to be linear elastic in nature since they are much stiffer than the matrix. The properties for the glass fibers are chosen from Doghri et al.[7]. The properties chosen are shown in table 2.1. The matrix phase undergoes plastic strain in quasi-static time scale and creep strain in large time scales. So to model the matrix phase there is a requirement of a fully coupled plasticity and creep law.

### 2.3.1 Kinematics

To completely describe the continuum mechanical behaviour of the creep-plasticity model the material model readily developed at Bosch was used. The modelling framework uses a J2 plasticity model involving an isotropic exponential linear hardening [20]. The creep has been developed with strain hardening exponential law for the primary stage and stress power [23] law for the secondary stage. During the initial ramp-up load for 8.5s, the coupled plasticity-creep law used in this work allows for inelastic material strains using the plasticity law and for long term loading uses a creep law. The symmetric small strain tensor $\varepsilon = \nabla_s \boldsymbol{u}$ (Effective strain) can be broken down into three different strains as shown in equation 2.9,

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^{\mathrm{e}} + \boldsymbol{\varepsilon}^{\mathrm{p}} + \boldsymbol{\varepsilon}^{\mathrm{c}}. \tag{2.9}$$

Here $\boldsymbol{\varepsilon}^{\mathrm{e}}$ is linear elastic strain which accounts for the strain that is reversible, $\boldsymbol{\varepsilon}^{\mathrm{p}}$ and $\boldsymbol{\varepsilon}^{\mathrm{c}}$ accounts for the inelastic strains that are not reversible and allows for permanent deformation of the material sample. The equation 2.10 shows the set of independent variables that can be used to define the constitive equations, here $\alpha$ denotes the isotropic hardening variable,

$$\boldsymbol{c} := \{\boldsymbol{\varepsilon}, \alpha, \boldsymbol{\varepsilon}^{\mathrm{p}}, \boldsymbol{\varepsilon}^{\mathrm{c}}\}. \tag{2.10}$$

### 2.3.2 Free energy function definition, driving forces and stresses

The free energy function gives a definition of the energy storage behaviour of the material and are dependent on the constitutive states defined in Equation 2.10. These equations also involve material constants like the initial yield stress $y_0$, the saturation yield stress $y_\infty$, the linear hardening modulus $h$, the hardening parameter $\omega$ and the elastic constants. Choosing the right variety of functions in combination with the material constants, the free energy equation is designed to model the behaviour of the material in focus. The free energy function is formulated in the context of Generalized Standard Material ($GSM$) which means that it imposes convexity of the function such that minimization of potential (convex function) always coverges to a unique solution [16]. The coupled plasticity and creep model used in this work has the free energy function as shown in equation 2.11,

$$\begin{aligned} \psi(\boldsymbol{c}) = \; & \frac{1}{2}\kappa\, tr^2[\boldsymbol{\varepsilon}] + \mu \left\| dev[\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{p}} - \boldsymbol{\varepsilon}^{\mathrm{c}}] \right\|^2 \\ & + \frac{1}{2} h\alpha^2 + (y_\infty - y_0) \cdot \left[ \alpha + \frac{1}{\omega}(exp[-\omega\alpha] - 1) \right]. \end{aligned} \tag{2.11}$$

The Youngs modulus E and Poisson‘s ratio $\nu$ relates to the elastic constants the compression modulus $\kappa$ and the shear modulus $\mu$ using the equation 2.3.2.

$$\kappa = \frac{E}{3\,(1-2\nu)} \quad \text{and} \quad \mu = \frac{E}{2\,(1+\nu)}. \tag{2.12}$$

The stresses and driving forces as a result of Coleman's exploitation [22],

$$\begin{aligned} \boldsymbol{\sigma} &= \partial_{\varepsilon}\psi &&= \kappa\, tr[\boldsymbol{\varepsilon}]\, 1 + 2\mu\, dev[\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{p}} - \boldsymbol{\varepsilon}^{\mathrm{c}}], \\ \beta^{\mathrm{p}} &= \partial_{\alpha}\psi &&= -h\alpha - (y_{\infty} - y_0)(1 - exp[-\omega\alpha]), \\ \boldsymbol{B}^{\mathrm{p}} &= \partial_{\varepsilon^{\mathrm{p}}}\psi &&= 2\mu\, dev[\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{p}} - \boldsymbol{\varepsilon}^{\mathrm{c}}], \\ \boldsymbol{B}^{\mathrm{c}} &= \partial_{\varepsilon^{\mathrm{c}}}\psi &&= 2\mu\, dev[\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{\mathrm{p}} - \boldsymbol{\varepsilon}^{\mathrm{c}}]. \end{aligned} \tag{2.13}$$

### 2.3.3 Dissipation function

This section lists down all the equations that are involved in this framework that defines the flowrule and dissipation. The section 2.3.1 briefly discussed about strains that can not be recovered. The ireversibility is defined in the context of a dissipation function and can be written as a creep and plastic part as shown in equation 2.14

$$\Phi(\dot{\alpha}, \dot{\boldsymbol{\varepsilon}}^{\mathrm{p}}, \dot{\boldsymbol{\varepsilon}}^{\mathrm{c}}; \alpha, \boldsymbol{\varepsilon}^{\mathrm{p}}, \boldsymbol{\varepsilon}^{\mathrm{c}}) = \Phi(\dot{\alpha}, \dot{\boldsymbol{\varepsilon}}^{\mathrm{p}}; \alpha, \boldsymbol{\varepsilon}^{\mathrm{p}}) + \Phi(\dot{\boldsymbol{\varepsilon}}^{\mathrm{c}}; \boldsymbol{\varepsilon}^{\mathrm{c}}). \tag{2.14}$$

In order to bound the yield or enforce the yield limit and set a condition for plasticity to start, a yield creterion function is chosen as shown in equation 2.15.

$$\phi(\boldsymbol{B}^{\mathrm{p}}, \beta^{\mathrm{p}}) = \|dev[\boldsymbol{B}^{\mathrm{p}}]\| - \sqrt{2/3}(y_0 - \beta). \tag{2.15}$$

Based on evolution equation defined in 2.13 and dissipation function for plasticity, the plastic strain rate can be defined as in equation 2.16,

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{p}} = \frac{1}{\eta}\langle\phi(\boldsymbol{B}^{\mathrm{p}}, \beta^{\mathrm{p}})\rangle_{+}\boldsymbol{n}. \tag{2.16}$$

Where $\boldsymbol{n}$ denotes flow direction is given by Equation 2.17,

$$\boldsymbol{n} := \frac{dev[\boldsymbol{\sigma}]}{\|dev[\boldsymbol{\sigma}]\|}. \tag{2.17}$$

And also the hardening variable is given by the Equation 2.18,

$$\dot{\alpha} = \frac{1}{\eta}\langle\phi(\boldsymbol{B}^{\mathrm{p}}, \beta^{\mathrm{p}})\rangle_{+}\sqrt{2/3}. \tag{2.18}$$

Considering a visco-elastoplastic setting the the dissipation function for plasticity is given by equation 2.19 [20],

$$\Phi(\dot{\alpha}, \dot{\boldsymbol{\varepsilon}}^{\mathrm{p}}; \alpha, \boldsymbol{\varepsilon}^{\mathrm{p}}) = \sup_{\boldsymbol{B}^{\mathrm{p}}, \beta^{\mathrm{p}}} \left[ \boldsymbol{B}^{\mathrm{p}} : \dot{\boldsymbol{\varepsilon}}^{\mathrm{p}} + \beta^{\mathrm{p}} \dot{\alpha} - \frac{1}{2\eta} \langle \phi(\boldsymbol{B}^{\mathrm{p}}, \beta^{\mathrm{p}}) \rangle_{+}^{2} \right]. \tag{2.19}$$

Similarly the dissiption function for creep can be given by the equation 2.20 showing the chosen power law and the linear term [15],

$$\Phi(\dot{\boldsymbol{\varepsilon}}^{\mathrm{c}}; \boldsymbol{\varepsilon}^{\mathrm{c}}) = \sup_{\boldsymbol{B}^{\mathrm{c}}} \left[ \boldsymbol{B}^{\mathrm{c}} : \dot{\boldsymbol{\varepsilon}}^{\mathrm{c}} - (1 + C\, exp[-\varepsilon^{\mathrm{c}}/k]) \left( \frac{(A_1\sqrt{3/2}\|dev[\boldsymbol{B}^{\mathrm{c}}]\|)^{n+1}}{(n+1)A_1\sqrt{3/2}} + \frac{(A_2\sqrt{3/2}\|dev[\boldsymbol{B}^{\mathrm{c}}]\|)^{2}}{2A_2\sqrt{3/2}} \right) \right]. \tag{2.20}$$

In equation 2.20 one has constants $C$ and $k$, $A_1$ and $A_2$ are the prefactors, $n$ is the creep exponent. Again from the dissipation function and the driving forces mentioned in equation 2.13 the creep strain rate is given by equation 2.21,

$$\dot{\boldsymbol{\varepsilon}}^{\mathrm{c}} = [1 + C\, exp[-\varepsilon^{\mathrm{c}}/k]] \left[ (A_1\sqrt{3/2}\|dev[\boldsymbol{B}^{\mathrm{c}}]\|)^{n} + (A_2\sqrt{3/2}\|dev[\boldsymbol{B}^{\mathrm{c}}]\|) \right] \boldsymbol{n}. \tag{2.21}$$

The flow rules mentioned by equations 2.16, 2.21, 2.18 under time based discrete integration scheme are solved using the classical prediction correction algorithm with return mapping. The algorithm used here was implemented in User Defined Material Subroutine (UMAT) in Abaqus [49]. The same can be used with FFT based solver FeelMath [13].

### 2.3.4 Parameter identification

In the previous section one sees a lot of parameters or constants that were part of the equations. So settings these parameters are an important part of modelling itself. One of the way this can be done with the coupled creep plasticity model is through experiments. For the model defined at Bosch, the matrix material sample was subjected to monotonic uniaxial tensile loading and an experimental curve is obtained. Similar experiments are done in a simulation with the derived model with randomly chosen constants. An optimization is run in Optislang [54] iteratively until a proper set of parameters is obtained for the creep-plasticity coupled model. Optislang uses the Adaptive Metamodel of Optimal Prognosis (AMOP) [53] algorithm for optimization and Latin Hypercube Sampling [34] for parameter sampling. The results of the final parameters chosen for plasicity model can be seen in table 2.1. For creep, parameter identification is done on similar lines wherein experiments at three different loading scenarios are considered and similar optimization run is considered keeping the plasticity parameters constant. The results of the entire process is summarized in table 2.1.

| | | | | |
|---|---|---|---|---|
| Matrix | | | | |
| Elastic | $E = 2399.3MPa$ | $\nu = 0.4$ | | |
| Plastic | $h = 346.9MPa$ | $\omega = 384.9$ | $y_0 = 20.9MPa$ | $y_\infty = 51.9MPa$ |
| Creep | $A_1 = 0.014MPa^{-1}$ | $n = 32.4$ | $A_2 = 1.5 \times 10^{-13}MPa^{-1}$ | $C = 1870.1$ |
| | $k = 0.016$ | | | |
| Fibers | | | | |
| Elastic | $E = 72000MPa$ | $\nu = 0.22$ | | |

**Table 2.1:** Material parameters for PBT GF30.

## 2.4 Data driven methods and hybrid modelling

"Modelling is one of the centre pieces that led to the scientific revolution"-Noah et al. [17]. A scientist performs experiments. Makes observation about the experiments. Uses the language of mathematics to connect those observations and finally frames theses relations into a mathematical theory to obtain model that describes the phenomena, see Noah [17]. This allows a designer to just use these models for different input design variable data and obtain a calculation result using a model in an algorithm. The coursework on "Machine Learning Methods in Mechanics" by Mielke and Ricken [35] gives an intuitive pictorial description of the classical approach shown in figure 2.9.

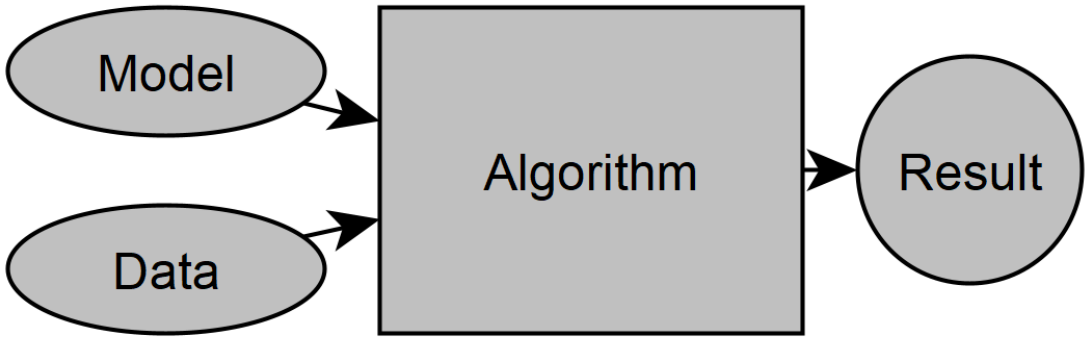



**Figure 2.9:** Classical Approach[35]

Usually models are built under suitable assumptions and in many cases good models are not available. So an alternate approach would be to provide the data and solution to the algorithm and let it learn the underlying model. This main idea of learning is what defines the field of machine learning. As stated by Mitchell et al. [36] "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by

P, improves with experience E". The figure 2.10 shows the pictorial depiction of the basic idea behind machine learning.

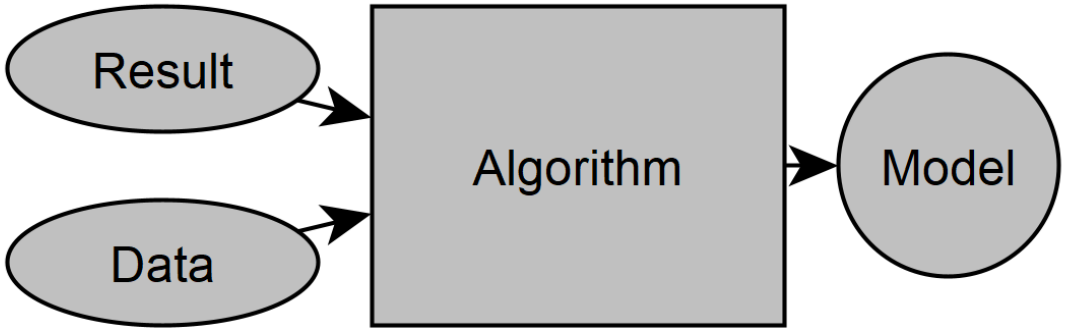



**Figure 2.10:** Machine Learning Approach [35]

In most cases the resulting model is not obtained by fundamental understanding of the physical phenomena at hand. This difference makes modelling approach very data efficient, casually explainable and dependable in its predictions. In the same essence, machine learning discovers hidden relations in the data based on statistical principles and helps us find models that were previously impossible to find.

### 2.4.1 Hybrid modelling and its need

The section 2.2 explained about multiscale modelling and homogenization. One of the numerical method proposed for solving the cell problem is FE-FFT method. For the nonlinear scenario of loading even this method becomes really slow despite the advancement in computational power and parallelization. One of the solution is to convert the cell problem into a parametric partial differential equation and solve it using model order reduction techniques. Gajek et al [10] discusses a few techniques that used till now like the Transformation Field Analysis (TFA) introduced by Dvorak et al. [8], self-consistent clustering Analysis introduced by Liu and co-authors [26] and $FE_{2R}$ reduced order developed by Fritzen and Hodapp [9]. Most of these method approximate the solution to the partial differntial equation and few of these are found to have very slow convergence. An alternative to such an approach is to approximate the properties that result from homogenization directly by operating in an higher dimensional domain. The main advantages of such an approach is that they are faster. The first set of methods that seldom comes to mind is ANN (artificial neural networks). Gajek et al. [10] discusses few of the successful implementations of using ANN to approximate the effective elastic energy or the stress strain relationships. There are also more sophisticated examples of using RNN (recurrent neural networks) that help model the history variables or a combined artificial neural network and switching

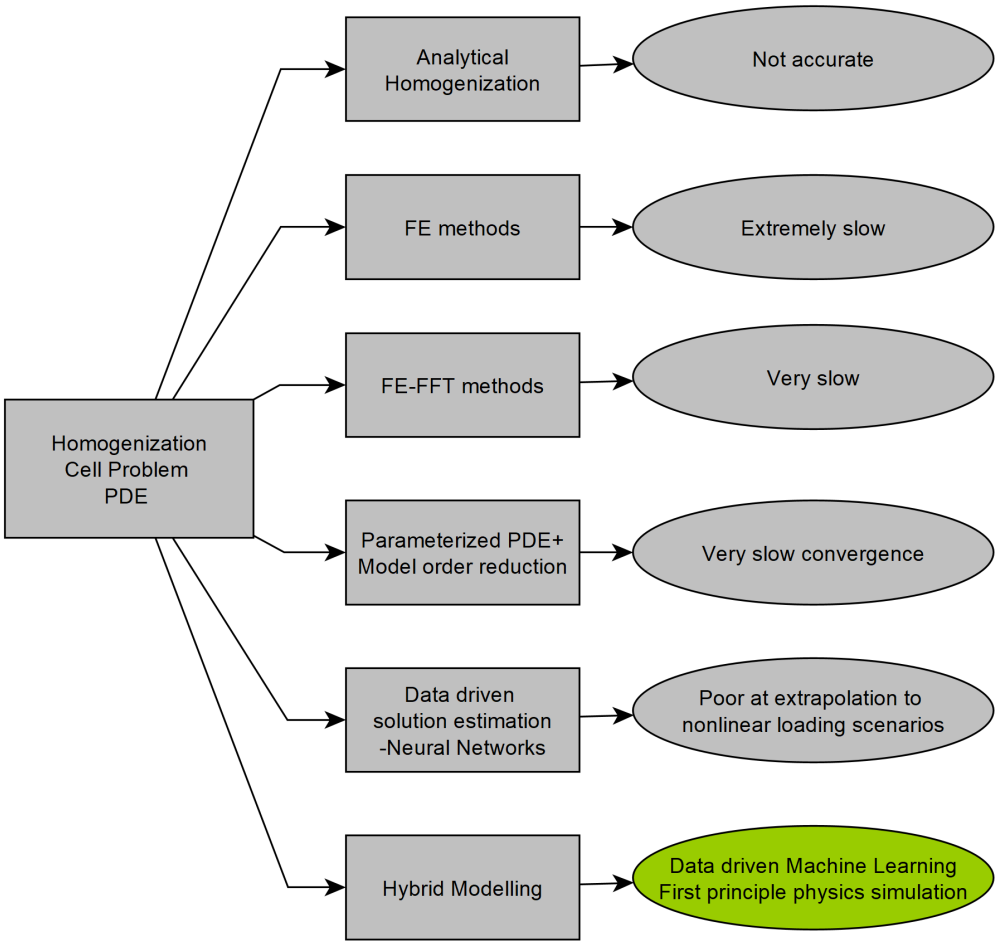



**Figure 2.11:** Approaches to perform Homogenization

reduced order model. One of the drawbacks of using artificial neural networks and associated techniques is that they are great at finding hidden relationship in the data and interpolation, but when it comes to predicting a data point in domains far away from its training data they perform poorly. Adding to this is their lack of accounting inherent physics and thermodynamic consistency. It is in these scenarios that the concept of hybrid modelling is developed where the concurrent modelling approaches derived via first principles help conserve the physics of the problem to be solved and the data driven machine learning methods help explore hidden relationship in the data allowing for fast simulation methods. The figure 2.11 summarizes the variety of approaches.

## 2.5 Deep material networks

This section presents a hybrid data driven surrogate model called deep material network first proposed by Liu and co-authors [27]. The framework is divided into online phase and the offline phase following the work of Gajek et al. [11] and Liu and Wu et al. [29]. Based on the offline phase there is further description of direct deep material network and deep material network with FOT interpolation.

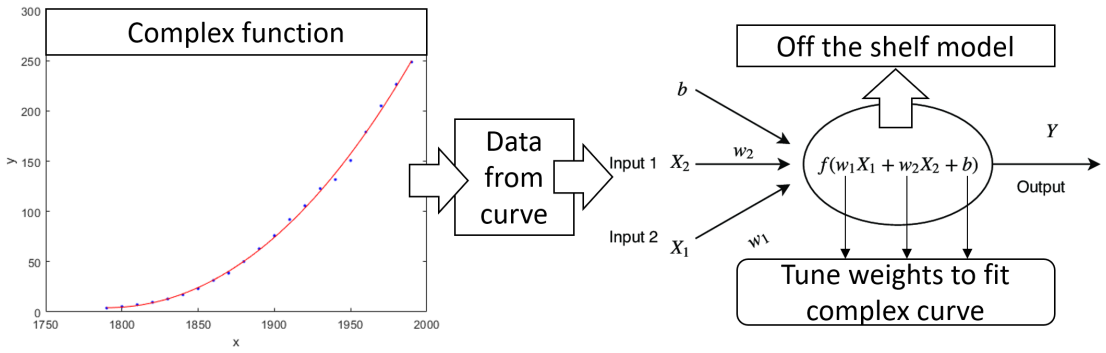
### 2.5.1 Offline training-direct deep material network

The DMN seeks to replace expensive full field simulations with a surrogate. As explained in section 2.4, machine learning is good at finding hidden relationship in the data and this aspect has been put to use in the offline phase. Here the DMN represents a complex microstructure as a two phase laminate arrangement as the building block. Phrasing this in the context of a regression problem, consider a complex function as shown in figure 2.12a. Any such complex function can be fit using an off-the-shelf multiplication addition function MAF and tuning the weights to fit the original complex function. One understands the DMN in the same context as shown in figure 2.12b, where a complex microstructure can be replaced with a series of laminates and in this way one tunes the quantity of material and arrangement of these laminates to fit the behaviour of the original complex microstucture. The homogenization of the two-phase laminates can be solved analytically with just a normal $\boldsymbol{n}$ which is a parameter indicating the direction of the laminates and volume fractions of the laminates $c_1$ and $c_2$.

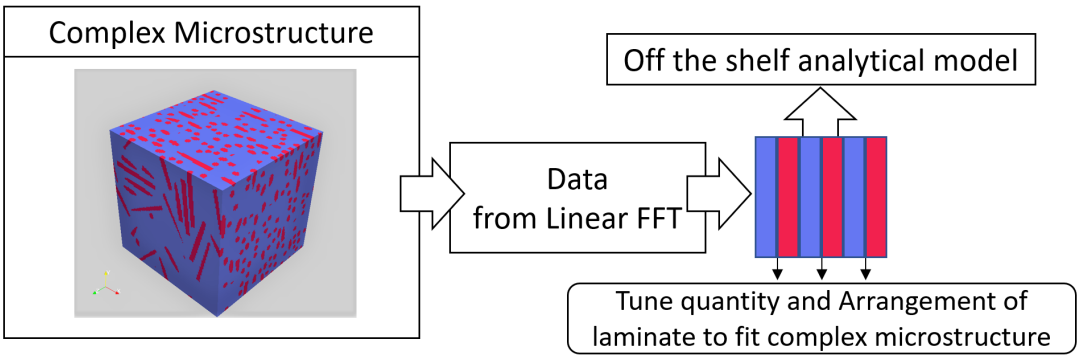
Using this concept Gajek et al. [11] develops the direct deep material network with a basic building block of a binary tree of laminate where output of a node is the homogenized stiffness $\bar{\mathbb{C}}$ under the homogenization function $\mathcal{DMN}_Y$ for a microstructure $Y$ as shown in equation 2.22 for given stiffnesses $\mathbb{C}_1$ and $\mathbb{C}_2$. The basic building block can be seen in figure 2.13b.

$$\bar{\mathbb{C}} = \mathcal{DMN}_Y(\mathbb{C}_1, \mathbb{C}_2) \tag{2.22}$$

Each of these individual block or node is connected in a binary tree structure with K-layers with each layer having index $k = 1, \ldots, K$. This architecture also fixes the number of nodes to $2^k - 1$ in the DMN with index $i = 1, \ldots, 2^k - 1$. Starting from the bottom nodes, each node gives an effective stiffness as shown in equation 2.23.

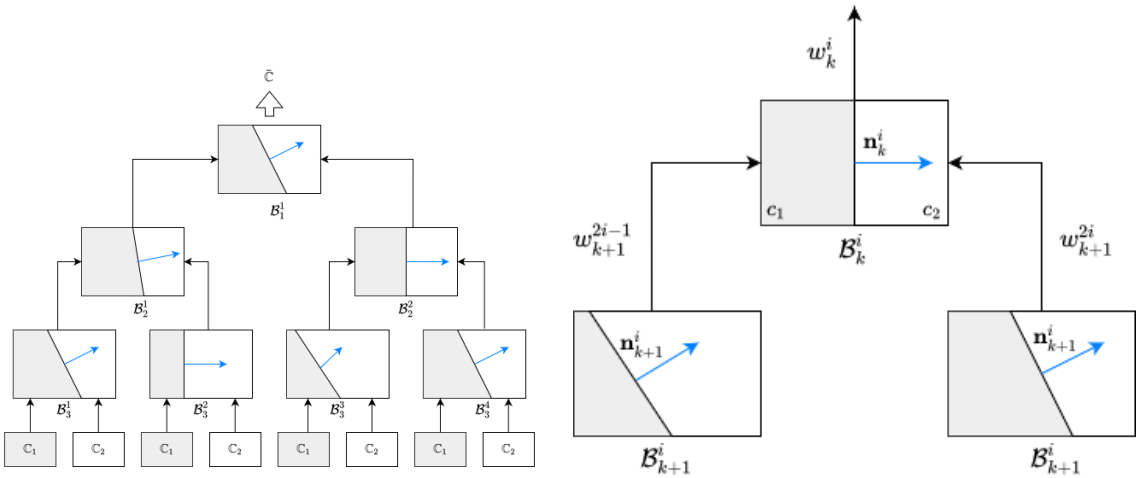



**(a)** Fitting a complex function with off the shelf regression model




**(b)** Fitting a complex microstructure using laminates and solving it using analytical equations

**Figure 2.12:** The figure shows an intuitive understanding of the offline phase




**(a)** A three layer binary tree of laminates

**(b)** Basic Building Block of DMN-A binary tree structure

**Figure 2.13:** The figure shows a two Phase Deep Material Network

$$\mathbb{C}_k^i = \mathcal{B}_k^i(\mathbb{C}_{k+1}^{2i-1}, \mathbb{C}_{k+1}^{2i}), \quad k = 1, \ldots, K, \; i = 1, \ldots, 2^{k-1}. \tag{2.23}$$

The stiffness is obtained by analytically solving the Milton equation as shown in equation 2.24,

$$\begin{gathered}\left(\mathbb{P}\left(\boldsymbol{n}_k^i\right) + \alpha\left[\mathbb{C}_k^i - \alpha\mathbb{I}_s\right]^{-1}\right)^{-1} = \\ c_1\left(\mathbb{P}\left(\boldsymbol{n}_k^i\right) + \alpha\left[\mathbb{C}_{k+1}^{2i-1} - \alpha\mathbb{I}_s\right]^{-1}\right)^{-1} + c_2\left(\mathbb{P}\left(\boldsymbol{n}_k^i\right) + \alpha\left[\mathbb{C}_{k+1}^{2i} - \alpha\mathbb{I}_s\right]^{-1}\right)^{-1}.\end{gathered} \tag{2.24}$$

As one traverses upwards, a more complex laminate is formed concluding at the single node at the top. The figure 2.13a shows a three layer DMN that shows the structure of a typical architecture.

The inputs to the direct deep material network, the material stiffness matrices of individual phases are assigned with the rule shown in equation 2.25

$$\mathbb{C}_{K+1}^i = \begin{cases} \mathbb{C}_1, & i \text{ odd}, \\ \mathbb{C}_2, & i \text{ even}. \end{cases} \tag{2.25}$$

Coming to the neural network parameters that are learnt, each input in the tree is parameterized by weights which is related to the volume fraction of the material chosen. The parameterization of the volume fraction using weights was proposed by

Liu et al. [28]. When volume fraction goes to zero an entire subtree is removed as a result of parameterization of neural network with volume fraction. So for given weights $w_{k+1}^{2i-1}$ and $w_{k+1}^{2i}$ in layer K+1, the weights are used for parameterization and are additively propagated as one goes up the tree as shown in equation 2.26 where $w_k^i$ is weight in layer K

$$w_k^i = w_{k+1}^{2i-1} + w_{k+1}^{2i}. \tag{2.26}$$

The volume fraction at each input is calculated as a normalized quantity of the weight parameters as shown in equation 2.6

$$c_1 = \frac{w_{k+1}^{2i-1}}{w_{k+1}^{2i-1} + w_{k+1}^{2i}} \quad \text{and} \quad c_1 + c_2 = 1. \tag{2.27}$$

Thus the direct deep material network can be defined by parameter weights $w_{K+1}^i$, which are defined at each input of a node and parameter $\boldsymbol{n}$ defined at every node. These parameters are learned using machine learning forward propagation and backward propagation of error from linear elastic training data. The machine learning problem is explained in detail in the further sections. Here weights $w_{K+1}^i$ are directly learnt and the direction of laminate $\boldsymbol{n}$ is learnt as a parameter set of three angles of rotation for a unit vector. So effectively a set of weights representing the quantity of material in the laminates and direction of those laminates $\boldsymbol{n}$ gives an equivalent laminate microstructure representation of the original complex microstructure. The word equivalent here is defined as the equivalence of mechanical response under similar loading conditions.

### 2.5.2 Offline training- deep material network with interpolation

The draw backs of the direct deep material network is the identified laminates ($w_{K+1}^i$) and their arrangement ($\boldsymbol{n}$) are closely tied to the complex microstructure it has been trained on. It is impractical to consider such an approach at the component level simulation where every part of the component has a new microstructure configuration. So there is a need for finding a way to allow the DMN to interpolate between microstructure configurations.

The word "microstructure configuration" is understood in reference to section 2.1.2. This section 2.1.2 introduces the FOT triangle, see figure 2.4. A microstructure configuration is identified by eigen value parameters $\lambda_1$, $\lambda_2$ of the fiber orientation tensor. Effectively there can be infinitely many microstructures with same $\lambda_1$, $\lambda_2$ but all of them must have the same mechanical response. So the DMN must be able to predict all possible mechanical responses over all the possible microstructure configurations identified on the FOT triangle 2.4a.

Keeping the goal in mind, one aims to parameterize the DMN with $\lambda_1$, $\lambda_2$ over the entire FOT triangle such that $\lambda_1$, $\lambda_2$ comes as an additional input along with the material stiffness matrix $\mathbb{C}_1$ and $\mathbb{C}_2$. As explained in the previous section 2.5.2 the weights are closely tied to the volume fractions of the individual phases. So for fixed volume fraction the learned weights are independent of the fiber orientation. Based on this Gajek et al. [10] suggests parameterizing the direction of laminates with point $\lambda_1$, $\lambda_2$. First one parameterizes the direction of lamination $\boldsymbol{n}_k^i$ with spherical coordinates $\boldsymbol{n}_k^i \in \boldsymbol{S^2}$. The equation 2.28 shows the relation between $\boldsymbol{n}_k^i$ and the angles. The angles $\alpha_k^i \in [0, \pi]$ and $\beta_k^i \in [0, 2\pi]$,

$$\boldsymbol{n}_k^i = \begin{bmatrix} \sin\left(\alpha_k^i\right)\cos\left(\beta_k^i\right) \\ \sin\left(\alpha_k^i\right)\sin\left(\beta_k^i\right) \\ \cos\left(\alpha_k^i\right). \end{bmatrix} \tag{2.28}$$

The angles $\alpha_k^i$ and $\beta_k^i$ are further parameterized over the FOT triangle using a linear interpolation shape function like the one used in finite element formulation. There are higher order interpolation possible but this work restricts itself to linear interpolation based on Gajek et al. [10], which shows linear interpolation to be effective [10]. Table 2.2 shows the shape functions used in the current work. Here M denotes number of shape functions. For higher order interpolation more shape functions are necessary.

| | | | | |
|---|---|---|---|---|
| Linear | $M = 3$ | $\phi_1 = \varphi_1$ | $\phi_2 = \varphi_2$ | $\phi_3 = \varphi_3$ |

**Table 2.2:** Shape function used for interpolating the angles over the FOT triangle

The FOT triangle parameters $\lambda_1$, $\lambda_2$ are transformed to the shape parameters $\varphi_1, \varphi_2$ and $\varphi_3$ under the transformation in equation 2.29,

$$\begin{bmatrix} 1 & 1/3 & 1/2 \\ 0 & 1/3 & 1/2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ 1 \end{bmatrix}. \tag{2.29}$$

The referred shape functions are expressed as the angles using equation 2.30,

$$\alpha_k^i\left(\lambda_1, \lambda_2\right) = \mathbf{p}_k^{iT}\boldsymbol{\phi}\left(\lambda_1, \lambda_2\right) \quad \text{and} \quad \beta_k^i\left(\lambda_1, \lambda_2\right) = \mathbf{q}_k^{iT}\boldsymbol{\phi}\left(\lambda_1, \lambda_2\right). \tag{2.30}$$

This increases the number of learned parameters for every node along with the learned weights at each input. In the case of linear interpolation one has interpolation parameters $\mathbf{p} = [p_1, p_2, p_3] \in \mathbb{R}_M$ and $\mathbf{q} = [q_1, q_2, q_3] \in \mathbb{R}_M$ used in the equation 2.30. The general procedure can be illustrated in figure 2.14.

Thus, the FOT triangle interpolation enters the machine learning model equations allowing the DMN to keep the same laminates ($w^i_{K+1}$) for fixed volume fraction and change the arrangement of these laminates ($\boldsymbol{n}$) using an interpolation function to represent any given configuration of microstructure. These simplified equivalent laminate microstructures are used in an numerical material routine to solve for mechanical response. These methods are explained in the online phase.

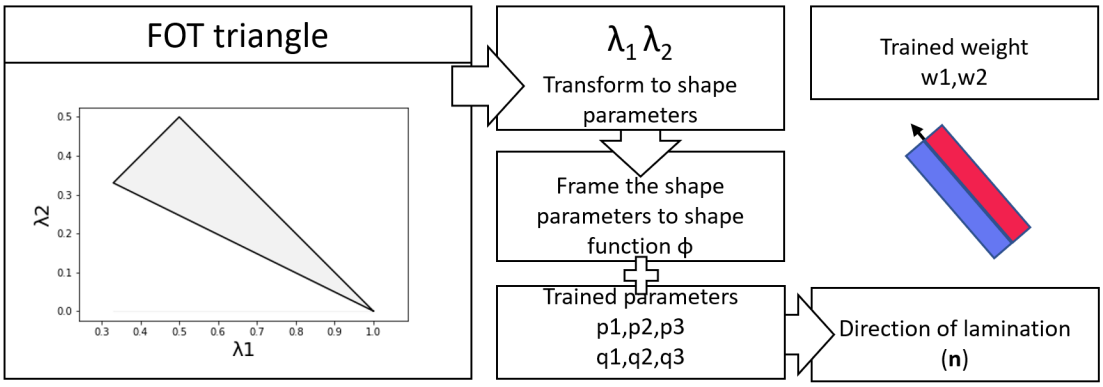



**Figure 2.14:** Approach to calculate the direction of lamination for a single 2 phase laminate of an interpolated DMN

### 2.5.3 Online evaluation

The online phase in this work has been implemented as an user-defined material subroutine in FFT and Abaqus. The material routine is written in FORTRAN and the approach is similar to Gajek et al. [11]. The approach uses a flattened representation of the entire DMN tree. The word flattened means that each input laminate additively contributes to the final stresses and strains in accordance with the trained weights, direction of laminates and chosen material model for each phase. The result is identified by a nonlinear solution method.

To understand the workflow one should consider a single building block. Starting with the calculation of the volume fraction of each laminate $c^1_1$ and $c^2_1$ using the weights obtained from the offline training, see equation 2.6. For the given $\lambda_1, \lambda_2$ from the FOT triangle, one calculates the direction of laminate for the particular building block as per the procedure mentioned in figure 2.14. With theses parameters at hand, one obtains the macroscopic strain as an input $\bar{\boldsymbol{\varepsilon}} = \nabla_s \bar{\boldsymbol{u}}$. Given these inputs, the end goal is to solve for the tangent stiffness and the stresses for each laminate. So one defines the laminate kinematics in terms of displacement jump vectors $\boldsymbol{a}^1_1$ as strains in both the phases as shown in equation 2.31

$$\begin{aligned} \boldsymbol{\varepsilon}^1_1(\bar{\boldsymbol{\varepsilon}}, \boldsymbol{a}^1_1) &= \bar{\boldsymbol{\varepsilon}} + c^2_1 \, \mathbb{N}^1_1 \cdot \boldsymbol{a}^1_1 \\ \boldsymbol{\varepsilon}^2_1(\bar{\boldsymbol{\varepsilon}}, \boldsymbol{a}^1_1) &= \bar{\boldsymbol{\varepsilon}} - c^1_1 \, \mathbb{N}^1_1 \cdot \boldsymbol{a}^1_1 . \end{aligned} \tag{2.31}$$

Here $\mathbb{N}_1^1$ is a third order tensor defined as shown in the equation 2.32 and depends on the direction of lamination

$$(\mathbb{N}_1^1)_{ijk} = \frac{1}{2}\left[(n_1^1)_i\delta_{jk} + (n_1^1)_j\delta_{ik}\right]. \tag{2.32}$$

Effectively the dot product with jump vector $\boldsymbol{a}_1^1$ reads as

$$\mathbb{N}_1^1 \cdot \boldsymbol{a}_1^1 = \mathbb{I}^{sym} : (\boldsymbol{n}_1^1 \otimes \boldsymbol{a}_1^1) \tag{2.33}$$

Setting up a variation problem for the free energy function under variational of the jump vector $\boldsymbol{a}$ one has

$$\bar{\psi}(\bar{\boldsymbol{\varepsilon}}) = \inf_{\boldsymbol{a}_1^1}\left[c_1^1\ \psi_1^1(\boldsymbol{\varepsilon}_1^1(\bar{\boldsymbol{\varepsilon}}, \boldsymbol{a}_1^1)) + c_1^2\ \psi_1^2(\boldsymbol{\varepsilon}_1^2(\bar{\boldsymbol{\varepsilon}}, \boldsymbol{a}_1^1))\right]. \tag{2.34}$$

Considering the chain rule, equation 2.34 reduces to,

$$\left[c_1^1\frac{\partial\psi_1^1}{\partial\boldsymbol{\varepsilon}_1^1} : \frac{\partial\boldsymbol{\varepsilon}_1^1}{\partial\boldsymbol{a}_1^1} + c_1^2\frac{\partial\psi_1^2}{\partial\varepsilon_1^2} : \frac{\partial\varepsilon_1^2}{\partial\boldsymbol{a}_1^1}\right] \cdot \delta\boldsymbol{a}_1^1 \equiv \left[c_1^1c_1^2\boldsymbol{\sigma}_1^1 : \mathbb{N}_1^1 - c_1^1c_1^2\boldsymbol{\sigma}_1^2 : \mathbb{N}_1^1\right] \cdot \delta\boldsymbol{a}_1^1 \overset{!}{=} 0. \tag{2.35}$$

One could conclude traction vectors as $\boldsymbol{t}_1^1 = \boldsymbol{t}_1^2$ from the above equation where $\boldsymbol{t}_1^1 = \boldsymbol{\sigma}_1^1 : \mathbb{N}_1^1$ and $\boldsymbol{t}_1^2 = \boldsymbol{\sigma}_1^1 : \mathbb{N}_1^2$ expressed at the interface. From the necessary condition mentioned in equation 2.35 one assigns an arbitrary jump vector $\boldsymbol{a}$ and solve this using a nonlinear solution method. Identification of this jump vector helps to evaluate the contribution of each laminate at the bottom layer.

Now that one understands the kinematic at a single building block, let us consider laminates of depth K, for each node in the tree one assigns an arbitrary jump vector $\boldsymbol{a}$,

$$\boldsymbol{a} = \boldsymbol{a}_k^i, \quad k = 1, \ldots, K,\ i = 1, \ldots, 2^{k-1}. \tag{2.36}$$

Then one gives a generalized representation of the necessary condition.

$$\bar{\psi}(\bar{\boldsymbol{\varepsilon}}) = \inf_{\boldsymbol{a}_k^i}\left[\sum_{I=1}^{2^K} w_K^I\ \psi_K^I(\boldsymbol{\varepsilon}_K^I)\right] \quad \text{for} \quad \begin{cases} k &= 1, ..., K \\ i &= 1, ..., 2^{k-1}. \end{cases} \tag{2.37}$$

Looking just at the bottom layer K,the strains and the free energy of the phases are denoted as

$$\boldsymbol{\varepsilon}_K^I = \boldsymbol{\varepsilon}_K^I(\bar{\boldsymbol{\varepsilon}}, \boldsymbol{a}_k^i) \tag{2.38}$$

$$\psi_K^I(\boldsymbol{\varepsilon}_K^I) = \begin{cases} \psi^1(\boldsymbol{\varepsilon}_K^I), & I \text{ odd}, \\ \psi^2(\boldsymbol{\varepsilon}_K^I), & I \text{ even}. \end{cases} \tag{2.39}$$

This way the strains are assigned in the same way using the jump vectors $\boldsymbol{a}$ as shown in equation 2.31. The stresses $\boldsymbol{\sigma}_K^I$ are given by,

$$\boldsymbol{\sigma}_K^I = \partial_{\boldsymbol{\varepsilon}_K^I} \psi_K^I. \tag{2.40}$$

Thus the necessary condition in equation 2.37 is rewritten in terms of stresses and jump vector as

$$\left[\sum_{I=1}^{2^K} w_K^I \; \partial_{\boldsymbol{\varepsilon}_K^I} \psi_K^I : \partial_{\boldsymbol{a}} \boldsymbol{\varepsilon}_K^I\right] \cdot \delta \boldsymbol{a} = 0. \tag{2.41}$$

So given a tree of depth K one needs to find the jump vector $\boldsymbol{a}_k^i$ arbitrarily chosen to satisfy the necessary conditions. Finding these vectors can be framed as a residuum solved using the Newton's method,

$$\boldsymbol{r}(\boldsymbol{a}) : \; = \sum_{I=1}^{2^K} w_K^I \; \partial_{\boldsymbol{\varepsilon}_K^I} \psi_K^I : \partial_{\boldsymbol{a}} \boldsymbol{\varepsilon}_K^I \stackrel{!}{=} 0. \tag{2.42}$$

Thus, a Newton method using a backtracking algorithm is implemented to avoid instabilities due to large gradients $\lambda \in (0, 1]$ . Residuum update is given as,

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - \lambda \left[\partial_{\boldsymbol{a}} \boldsymbol{r}\right]^{-1} \cdot \boldsymbol{r}. \tag{2.43}$$

The equation is solved until convergence is reached $\|\boldsymbol{r}\| \leqslant \texttt{tol}$. and the global algorithmic tangent is expressed as

$$D_{\bar{\boldsymbol{\varepsilon}}} \bar{\boldsymbol{\sigma}} = \partial_{\bar{\boldsymbol{\varepsilon}}} \bar{\boldsymbol{\sigma}} - \partial_{\boldsymbol{a}} \bar{\boldsymbol{\sigma}} : \left[\partial_{\boldsymbol{a}} \boldsymbol{r}\right]^{-1} : \partial_{\boldsymbol{r}} \bar{\boldsymbol{\sigma}}, \tag{2.44}$$

where one expresses $\partial_{\bar{\boldsymbol{\varepsilon}}} \bar{\boldsymbol{\sigma}}$ and $\partial_{\boldsymbol{a}} \bar{\boldsymbol{\sigma}}$ in terms of tangent at the individual phases $D_{\boldsymbol{\varepsilon}_K^I} \sigma_K^I$ as shown

$$\partial_{\bar{\boldsymbol{\varepsilon}}} \bar{\boldsymbol{\sigma}} = \sum_{I=1}^{2^K} w_K^I D_{\boldsymbol{\varepsilon}_K^I} \boldsymbol{\sigma}_K^I \quad \text{and} \quad \partial_{\boldsymbol{a}} \bar{\boldsymbol{\sigma}} = \sum_{I=1}^{2^K} w_K^I D_{\boldsymbol{\varepsilon}_K^I} \boldsymbol{\sigma}_K^I : \partial_{\boldsymbol{a}} \boldsymbol{\varepsilon}_K^I. \tag{2.45}$$

So the online phase reproduces any given boundary conditions for any given microstructure configuration using basic numerical solver like abaqus or FFT.

# 3 Methodology

## 3.1 NumProDeep-A Framework to identify Machine learning models

The idea of a framework in this context is to rollout a blueprint having a systematic workflow in which DMN models can be identified given a new material. The framework eases the user workload through automation and incorporates algorithms that aid in speeding-up decision making. The framework was named NumProDeep-numerical program for deep material network.

The figure 3.1 shows the overview of the steps involved during the model identification. Many of the processes are complex and produces a large quantity of data involving different file types. This makes managing the data very labour intensive and time inefficient. This mainly occurs because the workflow uses different calculation tools that have their own unique file formats.

One other facility is the availability of high performance cluster (HPC) within bosch. The advantage of the HPC is availability of high end computing required for model training and parallelization of computing jobs.

Considering all these the design requirements of the framework are listed:

- Automate management, documentation and tracking of data created at every stage of the model identification.
- Automate, integrate the repetitive tasks.
- Integrate different tools and facilitate exchange of information within different file formats at every stage of model identification.
- Parallelization of different data generation using object tracking and HPC jobs.
- Parallelization of Training and evaluation using object tracking in HPC.
- Create tools to support decision making in model identification.

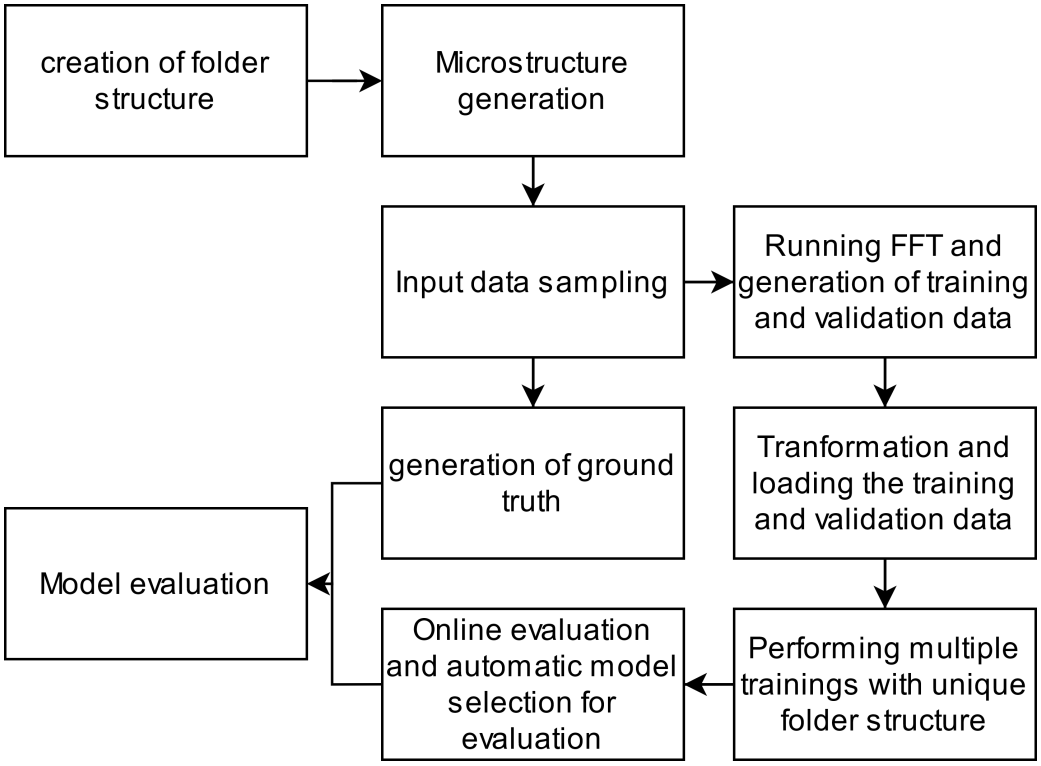
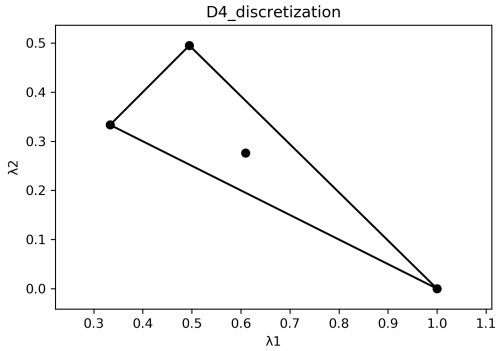



**Figure 3.1:** Overview of the steps involved in the model identification workflow

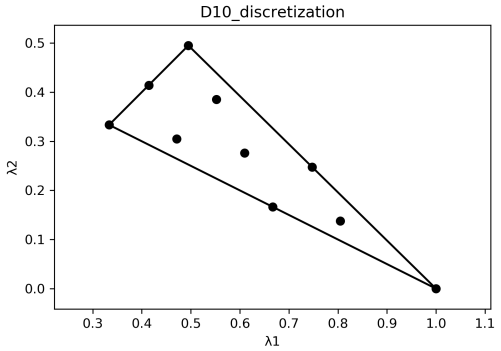
### 3.1.1 Discretization and FE software architecture

Each point on the FOT triangle signifies a microstructure and carries unique information related to the microstructure. When generating data for the training, sampling of points are done on the FOT triangle. Corresponding to each point, a microstrcuture is generated along with the necessity to run/track FFT simulations and maintain a separate folder structure. Getting inspiration from FE software architecture [50], the FOT triangle was assumed to be a finite element. The sampling of points gives the same analogy as meshing. It can be seen from figure 3.2, a higher sampling giving an analogy of a finer mesh and vicecersa. The framework was named NumProDeep because it was similar to the NumPro-lite[50] software from Institute for Structural Mechanics (IBB) at the University of Stuttgart.

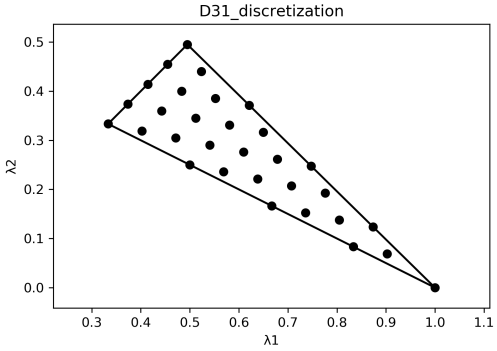
So a software platform was built using the traditional finite element framework initializing classes for element and node, all of them uniquely identified with objects. These objects can be used to track parameters and also allows for access and parallelization of these tasks without the need to run it sequentially. The figure 3.3 shows an overview of the different components involved in the framework. The entire framework is controlled with a central configuration file which the user uses to choose the parmeters used by the systems at different levels. The configuration parameters are divided into microstrcuture parameters, homogenization parameters, offline training parameters,




**(a)** D4 discretization




**(b)** D10 discretization




**(c)** D31 discretization

**Figure 3.2:** The figure shows different levels of sampling on the FOT triangle

optimizer parameters, learning rate modulation parameters, cost function parameters, ground truth generation parameters and online evaluation parameters.

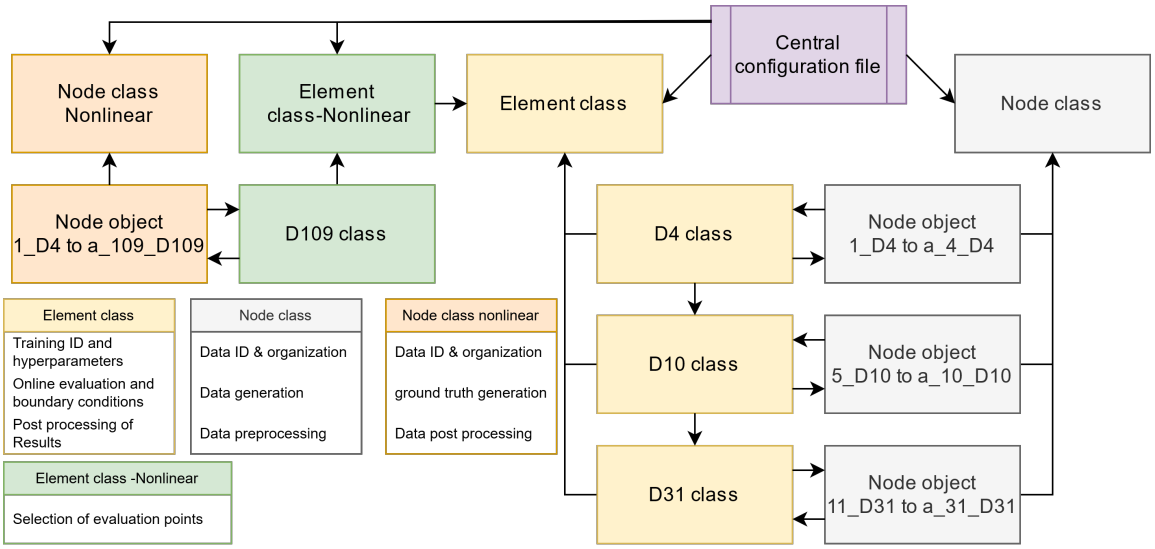



**Figure 3.3:** A basic class diagram showing the components of the NumProDeep framework

### 3.1.2 Element and bisection algorithm

The Element class is a parent class for different discretizations. The discretization D4, D10, D31 are child classes to Element class. It is built in such a way that initializing D31 automatically creates D10 and D4 so that data generation can happen at different levels. As an advantage there is re-intialization of objects such that data once created need not be created again. It also gives feature such that, if the data generation process stops in between it can be restarted from where it stopped. The need for a parent and child class is justified by a more organized tracking of parameters required for training. At a later point it can be seen that each discretization has a different input sampling and number of training data, which asks for different handling of folder structure. It also allows to track different training under different folder structure paralelly. The figure 3.4 shows a typical folder structure intricately managed by Element discretization class objects.

In order to perform discretization there has been a bisection algorithm developed which takes advantage of the Element class to create multiple bisected dummy triangles. For example an object of D4 has a single Element object and D10 has four such Element objects appended to it, see figure 3.5. The Element parent class is just a dummy class holding just the geometrical information bounding the element. Even though these object in the current stage hold only the geometrical information, these objects can be

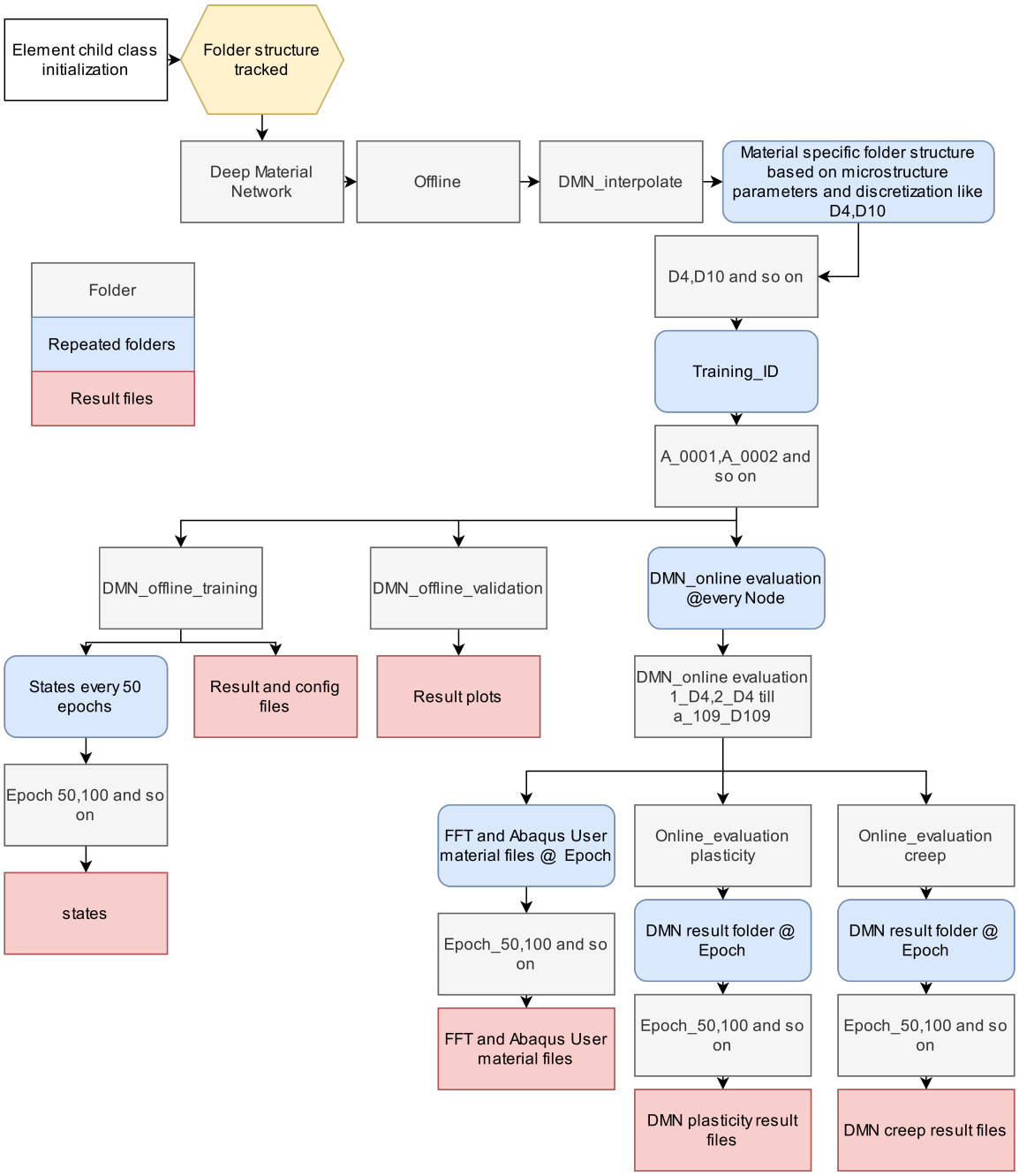



**Figure 3.4:** Folder structure tracked by Element discretization object

customized to hold error information or build mesh refinement to build sophisticated analytics for understanding the deep material networks or prediction methods. This was the reason for having a dummy parent class which holds general properties of a discretization and sophisticated child classes like D31 which hold properties required for tracking training/hyper parameters and their folder structures. Everytime a

discretization object is initialized, a folder structure for a training is created and this object is uniquely identified using a training-ID. This way many trainings can be parallelized using HPC and once an evaluation in the linear elastic training is validated, the same object can be reinitialized to perform the online validation based on decisions taken by the user.

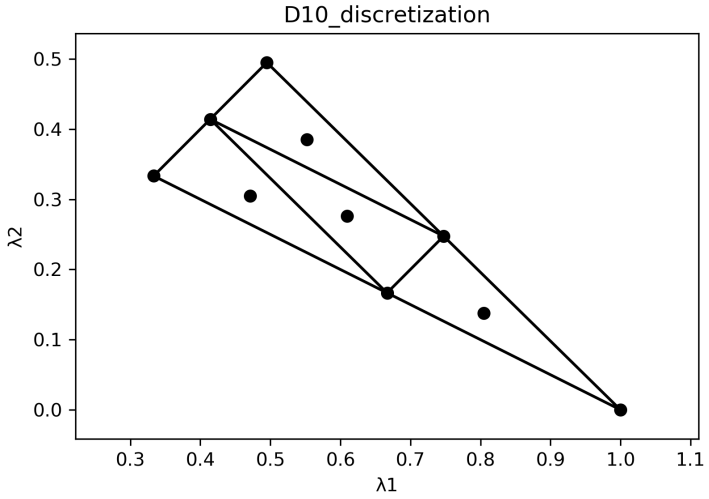



**Figure 3.5:** D10 discretization having four dummy Element objects

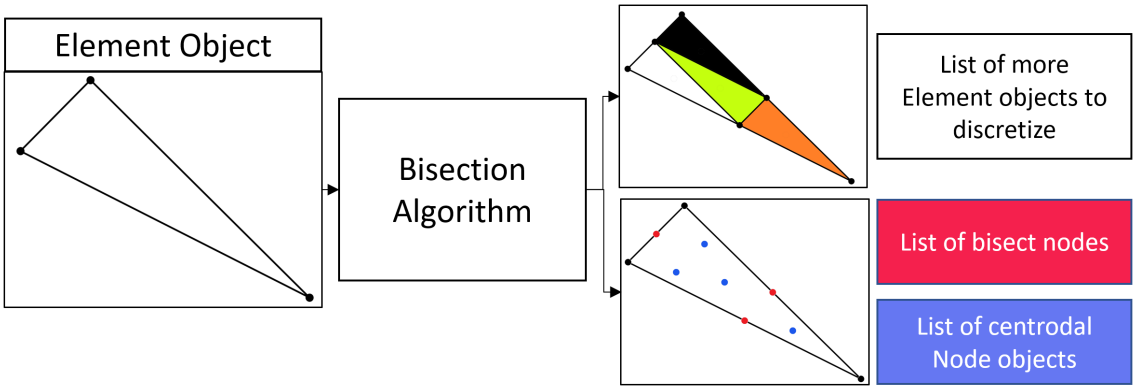



**Figure 3.6:** Bisection algorithm outputs

### 3.1.3 Node-linear

The Node class associates the Element object and creates multiple objects with unique identification based on the discretization chosen. As soon as an Element discretization is chosen an object of that discretization is created, corresponding number of Node

objects are intialized with unique folder structure identified by the respective node names. Unlike the element class and its children, the Node class objects are mainly built to track the data generation procedure from the FFT [13] or Abaqus [49]. The microstrcuture parameters including the FOT triangle eigen values are automatically assigned and calculated during the bisection algorithm stage. The figure 3.6 shows the outputs of the Bisection algorithm. The algorithm takes in Element object with only three objects at the vertex and creates more Nodes and Elements which can be further discretized to get a finer mesh. As a side note the algorithm can be modified to adapt the Element sizes in particular regions to make the node spacing finer or coarser. The figure 3.7 shows a typical folder structure intricately managed by Node class objects.

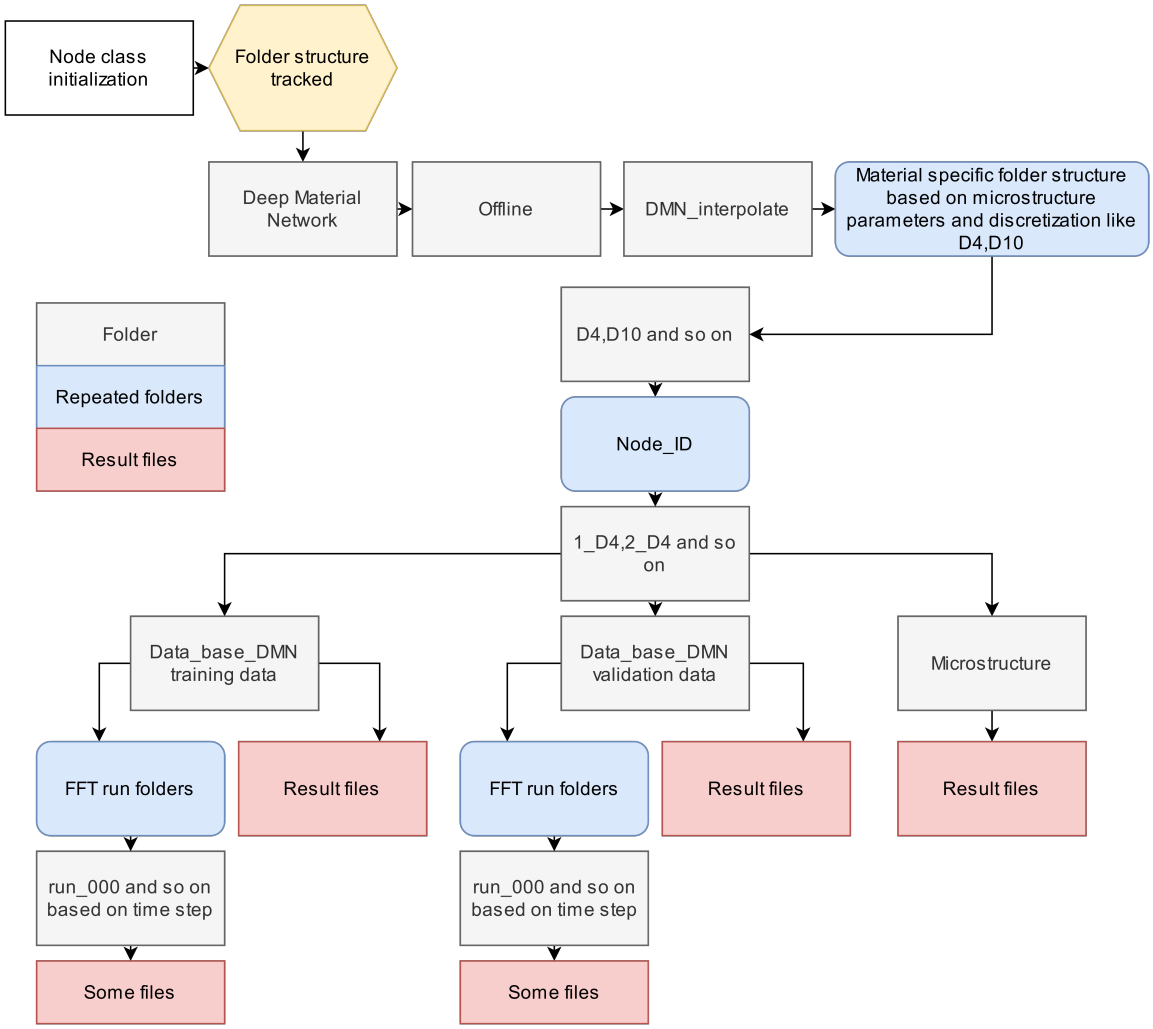



**Figure 3.7:** Folder structure tracked by Node class object

The Node naming convention can be seen in table 3.1. The figure 3.8 shows the naming convention for D10 discretization. For the finer discretization of D31, the

additionally added 21 nodes are named with Discretization-type= D31.

| | |
|---|---|
| corner nodes | $(Node-number)\_(Discretization-type)$ |
| bisection nodes | $(Node-number)\_(Discretization-type)$ |
| centroidal nodes | $a\_(Node-number)\_(Discretization-type)$ |

**Table 3.1:** Naming convention for Nodes

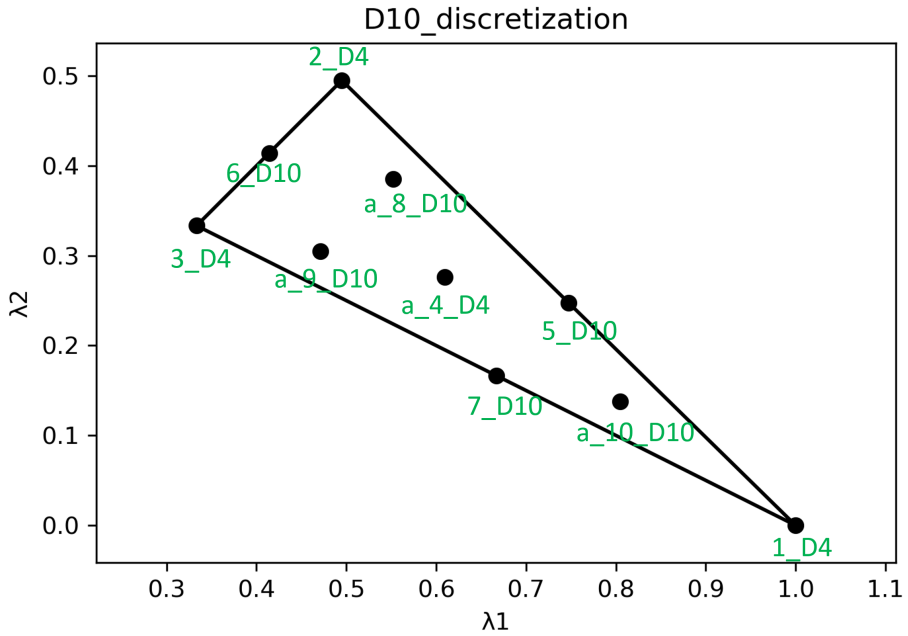



**Figure 3.8:** Node naming of D10 discretization

### 3.1.4 Node-nonlinear

The generation of ground truth is an important task in order to evaluate the performance of the identified model. This uses the same steps such as, generation of microstructure the same way we generate for linear FFT, application of the boundary conditions and settings for FFT FeelMath and tracking/managing folder structure. In the current study the DMN is evaluated on 109 different microstructure configurations. Generation of ground truth data is very time expensive because it is a nonlinear FFT simulation. This makes the parallelization and utilization of many licenses an essential pursuit. These design requirements are once again achieved by having a separate nonlinear Element class called “D109” and associated nonlinear Node class which exclusively tracks data from ground truth generation. An object from the D109 class is automatically associated with the Discretization Element object such that

any training that is run is provided with access to ground truth to run evaluation on selected set of points. The results are also evaluated and stored in convoluted folders structures allowing tracking of results for further aggregated post processing. Later it can also be seen that the DMN is not just evaluated at a single end state but throughout the training process, states of the DMN are stored and evaluated on all these 109 points to identify the best model for our application. The Node objects are paralleized to run on different FeelMath licences and their respective objects are tracked. In this work only three nodes have been parallelized. The achievable parallelization using the framework is limited only by the number of licenses. The figure 3.9 shows a typical folder structure intricately managed by Nonlinear Node class objects.

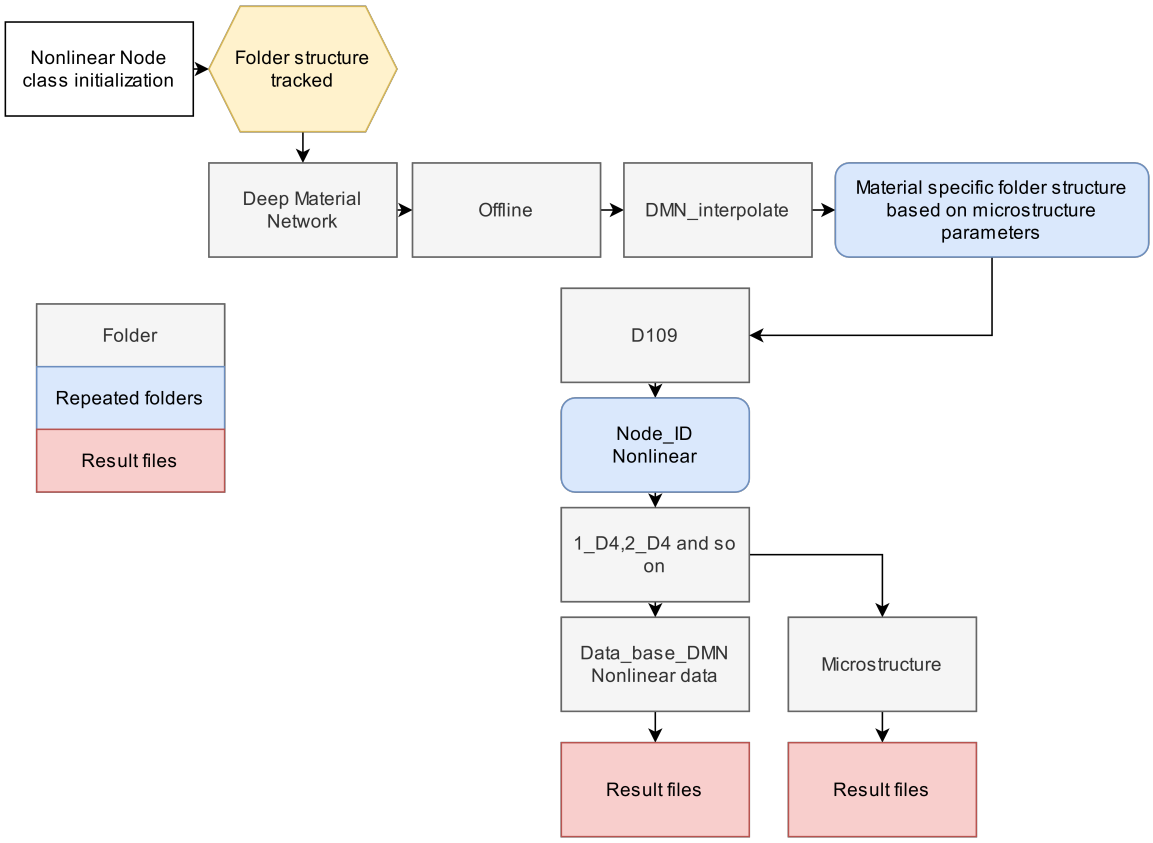



**Figure 3.9:** Folder structure tracked by Nonlinear Node class object

### 3.1.5 Microstructure parameters

The microstrcuture generation is done using an FiberGenerator [14] based on sequential addition and migration algorithm for generating Representative Volume Element

(RVE) proposed by Schneider [47]. The chosen parameters for the current work are given as

| Parameters | Values |
|---|---|
| Number of voxels | $128^3$ |
| Fiber diameter | 10 $\mu$-m |
| Fiber length | 100 $\mu$-m |
| Length of unit cell | 256 $\mu$-m |
| Minimum distance between fibers | 4 $\mu$-m |
| Volume fraction | 17.8 % |

**Table 3.2:** Chosen microstructure parameters.

### 3.1.6 Material sampling and sampling size

The current work uses the same methodology for material sampling as followed by Liu and Wu [29]. In this work only orthotropic stiffness tensors are used, so the number of independent parameters reduce to nine from 21 parameters for fully anisotropic stiffness tensor. For a two phase composite like SFRT two such orthotropic tensors $\mathbb{C}_1$, $\mathbb{C}_2$ are to be sampled. These 19 independent parameters are sampled from a quasi random sampling distribution methodology called Latin-hyper-cube-sampling (LHS) [21]. The components of the stiffness matrix are expressed by elastic moduli $E_{kk}$, Poisson's ratio $\nu_{kl}$ and shear moduli $G_{kl}$ where $k, l \in \{1, 2, 3\}$ and $k \neq l$.

The elastic modulus is given by the formula

$$E_{kk} = \xi \frac{\hat{E}_{kk}}{\left(\hat{E}_{11}\hat{E}_{22}\hat{E}_{33}\right)^{1/3}} \quad \text{with} \quad \hat{E}_{kk} \in \mathbb{U}[0.1, 10]. \tag{3.1}$$

Here the hat denotes random samples drawn from the LHS method and $\mathbb{U}$ denotes uniformly sampled distribution over the range specified. $\xi$ denotes the scaling factor which is given by

$$\xi^1 = 1, \quad \xi^2 = \hat{\xi}^2 \in \mathbb{U}[0.001, 1000]. \tag{3.2}$$

. The scaling factor for the second phase $\xi^2$ determines the phase contrast of the stiffness matrices.

The poissons ratio is selected to ensure positive definite stiffness tensor and calculated as

$$\nu_{kl} = \hat{\nu}_{kl}\sqrt{\frac{E_{ll}}{E_{kk}}} \quad \text{with} \quad \hat{\nu}_{kl} \in \mathbb{U}[0.0, 0.5]. \tag{3.3}$$

And the shear modulus is given by the equation

$$G_{kl} = \hat{G}_{kl}\sqrt{E_{kk}E_{ll}} \quad \text{with} \quad \hat{G}_{kl} \in \mathbb{U}[0.25, 0.50]. \tag{3.4}$$

Every discretization has its own number of samples $N_s$. The table 3.3 shows the sampling size for each discretization chosen. The chosen samples are equally assigned to each Node alternately and using the procedure described, input stiffness $\mathbb{C}_1^s$ and $\mathbb{C}_2^s$ are calculated. Along with the input stiffness there are also the eigenvalue parameters of FOT $\lambda_1$,$\lambda_2$, which are associated with the microstructure. The eigenvalues were calculated using the bisection algorithm previously discussed in section 3.1.2. The current work uses FFT based numerical homogenization technique using commertial solver called FeelMath [13] for generating linear elastic training data $\bar{\mathbb{C}}_{FFT}^s$. In total the training data is comprised of $\{\mathbb{C}_1^s, \mathbb{C}_2^s, \lambda_1, \lambda_2, \bar{\mathbb{C}}_{FFT}^s\}$ set.

| Parameters | D4 | D10 | D31 |
|---|---|---|---|
| Number nodes | 4 | 10 | 31 |
| Number of training samples per node | 180 | 90 | 45 |
| Number of validation samples per node | 20 | 10 | 5 |
| Total number of training data | 720 | 900 | 1395 |
| Total number of validation data | 80 | 100 | 155 |

**Table 3.3:** Sampling size

### 3.1.7 Offline phase implementation

The DMN for a two phase laminate arrangement forms a binary tree neural network [11] of depth K. The implementation is done in Pytorch with a bottom layer initialized with weights $w_K^i$ for each of the laminates assigned. The number of laminates or weights at the $K^{th}$ layer is $2^k$. The initialization is done according to range

$$w_K^i = \frac{\hat{w}_K^i}{2^{K-1}} \quad \text{with} \quad \hat{w}_K^i \in \mathbb{U}[0, 1]. \tag{3.5}$$

The weights for each phase is assigned alternately as

$$w_K^{i,j} = \begin{cases} w_K^{i,1}, & i \text{ odd}, \\ w_K^{i,2}, & i \text{ even}. \end{cases} \tag{3.6}$$

ReLU activation is applied to the weights such that $w_K^i = \max(w_K^i, 0)$ similar to implementation proposed in Gajek et al. [10]. Due to the ReLU activation if any weight goes to zero then the corresponding tree is redundant. Avoiding this, in this work the weights are reinitialized by a small value given by $w_K^i = \max(w_K^i, 10^{-12})$,

such that the laminate is kept active during the training process. The implementation was adapted from Dey et al. [5]. In his work Dey et al. explain, the reason for keeping the laminates active: a smaller number of active laminates lead to lower prediction accuracy in long term loading [5]. Along with the weights the bottom most layers are given the input stiffness tensors $\mathbb{C}_1^s$ and $\mathbb{C}_2^s$ of the two phases as shown in equation 2.25.

The upper layers are initialized with linear interpolation parameters $p = [p1, p2, p3]$ and $q = [q1, q2, q3]$ which are identified at each neuron. The parameter sampling is done by

$$p1, p2, p3 \in \mathbb{U}[0, 2\pi], q1, q2, q3 \in \mathbb{U}[0, 2\pi]. \tag{3.7}$$

During the forward propagation, the stiffnesses at the bottom layer is homogenized using equation 2.24. Travelling up the layer results in an effective predicted stiffness $\bar{\mathbb{C}}^s_{DMN}$. The stiffness when compared with $\bar{\mathbb{C}}^s_{FFT}$ provides the error metric for the back propagation.

In the back propagation stage, the weights and parameters that influence the error are updated with the gradients of the loss function with respect to the corresponding weights and parameters. The structure of such a loss function holds the error value of the predicted stiffness $\bar{\mathbb{C}}^s_{DMN}$ and actual stiffness obtained by linear FFT $\bar{\mathbb{C}}^s_{FFT}$ given by $\|\bar{\mathbb{C}}^s_{FFT} - \bar{\mathbb{C}}^s_{DMN}\|_1$.

The loss function in the current case also incorporates regularization of the weights. In the current case the individual phases are separately controlled to add up to the respective volume fractions similar to Dey et al. [5]. There are other methods given by Liu and Wu [29] or Gajek et al. [11], that were tried where the weights collectively add up to full volume. For the current work individual regularization gave better results as compared to collective regularization, thus fixing the loss function. The resulting loss function can be seen in equation 3.8

$$J(\boldsymbol{n}, \boldsymbol{w}) = \frac{1}{N_b} \sqrt[q]{\sum_{i=1}^{N_b} \left( \frac{\|\bar{\mathbb{C}}^s_{FFT} - \bar{\mathbb{C}}^s_{DMN}\|_p}{\|\bar{\mathbb{C}}^s_{FFT}\|_p} \right)^q} + \lambda \sum_{j=1}^{2} \left( \sum_{i=1}^{\frac{2^{k-1}}{2}} w_K^{i,j} - c^j \right)^2 . \tag{3.8}$$

The parameter $\boldsymbol{n}$ is the direction of laminate and $\boldsymbol{w}$ is a list of weight parameters. The parameters p and q are power to which the error metric is raised and form part of the hyperparameter tuning set. Upon investigation considering online error p=q=1 was found to show good performance. This work uses mini batch gradient descent, where updates to the weights happen after forward propagation done on a subset of the entire training data. So here, $N_b$ is the batch size and upon investigation during the hyper-parameter tuning $N_b = 45$ was found to have good performance. The regularization parameter $\lambda$ controls the update of the weights and penalizes it [40]. In the current work $\lambda = 1000$ was seen to show good performance.

The gradient of such a loss with respect to the parameters gives the direction in which update must happen. Gradient gives the direction of steepest ascent [40]. So to minimize the error an update is created in the direction of steepest descent, thus we add a negative of the gradient to the influencing parameter. The gradients for the update is calculated using automatic differentiation module in Pytorch [42]. The update happens via optimization algorithms which specifies how the update is carried out. In the current work ADAM (Adaptive Moment Estimation) optimizer is used [6]. The optmizer ADAM combines the advantages of two other extension of stochastic gradient descent called AdaGrad (Adaptive Gradient Algorithm) and RMSProp (Root Mean square Propagation). The optmizer ADAM has additional hyper-parameters. The parameter $\beta_1$ specifies the momentum parameter for exponential weighted average of first moment of the gradients. The parameter $\beta_2$ specifies the momentum parameter for exponential weighted average of second moment of the gradients [40]. Further we choose to use AMSgrad in ADAM. The momentum parameters chosen were $\beta_1 = 0.99$ and $\beta_2 = 0.99$

The parameters are updated as per the equations 3.9

$$\begin{aligned} \mathbf{p}_{j+1} &= \mathbf{p}_j - \alpha_p \frac{\partial J}{\partial \mathbf{p}} \left(\mathbf{p}_j, \mathbf{q}_j, \mathbf{w}_j\right) \\ \mathbf{q}_{j+1} &= \mathbf{q}_j - \alpha_q \frac{\partial J}{\partial \mathbf{q}} \left(\mathbf{p}_j, \mathbf{q}_j, \mathbf{w}_j\right) \\ \mathbf{w}_{j+1} &= \mathbf{w}_j - \alpha_w \frac{\partial J}{\partial \mathbf{w}} \left(\mathbf{p}_j, \mathbf{q}_j, \mathbf{w}_j\right). \end{aligned} \tag{3.9}$$

The parameters $\alpha_p, \alpha_q$ and $\alpha_w$ were chosen to be same, $\alpha$ as given by S.Gajek et al. [10]. The parameter $\alpha$ is called the learning rate and controls the update of weights. Having a low learning rate slows down descent to a minima, high learning rate may lead to overshooting the minima and might make the system unstable. In the current work, a learning rate modulation is used to change the learning rate during the learning process. The first learning rate scheduler is a decay method which reduces the learning rate as the training goes on in a geometric progression. The resulting module is exponential learning rate decay in Pytorch [43]. The second learning rate scheduler chosen is cosine annealing with warm restart implemented in Pytorch [30]. The resulting modulation of learning rate can be seen in equation 3.10

$$\alpha : \mathbb{N} \to \mathbb{R}, \quad m \mapsto \gamma^m \left(\alpha_{\min} + \frac{1}{2}\left(\alpha_{\max} - \alpha_{\min}\right)\left(1 + \cos\left(\pi \frac{m}{M}\right)\right)\right). \tag{3.10}$$

Here m denotes the current epoch, the parameter $\gamma$ denotes the multiplying factor of the learning rate decay. In this work it is chosen as $\gamma = 0.999$. The parameter $M$ denotes the epoch at which learning rate is restarted, in this case it is chosen to be $M = 20$. Instead of having a fixed learning rate one need to specify a maxima and mininma for the learning rate in warm restart. This annealing type decay is geometrically decayed by the exponential decay. The learning rate maximum is chosen to be $\alpha_{\max} = 0.0005$ and the learning rate minimum is chosen to be $\alpha_{\min} = 0.00005$.

An epoch is completed when the minibatch training cycle covers the full set of the training data. A training cycle includes forward propagation step of the minibatch and

backward propagation update. This training procedure is carried out for 6000 epochs and training states every 50 epoch is saved. This is because, good fit of the Linear elastic data does-not imply very low online errors. So there is a need to investigate the progression of online error over the entire range of 6000 epochs.

To check if the training is effective for the linear Elastic FFT data, the validation is performed at every epoch and stored for postprocessing. Given that the number of possible variations of training are huge and knowing that the online evaluation of a model is lot more time expensive than the training itself, decision had to be taken on what model to choose for online evaluation. The best DMN is the model that performs well in the nonlinear elastic domain (In this case plasticity). The co-relation to linear elastic domain is not quantifiable. But under initial investigation it was concluded that:

- The chances of identifying a good interpolated DMN are higher for the training with the lowest validation error.
- The behaviour of the training curve had some role in error estimates of the online phase. For example a well behaved training curve without abrupt changes in the loss gave better results in the online phase.

Up until this point in the framework, no manual intervention is required. The user always interacts from the central configuration file. Now based on the plots that are automatically documented in the folders, a user of the framework will take a decision on what model to proceed with for the next stage of the evaluation.

In the current case study, a model was chosen for online evaluation based on the points listed. The table 3.4 shows the hyper-parameters chosen for the best training. It should be noted that since the co-relation of the online phase with the training is not quantifiable, the rules listed might change for different materials properties chosen from the configuration file. The identified model has a layer depth of nine. It was seen that higher depth till nine, the validation error reduced. Beyond nine layer depth it became increasingly time consuming to train. Similar trend can be seen in the online evaluation stage. Indeed the choice of layer depth and other hyper-parameters were mainly done, keeping in mind the errors in the online evaluation stage. It was seen that a lower depth of seven and even eight were seen to have insufficient performance in the online performance on the D10 discretization.

### 3.1.8 Online evaluation and methods

The goal of the online evaluation stage is to identify the epoch at which the best DMN occurs and document the corresponding ability of the DMN to interpolate and generalize. The next stages of the framework needed tools to aid the user in user in identifying the best DMN.

| Parameters | Values |
|---|---|
| Discretization chosen | D31 |
| Interpolation type | Linear |
| Weight limit | $1.0 \times 10^{-12}$ |
| P-norm local p | 1 |
| P-norm global q | 1 |
| Batch size | 45 |
| Regularization $\lambda$ | 1000 |
| Type of regularization for weights | Individual phases add-up to volume fractions |
| Optimizer | Adam |
| Learning rate modulation 1 | Exponential decay |
| Learning rate modulation 2 | Cosine annealing with warm restart |
| Momentum parameter first moment $\beta_1$ | 0.99 |
| Momentum parameter second moment $\beta_2$ | 0.99 |
| Numerical stability parameter $\epsilon$ | 1e-08 |
| AMSGrad | True |
| Number of epochs | 6000 |
| Layer Depth | 9 |
| Exponential decay parameter $\gamma$ | 0.999 |
| Learning rate minima $\alpha_{\min}$ | 0.00005 |
| Learning rate maxima $\alpha_{\max}$ | 0.0005 |
| Restart Epoch count M | 20 |
| Multiplying factor for Restart epoch | 1 |

**Table 3.4:** Hyper parameter chosen

The evaluation is done using the procedure described in section 2.5.3. Any simulation requires geometry, material properties and boundary conditions [25]. As explained in section 2.5.2 the DMN learns a simplified version of a microstructure in the form of laminates. Even though this laminate arrangement is a mathematical description, one can still find an analogous geometry. The material properities of the respective laminates are assigned with the models described in section 2.3.

The boundary condition of the loading is dependent on the choice of case study. For plasticity this work uses an orthotropic boundary condition. Figure 3.10 shows the loading condition for a single microstructure for plasticity obtained from a previous study by Dey et al. [5]. For plasticity uniaxial hysteresis strain loading $\bar{\epsilon}_{ij}$ applied for a time scale of 4 seconds with a strain amplitude of 2.5% scaling 40 loadsteps. In total there are six loadcases evaluated per node. The resulting curve from DMN is compared with the loading curve on the complex microstructure and the relative

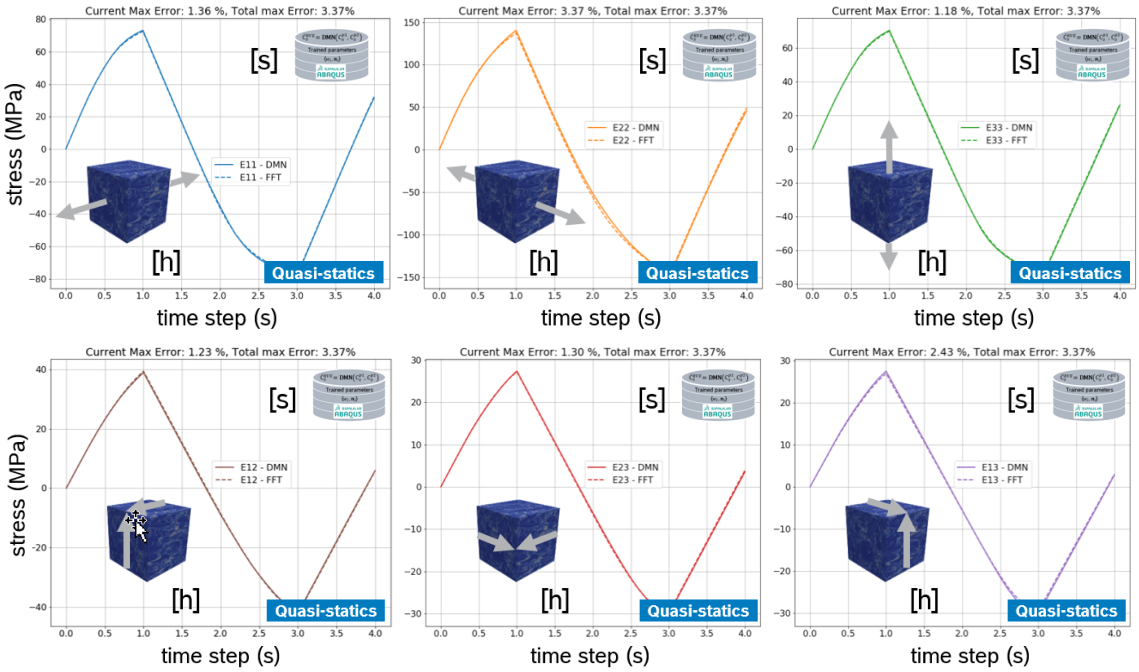



**Figure 3.10:** Boundary condition for plasticity [4]

error metric is evaluated for each of the load cases given by

$$e_{ij,t}^{\text{plast}} = \frac{\left\| \bar{\boldsymbol{\sigma}}_{ij,t}^{\text{DMN}} - \bar{\boldsymbol{\sigma}}_{ij,t}^{\text{FFT}} \right\|}{\left\| \bar{\boldsymbol{\sigma}}_{ij,max}^{\text{FFT}} \right\|}. \tag{3.11}$$

Since the work is aimed at product engineering, the cretiria for evaluation would be the maximum error for all the load cases over all the time steps $\mathcal{T}$ and is given by

$$e_{max}^{\text{plast}} = \max_{i\in(1,3),j\in(i,3)} \max_{t\in\mathcal{T}} (e_{ij,t}^{\text{plast}}). \tag{3.12}$$

Calculations can also be done to find the mean value over all the load cases and time steps as shown

$$e_{mean}^{\text{plast}} = \max_{i\in(1,3),j\in(i,3)} mean_{t\in\mathcal{T}} (e_{ij,t}^{\text{plast}}). \tag{3.13}$$

In the above formulas ij refers to the loadcases and t refers to time steps. On a similar note, the max error for individual load cases are also calculated

$$e_{max,ij}^{\text{plast}} = \max_{t\in\mathcal{T}} (e_{ij,t}^{\text{plast}}). \tag{3.14}$$

Calculations are done to find the mean value for individual load cases and time steps as shown

$$e_{mean,ij}^{\text{plast}} = mean_{t\in\mathcal{T}} (e_{ij,t}^{\text{plast}}). \tag{3.15}$$

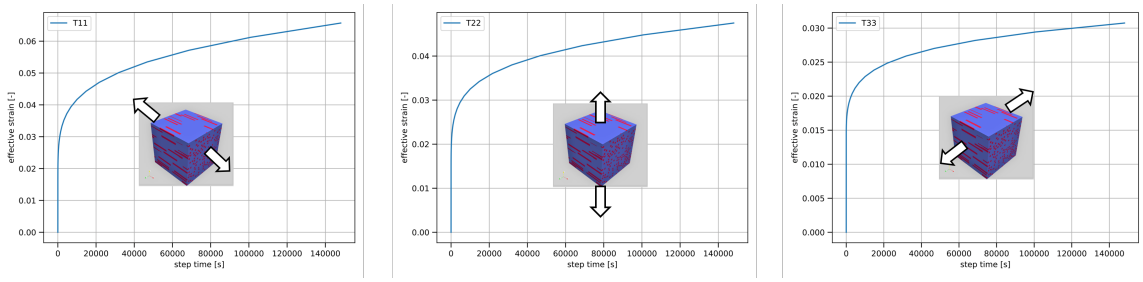



**Figure 3.11:** Boundary condition for creep

Just as a side note, validation of DMN in plasticity is done on 109 different nodes or microstructure. For a full evaluation of the DMN state in the training a total of $109 \times 6 = 654$ computations are to be performed.

For creep this work uses a uni-axial long term tensile testing boundary condition performed on principle directions x,y,z. Figure 3.11 depicts loading condition for a single microstructure. Since activating the creep response is closely tied to the extent to which matrix is loaded, the fiber orientation affects the choice of loading duration and load levels. So for every sampling point a new load level has to be chosen. The table 3.5 shows the chosen loads for all four points. Within the scope of this work it was decided to perform creep evaluation for D4 discretization only. During the tensile testing the load is ramped up for the first 8.5 seconds and then kept constant for $1.48 \times 10^5$ seconds in total spanning 32 time steps. The relative error of the strain over time is calculated using equation as shown

$$e^{\text{creep}}_{ii,i\in(1,3),t} = \frac{\left\| \bar{\boldsymbol{\varepsilon}}^{\text{DMN}}_{ii,t} - \bar{\boldsymbol{\varepsilon}}^{\text{FFT}}_{ii,t} \right\|}{\left\| \bar{\boldsymbol{\varepsilon}}^{\text{FFT}}_{ii,max} \right\|}. \tag{3.16}$$

Since the work is aimed at product engineering, the criteria for evaluation would be

| Parameter | Load case | 1_D4 | 2_D4 | 3_D4 | a_4_D4 |
|---|---|---|---|---|---|
| Load level in MPa | T11 | 80 | 60 | 50 | 60 |
| | T22 | 50 | 60 | 50 | 50 |
| | T33 | 50 | 50 | 50 | 50 |

**Table 3.5:** Load levels chosen for D4 discretization

the maximum error over all the time steps $\mathcal{T}$ and is given by

$$e^{\text{creep}}_{ii,i\in(1,3),max} = \max_{t\in\mathcal{T}}(e^{\text{creep}}_{ii,t}). \tag{3.17}$$

The microsimulation requires material properties, boundary conditions and geometry which in the actual simulation is the complex microstructure. In this work the simplification of the geometry aspect to an equivalent laminate microstructure makes

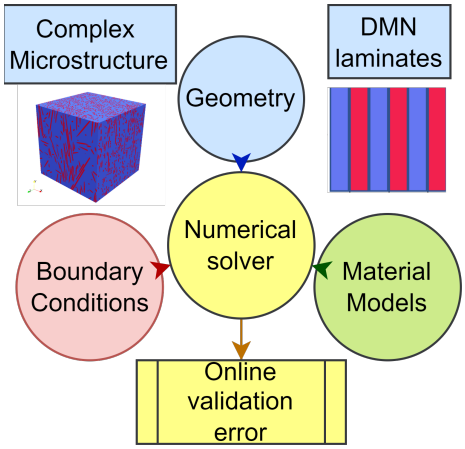



**Figure 3.12:** Online evaluation

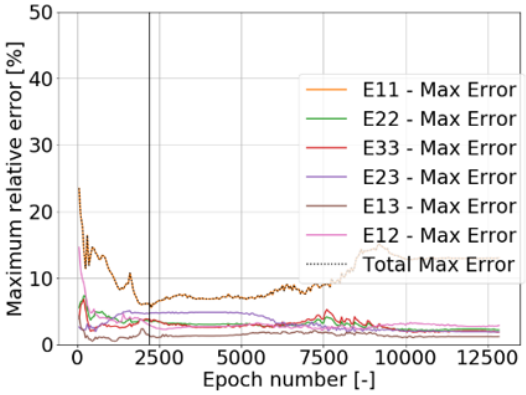
the overall micro-simulation faster. The evaluation of a good trained model is done by comparing of the two version of nonlinear solution. Figure 3.12 gives an explanation.

In this work a trained D31 model is run on 109 different microstructures for a full evaluation of plasticity. For the sake of illustration we plot a full online error plot for online evaluation over the entire training epochs for a model run for 12000 epochs. To obtain these plot, states at every 50 epochs are stored and evaluation is run for four microstructures on the FOT triangle. The error plots are shown in figure 3.13.

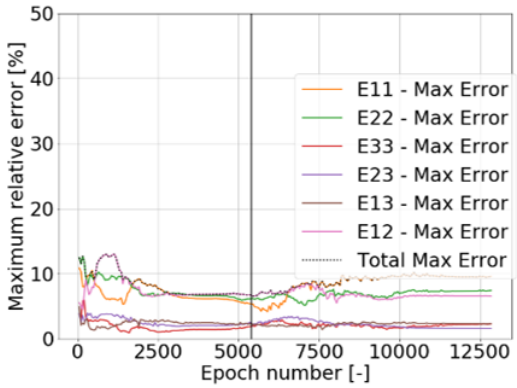
Upon multiple such investigations few conclusions were derived

- The error plots in most cases have very little co-relation with the behaviour in the training curve.
- Co-relation between the microstructure configurations are very little. For example in figure 3.13 errors over 5000 Epoch raises in 1_D4 and 2_D4,but reduces in 3_D4.
- The model state with the lowest error are mostly found in the intermediate epochs.
- The model state with the lowest error are identified at different epochs for different microstructures. For example in figure 3.13 the best DMN for 1_D4 occurs at around 2500 epochs, but for 2_D4 it occurs at around 5000 epochs.
- The DMN reproduces the behaviour of few microstructures well campared to few others. 3_D4 has lower errors than a_4_D4.

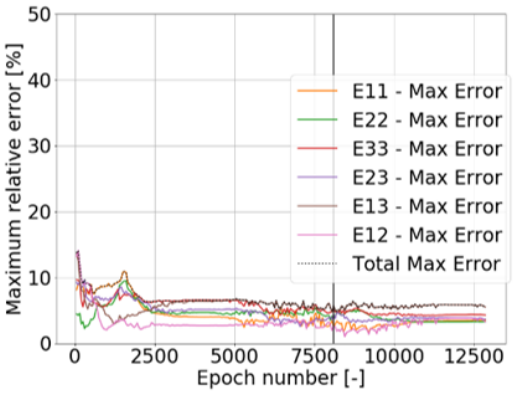
The reference from the observation is that finding a single model that reasonably works for all the 109 points is a very challenging task because of the time required for



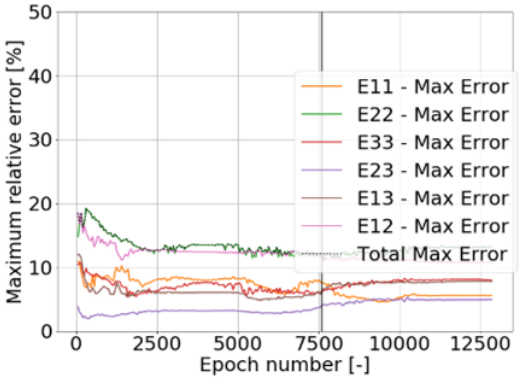

**(a)** 1_D4 microstructure error plot




**(b)** 2_D4 microstructure error plot




**(c)** 3_D4 microstructure error plot




**(d)** a_4_D4 microstructure error plot

**Figure 3.13:** Error plot of all six loading cases of plasticity over the entire epoch range. This work is more concerned with the maximum of all the errors. The verticle line points out the epoch state at which the DMN has the lowest error in the entire range.

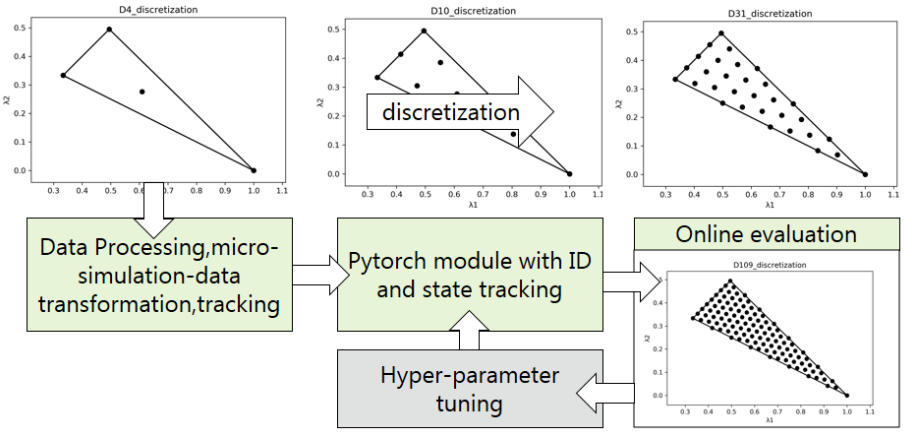



**Figure 3.14:** The image shows the major stages in the framework

full evaluation. Not to mention that there also has to be an informed decision to be made for hyper-parameter re-tuning as seen in figure 3.14.

### 3.1.9 Selective distillation algorithm

Looking at the expense for a full evaluation as shown in table 3.6,there is a need to selectively reduce the computations.

| Parameters | Plasticty case study |
|---|---|
| Time for one online evaluation using DMN | 60-90 seconds |
| Number of epochs | 6000 |
| States storage frequency | Every 50 epochs |
| Number of states to be evaluated | 6000/50=120 states |
| Number of nodes on which evaluation is run | 109 |
| Evaluations for orthotropic boundary condition | 6 load cases |
| Total number of evaluations to fully validate training | $120 \times 109 \times 6 = 78480$ evaluations |
| Time taken under parallelization of 6 load cases | **9-14 days** |

**Table 3.6:** Time taken for full evaluation

In order to reduce the time consumed the selective distillation algorithm was developed. The algorithm considers the 109 nodes as filter papers stacked over each other and these filter papers have an error metric which is analogous to pore sizes. The error-metric is dynamically changed such that it filters out the states with high errors. At the end of the distillation process one finds the state which survives with the least error over 109 nodes. The algorithm is as shown in algorithmic table 1.

**Algorithm 1** Selective distillation algorithm

state-list : $S_1 \dots S_M$

Node-list : $N_1 \dots N_N$

**while** $count = 109$ **do** ▷ Until we reach the bottom-most filter
  $count = 0$
  **for** `node in node-list` **do**
    **if** `node = first node` **then**

      **for** `state in state-list` **do**
        $E_1^{state}$ `evaluate to list` $E_1^M$
      **end for**
        Error-metric = min of list $E_1^M$
        Update= $\frac{maxE_1^M - minE_1^M}{1000}$
        `state-list : state with` $E_1^{state} \leq Error - metric$
      **else**
        **for** `state in state-list` **do**
          $E_{node}^{state}$ `evaluate to list` $E_{node}^m$ ▷ $m \leq M$ states removed
        **end for**
        state-list : state with $E_{node}^{state} \leq Error - metric$
      **end if**
    **end for**
    **if** `state-list is empty` **then**
      Error-metric = Error-metric + Update
    **end if**
  **end while**

Further several improvements to the algorithm were tried-out.

- Update parameter is very gradual and equally spaced. A modification of this would be to set the update such that only one state is allowed to the next stage. This reduces unnecessary computation for a bad training case study.
- Another modification that was helpful was introducing node ranking. One could understand it as changing the priority of the nodes based on the last point of the conclusion 3.1.8.

# 4 Numerical investigation and results

## 4.1 Training and validation data generation

The training/validation data generation is done using FFT solver FeelMath [13]. The automation workflow allows for utilization of multiple FFT runs for a single microstructure with different boundary conditions and material properties. The framework can create multiple instances of this automation allowing FFT runs for different microstructures or geometries in parallel, thus making the process more convenient. The time expense during data generation remains the same.

## 4.2 Ground truth data generation

The ground truth data generation requires running of nonlinear FFT runs on complex microstructure geometries. The time consumed is greatly dependent on the resolution of the microstructure, ratio of the stiffness of the two phases, geometry of the fibers along with its distribution in the matrix and many more. In this work ground-truth generation workflow was only utilized for plasticity case study. Since the online phase is evaluated on chosen material properties of interest, the ground truth generation is also run on the same properties. Plasticity has equivalent load levels and boundary conditions for each microstructure on the FOT triangle. This will change for creep case study since each microstructure on the FOT triangle has different load level and loading duration. The framework allows for use of more FFT runs at a time. As a case study we ran three microstructures at a time. The table 4.1 shows the speed up of results with and without the framework.

| Parameters | Value without framework | Value with framework |
|---|---|---|
| Time for one FFT run | 0.5 to 4 hours | 0.5 to 4 hours |
| Number of nodes to run | 109 | 109 |
| Number of loadcases | 6 | 6 |
| Parallelization | N/A | three microstructures at a time |
| Number of licenses used | 6 | 18 |
| Time expense | 54.5 hours to 436 hours | 18 hours to 145 hours |

**Table 4.1:** Time expense comparision for ground truth generation.

## 4.3 Training Framework

The framework allows for parallelization of training and online evaluation. Each instance of the training is identified with a unique identification number given by $(Alphabet - series)\_(four - digit - training - number)$. Before the training is initialized the hyper parameters are documented and thus one can track the parameters used. The data is stored with the corresponding training-ID and post training, the corresponding objects can be re-initialized for online evaluation. The convenience of the set-up helps one to run multiple trainings at once on the HPC. As an improvement to this, the usage of GPU cluster can further speed up each of the trainings, thus making it an efficient training framework for model identification. In the current case study gains of three to five times have been achieved.

| Parameters | Value without framework | Value with framework |
|---|---|---|
| Timeline for training | 3-5 months | 1-1.5 months |

**Table 4.2:** Timeline for model training

## 4.4 Online evaluation using selective distillation algorithm

The online evaluation process was significantly speed-up in this work using the selective distillation algorithm explained in section 3.1.9. The speed-up mostly depends on the maximum error-metric the algorithm reaches. The higher the error-metric more is the number of evaluations, thus increasing the time for full evaluation of DMN on the FOT triangle. As a case study for a good training hyper-parameters mentioned in table 3.4, the time gains are listed in table 4.3.

| Parameters | without distillation | with distillation |
|---|---|---|
| states to be evaluated | 120 states | 120 states |
| Number of nodes | 109 | 109 |
| Number of load cases | 6 load cases | 6 load cases |
| Full evaluation number | $120 \times 109 \times 6 = 78480$ evaluations | 800 evaluations |
| Total time taken | 9-14 days | **Approximately 15 hours** |

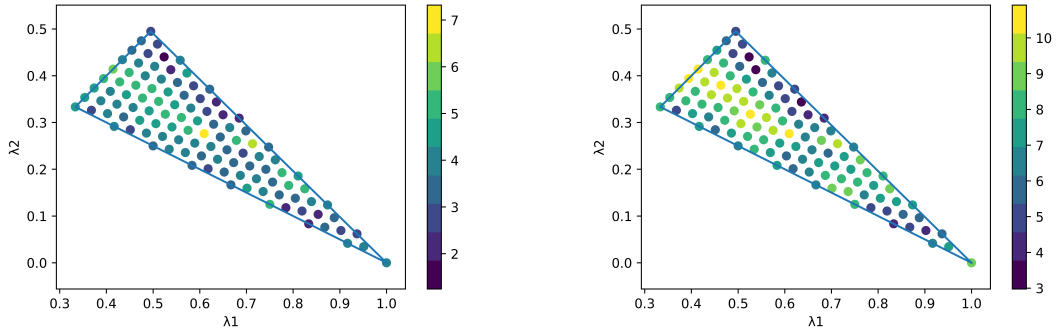
**Table 4.3:** Time comparision for full evaluation

## 4.5 Hyper-parameters and numerical experiments for plasticity

Each of the parameters listed in table 3.4 has its own effects on the online error. One of the main road-blocks in automating model identification is quantifying the effects of hyperparameter tuning on online error. Even though within the scope of this study, it was not possible to quantify the effects, having automated documentation helps the user make informed decisions for changing the hyper-parameters. Upon successive trainings in the current case-study few intuitions over the hyper-parameters could be listed:

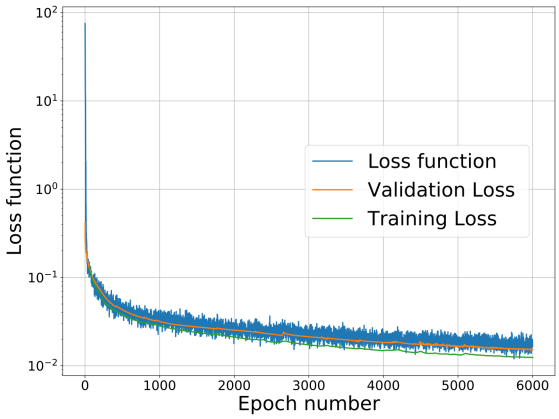
- Training on higher discretization reduces online error.
- Lower discretizations (D10) performed well in the online phase when trained without warm restart.The opposite was true for D31 discretization.
- The first moment momentum parameter had significant relevance in reducing the online error in addition with warm restart.
- Choosing higher local and global powers did not change the validation error and thus the online errors. Mainly 1 and 10 were tried out from previous literature [10], [5]. It could be observed that the training behaviour did not change with the two variations.
- Online error reduces with increase in layer depth.
- The maximum and minimum learning rates have some effect in reducing online error.
- Numerical experiments have been conducted to find the right values for restart parameters and batch size which had some relevance to achieve lower validation error and online error.

The best training was obtained with the hyper-parameters listed in table 3.4. As described in section 3.1.8, the maximum and mean errors on the 109 validation points of the FOT triangle over all the 6 load cases are plotted. The individual loadcases are also plotted just to see if a single loadcase has consistent error, see chapter 6.




**(a)** Training A_0031 mean error distribution as given by equation 3.13

**(b)** Training A_0031 maximum error distribution as given by equation 3.12




**(c)** Training A_0031

**Figure 4.1:** Aggregated plots of both Maximum and mean error shown along with the training curve

The table 4.4 shows the values achieved for each load case. It gives a relative idea of the signifance of few load cases, where in some loadcases have higher errors than few others.

| Loadcase | Maximum error in % | Mean error in % | Variance of distribution |
|---|---|---|---|
| Aggregate | 10.9168 | 7.3187 | 3.1287 |
| E11 | 9.9367 | 6.1844 | 4.3623 |
| E22 | 10.5649 | 7.3187 | 2.0160 |
| E33 | 4.4450 | 2.3309 | 0.7727 |
| E12 | 10.9168 | 5.6051 | 7.7286 |
| E23 | 10.7361 | 5.4628 | 6.6072 |
| E13 | 7.9556 | 4.6762 | 2.0638 |

**Table 4.4:** Values achieved in online evaluation phase of each loadcase

Looking at the results presented and multiple experiments conducted a few observations can be noted

- It should be noted that DMN has the tendency to preform well on few microstructures and relatively bad on few others.
- The best model is usually the one whose maximum error is the lowest and has high variance on the error distribution. The next stage of the training should involve a study to identify change of variance over the epochs and dynamically stop the training when the error is within engineering factors and when there is reduction in variance.
- A more detailed version for model identification would be to understand the error distribution over the FOT triangle. This would help quantify the mapping between the offline training space and performance in the online evaluation. Effectively it could be suggested that a meta model of error distribution can be created over the FOT triangle and linking it to the hyper-parameters chosen in the offline training to automate model identification.
- The selective distillation algorithm combines the FOT triangle and error distribution in a deterministic way of identifying the model. The algorithm does-not involve any statistical methods to quickly guess the best epoch to run the evaluation on, which could further reduce the number of evaluations. The model was kept deterministic because of the risk involved in skipping the best model. The next stage might require more sophisticated methods to reduce evaluations.

## 4.6 Creep or long term loading

Considering the timelines and given that the errors were within engineering tolerance for plasticity, the investigations were continued for creep. The creep case study was run for D4 discretization and the load levels were chosen as per table 3.5. The creep case study was not performed using the framework and the experiments were performed manually. Given the timelines, it was not feasible to retrain the model for a good creep evaluation. So this work presents the results obtained for creep evaluation using the best model obtained during plasticity, which in this case is model A_0031 of D31 discretization. The evaluations were performed using long term tensile testing as per procedure explained in section 3.1.8.The figures 4.2 shows the comparision of the experiments with the full nonlinear FFT.

A few observations can be noted:

- The DMN microstructure is seen to be stiffer in few cases and softer than the complex microstructure in few other cases. So errors are not systematic.

- The errors are significantly higher when compared to plasticity loadcases. Although performing offline training for improving performance in creep was out of the scope of this work, D4 creep evaluation case-study will provide some benchmark for decision making in the next stages.

- In all of the case-studies one could see the chosen loading duration and load-levels are sufficient to initiate a creep behaviour in the material. Most of these choices were based on previous experience with trial and error. This decison making will get challenging for evaluation of D109 discretization. Intuitively, similar load-levels and loading durations can trigger creep response in similar nearby microstructure on the FOT triangle. So for D109 case, a mapping might be possible to set the load levels.

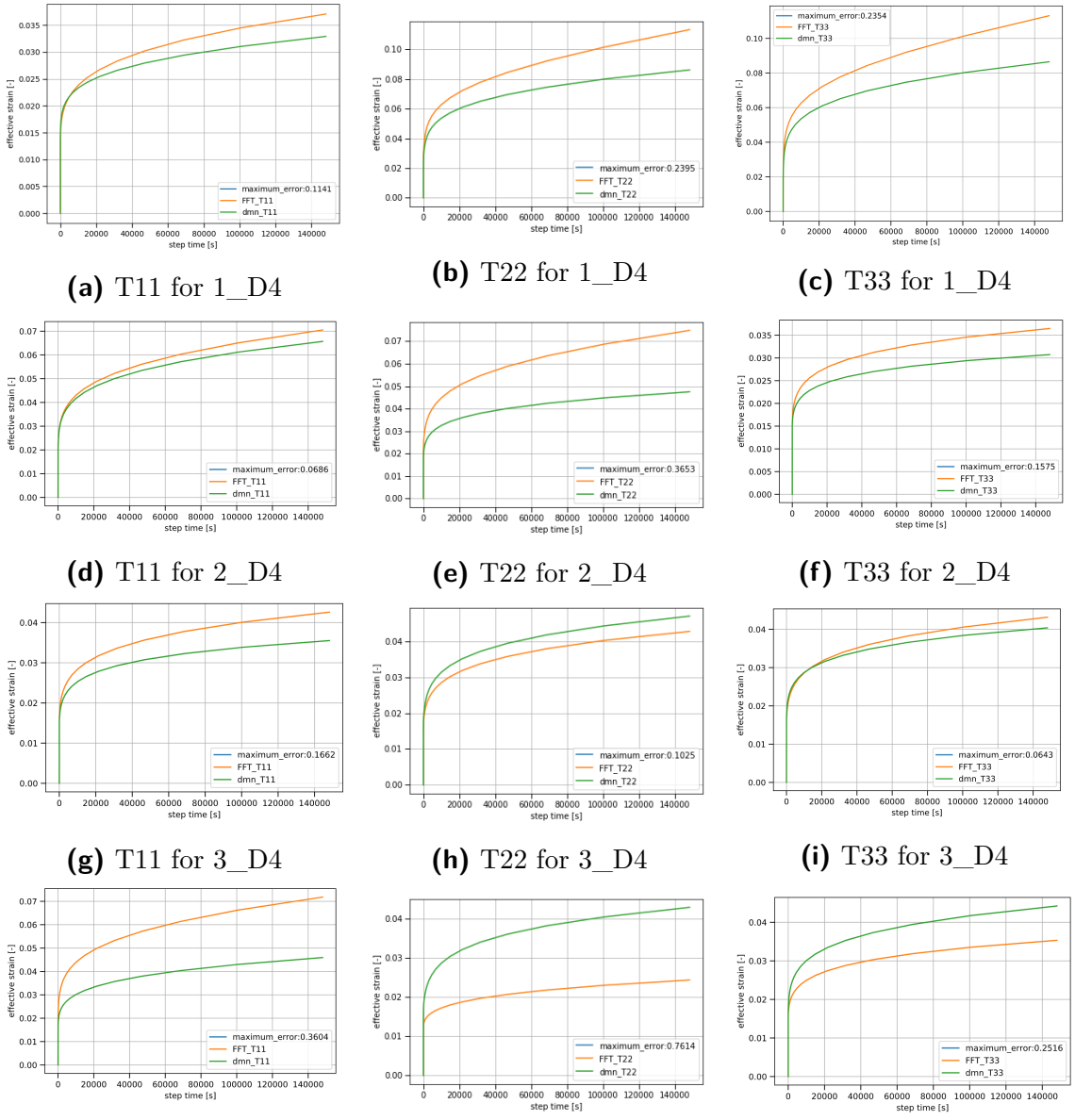
- It can be seen that there was standardization of loading duration for all the 4 nodes. This allows for relative comparision of the experiments in different loading direction and microstructure variation. Since the creep response mainly depends on load carried by the polymer matrix, the load-levels for D109 can be set considering a relation between fiber orientation and loading direction to trigger a creep response in the material.

- On an observable note, more is the duration of loading and load levels, higher is the nonlinearity leading to higher errors.

| loadcase | max error in % T11 | max error in % T22 | max error in % T33 |
|---|---|---|---|
| 1_D4 | 11.41 | 23.95 | 23.54 |
| 2_D4 | 6.86 | 36.53 | 15.75 |
| 3_D4 | 16.62 | 10.25 | 6.43 |
| a_4_D4 | 36.04 | 76.14 | 25.14 |

**Table 4.5:** Comparision of long term loading on complex and DMN microstructure

Due to high number of laminates, creep evaluation does not show any significant gain in speeds when compared to standard FFT based homogenization-see table 4.6. So when identifying a model for evaluating creep it is necessary that these speeds have to be increased by making the nonlinear solution method routines more efficient.It should be noted that creep evaluation using FFT based homogenization depends greatly on the microstructure image which in this case is very simplified. This is also the reason why evaluation seems very comparable.




**(a)** T11 for 1_D4 **(b)** T22 for 1_D4 **(c)** T33 for 1_D4

**(d)** T11 for 2_D4 **(e)** T22 for 2_D4 **(f)** T33 for 2_D4

**(g)** T11 for 3_D4 **(h)** T22 for 3_D4 **(i)** T33 for 3_D4

**(j)** T11 for a_4_D4 **(k)** T22 for a_4_D4 **(l)** T33 for a_4_D4

**Figure 4.2:** Plots showing the D4 creep evaluation comparision of DMN and full nonlinear FFT

| Node | loadcase | homogeneization time-FFT | homohenization time-DMN |
|---|---|---|---|
| 1_D4 | T11 | 10410 seconds | 1470 seconds |
| | T22 | 11393 seconds | 2477 seconds |
| | T33 | 8454 seconds | 2383 seconds |
| 2_D4 | T11 | 3255 seconds | 1736 seconds |
| | T22 | 8128 seconds | 1472 seconds |
| | T33 | 2478 seconds | 1358 seconds |
| 3_D4 | T11 | 6166 seconds | 1492 seconds |
| | T22 | 1727 seconds | 1675 seconds |
| | T33 | 2371 seconds | 1513 seconds |
| a_4_D4 | T11 | 13677 seconds | 1488 seconds |
| | T22 | 1421 seconds | 1557 seconds |
| | T33 | 1306 seconds | 1591 seconds |

**Table 4.6:** Comparision of simulation runtime in long term loading on complex and DMN microstructure

## 4.7 Identified model in the context of multiscale simulations

Consider a case study of component level multiscale simulation with over a million mesh elements something like figure 2.7. Running a multiscale simulation with the material properties mentioned in table 3.2 is very time expensive while using a standard solver. One other aspect of the Hybrid solver is that speed of the solution does not depend on the topological image of the microstructure, since eigen values of the FOT is used in hybrid solver. The same does not apply to standard solver, since the speeds vary based on the microstructure image. One can identify the drastic reduction in compute time measured in years as shown in table 4.7. Running a multiscale simulation using standard solver is often impractical given the projected timescale.

| Parameter | standard solver | hybrid solver |
|---|---|---|
| component elements | 1 million | 1 million |
| homogenization time per element | 0.5-4 hours | 60-90 seconds |
| Total compute time | 57-457 years | 2-3 years |

**Table 4.7:** Comparision of compute time for multiscale simulation of plasticity

# 5 Conclusion and outlook

## 5.1 Conclusions

This work introduces a hybrid solution paradigm for performing homogenization called deep material network. In this solution method machine learning is used to learn the representation of a complex microstructure in the form of laminate arrangement, such that under similar boundary conditions and material models, both microstructures give equivalent mechanical response. This is called the direct deep material network. The current work starts with the extension of direct-deep material network. Additionally an interpolation function is introduced in the machine learning model to jointly identify the laminates over a fiber orientation triangle, such that the arrangement of these laminates can be interpolated to reproduce the mechanical response of any possible microstructure variation occuring in chosen SFRT material. This interpolated deep material network (interpolated FE-DMN) forms the hybrid solver which which provides a markes speed-up compared to the traditional fast fourier transform-finite element (FFT-FE) based homogenization without a loss in accuracy.

One of the needs of such a solution method is that, based on the composite parameters chosen a new model has to be identified for every material case-study. One could imagine a database full of such models created for every possible SFRT material available in industrial use. Given the number of such materials available in industrial practices, this work proposes a framework to ease and benchmark the procedure required for machine learning model identification. The engineered framework called NumProDeep, utilizes HPC for running multiple calculations and tracks the large number of variables involved in the process using a finite element like software architecture. Here the fiber orientation triangle with a certain sampling is given an analogy of elements, microstructure configuration are given an analogy of nodes and sampling over the triangle is given an analogy of meshing making it like a mini-FE like software package. It also provides useful functions and algorithms that help speed-up the process and help the user make informed decisions for hyper-parameter tuning. Most of all the framework is made scalable allowing integration of more sophisticated functions and methods for full automation of hyper-parameter tuning and machine learning model identification. Usage of a data-science platform based on the FE software architecture opens up many possibilities for exploration.

As a case study, the framework was used for model identification on a microstructure parameter set. In the first phase, the case study was for plasticity. The data generation workflow was successfully implemented for all the mirostructure sampling, image-data generation and parallelized FFT simulation runs on the FOT triangle. The workflow executes end to end processing without any manual intervention. Additionally parallelization algorithm designed for ground truth generation reduced compute time by three times. The training framework also integrated the existing online evaluation workflow allowing for an end to end integrated environment to build more sophisticated algorithms. The implementation of self documentation and automated management of trainings reduced the thesis timeline by three to four months. This also helped towards a structured understanding of the training and its behaviour in the online evaluation phase. As a pointer in this direction, a deterministic algorithm called the selective distillation was developed using the error-distribution over the FOT triangle. The prime focus of the algorithm was to reduce computation required for full evaluation of a training. The reasoning for these sophistications is the difficulty in quantifying co-relation of offline training with its behaviour in online evaluation for 109 sampling points. The choices of loss function structure and hyper parameter tuning were based on available literature and experience. The best model identified for plasticity was seen to have a maximum error within 10.91% and a mean error within 7.31%. Given that industrial parts have a high factor of safety, it was decided that the search for the best model was satisfactory for commercial use.

In the next phase of the thesis, creep evaluation was attempted using the same model identified in plasticity. The framework was not used for the creep case study owing to the workload required within the timeline of the study. The aim of the study was to benchmark online evaluation procedure on D4 discretization, setting future directions and procedures for identifying a hybrid solver for creep. Based on numerical experiments on the microstructures, decisions were made on the load levels and duration of loading for D4. This led to standardization of loading duration allowing for relative comparision of load-levels for different loading directions and fiber orientations. The errors obtained during creep evaluation was relatively high compared to plasticity case. In general it was seen that higher load levels,higher duration of loading leads to a higher error due to increase in nonlinearity. It was also seen that the compute time for creep evaluation was comparable to FFT based creep evaluation due the high number of laminates in DMN creep evaluation.

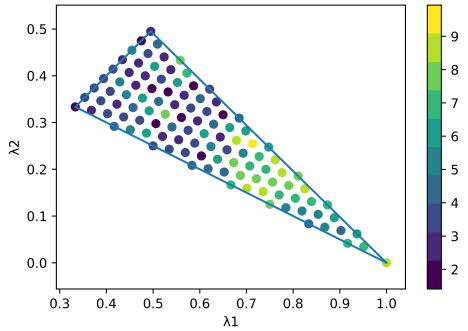
## 5.2 Outlook

DMN model identification is a complex process and this thesis provides a lot of pointers towards making this process simpler.

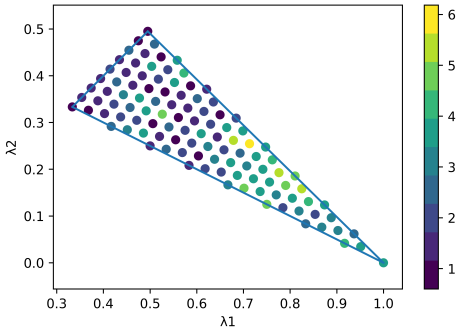
- The paradigm of a FE-framework for identification of such a model allows for advanced experiments on the FOT traingle. Sampling over the FOT triangle can be made coarser or finer based on the error distribution. Currently the discretization algorithm is just a bisection algorithm. A different version of the algorithm can be employed because FOT triangle is an isosceles triangle. So a bisection algorithm based sampling might introduce bias in the training since one of the sides is smaller than the other two.
- The element and node objects are powerful data-structures to store information. It might allow for more interpolation concepts for many other parameters like mechanical responses.
- There is a possibility to integrate advanced design of experiment-DOE frameworks like optuna [41] to manage multiple experiments at once using the power of the HPC. Since the only limitation to running experiments in parallel is the compute power available in HPC, there is scope to create such a DOE framework for any given material.
- Characterizing the training behaviour on the fly with online evaluation will be important for creep evaluation. Allowing for a D10 evaluation in intermedient epochs might give a good idea of chosen hyper-parameters during training. More over understanding the dynamics of the error distribution change over the FOT triangle as a function of all the hyper-parameters like epochs,etc. will help make model identification process faster.
- Given that the creep evaluation was slow, there is a need for more efficient implementation of the online evaluation which currently is a FORTRAN routine. Usage of efficient linear algebra libraries might help speed-up evaluation.
- The identified model for plasticity should be now, used to build a workflow towards component simulations.

All in all, one can propose many changes that prepare the NumProDeep framework as an effective software tool for identifying efficient hybrid solvers for SFRT materials of the future.

# 6 Individual plots

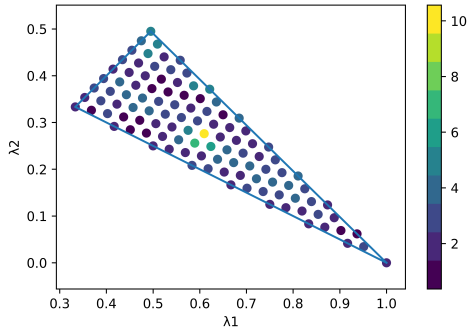



**(a)** Maximum error distribution for E11 load case-eqn 3.14

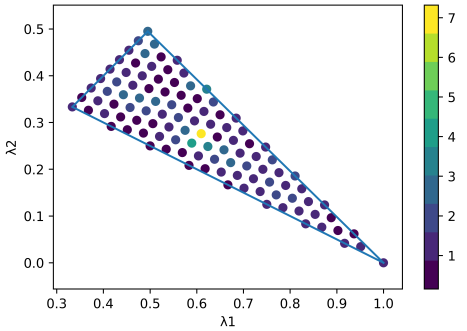



**(b)** mean error distribution for E11 load case-eqn 3.15

**Figure 6.1:** Plots showing distribution for E11 load case

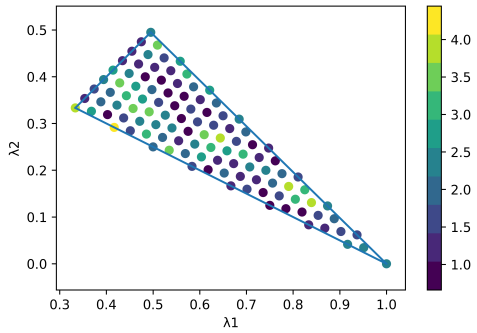



**(a)** Maximum error distribution for E22 load case-eqn 3.14

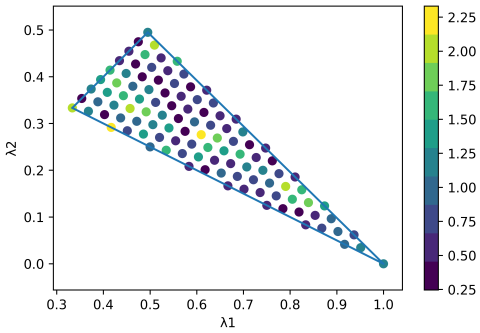



**(b)** mean error distribution for E22 load case-eqn 3.15

**Figure 6.2:** Plots showing distribution for E22 load case

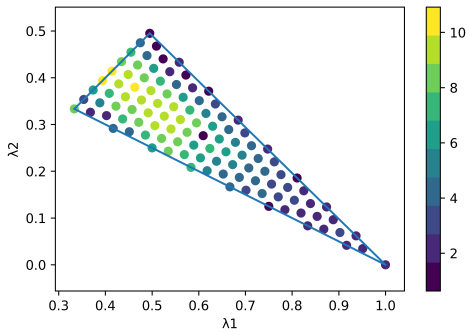



**(a)** Maximum error distribution for E33 load case-eqn 3.14

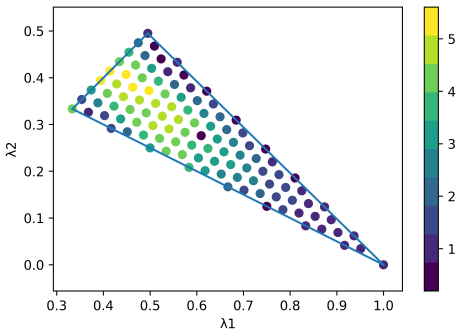



**(b)** mean error distribution for E33 load case-eqn 3.15

**Figure 6.3:** Plots showing distribution for E33 load case

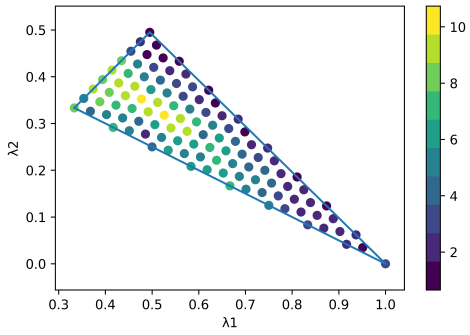



**(a)** Maximum error distribution for E12 load case-eqn 3.14

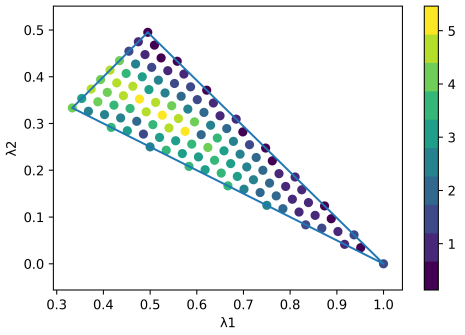



**(b)** mean error distribution for E12 load case-eqn 3.15

**Figure 6.4:** Plots showing distribution for E12 load case

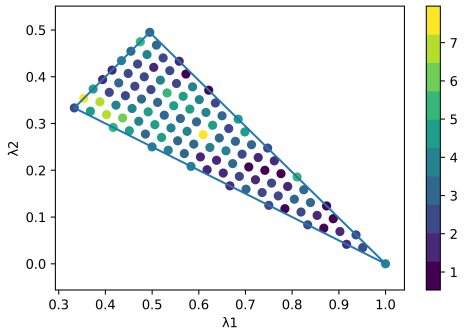



**(a)** Maximum error distribution for E23 load case-eqn 3.14

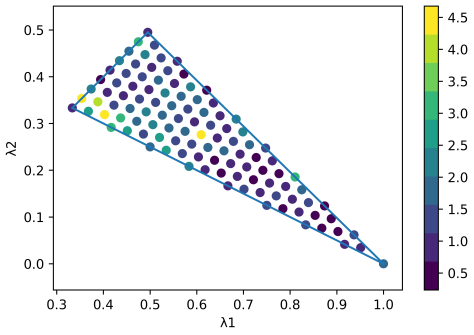



**(b)** mean error distribution for E23 load case-eqn 3.15

**Figure 6.5:** Plots showing distribution for E23 load case




**(a)** Maximum error distribution for E13 load case-eqn 3.14




**(b)** mean error distribution for E13 load case-eqn 3.15

**Figure 6.6:** Plots showing distribution for E13 load case

# Bibliography


[1] Suresh G. Advani and Kuang-Ting Hsiao. *Manufacturing techniques for polymer matrix composites(PMCs)-ch2*. eng. 1. ed., repr. United Kingdom: Oxford: Woodhead Publishing, an imprint of Elsevier, 2012. 504 pp. ISBN: ISBN 978-0-85709-625-8.

[2] Suresh G. Advani and Charles L. Tucker III. "The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites". In: (vol. 31, no. 8, pp. 751–784 1987). DOI: doi.org/10.1122/1.549945.

[3] François Alouges. "Introduction to Periodic Homogenization". In: *Interdisciplinary Information Sciences* 22 (Dec. 2016), pp. 147–186. DOI: 10.4036/iis.2016.A.01.

[4] Argha Protim Dey and Fabian Welschinger. URL: https://hym-handbook.ai.bosch.com/contrib/21_submission/extended_abstract.html.

[5] Argha Protim Dey et al. "Train hard, stop early how to teach deep material networks to reproduce creep loading of short fiber reinforced thermoplastics". In: (2022).

[6] Jimmy Lei Ba Diederik P. Kingma. "Adam: A Method for Stochastic Optimization". In: *ICLR* (2017).

[7] I. Doghri et al. "A second-moment incremental formulation for the mean-field homogenization of elasto-plastic composites". In: *International Journal of Plasticity* 27.3 (2011), pp. 352–371. ISSN: 0749-6419. DOI: https://doi.org/10.1016/j.ijplas.2010.06.004.

[8] George J. Dvorak and Yakov Benveniste. "On transformation strains and uniform fields in multiphase elastic media". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 437.1900 (1992), pp. 291–310.

[9] Felix Fritzen and Max Hodapp. "The finite element square reduced (FE2R) method with GPU acceleration: towards three-dimensional two-scale simulations". In: *International Journal for Numerical Methods in Engineering* 107.10 (2016), pp. 853–881. DOI: https://doi.org/10.1002/nme.5188.

[10] Sebastian Gajek, Matti Schneider, and Thomas Böhlke. "An FE–DMN method for the multiscale analysis of short fiber reinforced plastic components". In: *Computer Methods in Applied Mechanics and Engineering* 384 (Oct. 2021), p. 113952. DOI: 10.1016/j.cma.2021.113952.

[11] Sebastian Gajek, Matti Schneider, and Thomas Böhlke. "On the micromechanics of deep material networks". In: *Journal of the Mechanics and Physics of Solids* 142 (2020), p. 103984.

[12] S.C. Garcea, Y. Wang, and P.J. Withers. "X-ray computed tomography of polymer composites". In: *Composites Science and Technology* 156 (2018), pp. 305–319. ISSN: 0266-3538. DOI: doi.org/10.1016/j.compscitech.2017.10.023.

[13] Fraunhofer Gesselscahft. *FeelMath*. 2020. URL: https://www.itwm.fraunhofer.de/de/abteilungen/sms/produkte-und-leistungen/feelmath.html.

[14] Fraunhofer Gesselscahft. *FiberGenerator*. 2020. URL: https://www.itwm.fraunhofer.de/de/abteilungen/sms/produkte-und-leistungen.

[15] Yevgen Gorash, Holm Altenbach, and Konstantin Naumenko. "Modeling of primary and secondary creep for a wide stress range". In: *PAMM* 8.1 (2008), pp. 10207–10208. DOI: https://doi.org/10.1002/pamm.200810207.

[16] N. Halphen and Q. Nguyen. "Sur les Matériaux standards generalisés". In: *Journal de Mécanique* 14 (1975), pp. 508–520.

[17] Yuval Noah Harari. *Sapiens:A brief History of HumanKind*. 2015.

[18] Patrick Hessman et al. "Microstructural analysis of short glass fiber reinforced thermoplastics based on x-ray micro-computed tomography". In: *Composites Science and Technology* 183 (July 2019), p. 107752. DOI: 10.1016/j.compscitech.2019.107752.

[19] R. Hill. "Elastic properties of reinforced solids: Some theoretical principles". In: *Journal of the Mechanics and Physics of Solids* 11.5 (1963), pp. 357–372. ISSN: 0022-5096.

[20] J. C. SimoT. J. R. Hughes. *Computational Inelasticity*. 1998.

[21] Ruichen Jin, Wei Chen, and Agus Sudjianto. "An efficient algorithm for constructing optimal design of computer experiments". In: *Journal of Statistical Planning and Inference* 134.1 (2005), pp. 268–287.

[22] Mark Andre Keip. *Lecture notes in Computational mechanics of Materials*. Oct. 2019.

[23] Yevgen Kostenko and Konstantin Naumenko. "Power Plant Component Design Using Creep and Fatigue Damage Analysis". In: (2007).

[24] Susanne Kugler et al. "A Flow-Dependent Fiber Orientation Model". In: *Journal of Composites Science* 4 (July 2020), p. 96. DOI: 10.3390/jcs4030096.

[25] W. Kuntjoro. *An Introduction To The Finite Element Method*. McGraw-Hill, 2005. ISBN: 9780071241441. URL: https://books.google.de/books?id=uo59AAAACAAJ.

[26] Zeliang Liu, M. A. Bessa, and Wing Kam Liu. "Self-consistent clustering analysis: An efficient multi-scale scheme for inelastic heterogeneous materials". In: *Computer Methods in Applied Mechanics and Engineering* 306 (July 2016), pp. 319–341.

[27] Zeliang Liu, C. T. Wu, and M. Koishi. "A Deep Material Network for Multiscale Topology Learning and Accelerated Nonlinear Modeling of Heterogeneous Materials". In: *CoRR* abs/1807.09829 (2018).

[28] Zeliang Liu, C.T. Wu, and M. Koishi. "A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materi-

als". In: *Computer Methods in Applied Mechanics and Engineering* 345 (Mar. 2019), pp. 1138–1168. ISSN: 0045-7825.

[29] Zeliang Liu and Ct Wu. "Exploring the 3D architectures of deep material network in data-driven multiscale mechanics". In: (Jan. 2019).

[30] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Restarts". In: *CoRR* abs/1608.03983 (2016).

[31] Wolfgang Lutz et al. "Damage development in short-fiber reinforced injection molded composites". In: *Computational Materials Science - Compu mat sci* 45 (May 2009), pp. 698–708. DOI: 10.1016/j.commatsci.2009.02.001.

[32] Caren Rosales Maria Alejandra Costantino and Valeria Pettarin. "Polypropylene Blends and Composite: Processing-Morphology-Performance Relationship of Injected Pieces". In: (2019). DOI: 10.5772/intechopen.85634.

[33] Jerrold E. Marsden, Thomas J. R. Hughes, and Donald E. Carlson. *Mathematical foundations of elasticity*. 1982.

[34] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245.

[35] Andre Mielke. "Lecture notes in Machine Learning Methods in Mechanics". In: (Nov. 2021).

[36] Tom M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN: 978-0-07-042807-2. URL: https://www.worldcat.org/oclc/61321007.

[37] Stephen Montegomery-smith et al. "Exact tensor closures for the three-dimensional Jeffery's equation". In: *Journal of Fluid Mechanics* 680 (2011), pp. 321–335. DOI: 10.1017/jfm.2011.165.

[38] H. Moulinec and P. Suquet. "A numerical method for computing the overall response of nonlinear composites with complex microstructure". In: *Computer Methods in Applied Mechanics and Engineering* 157.1 (1998), pp. 69–94. ISSN: 0045-7825.

[39] Viktor Müller and Thomas Böhlke. "Prediction of effective elastic properties of fiber reinforced composites using fiber orientation tensors". In: *Composites Science and Technology* 130 (2016), pp. 36–45. ISSN: 0266-3538. DOI: doi.org/10.1016/j.compscitech.2016.04.009.

[40] Andrew Ng. *Machine Learning*. Online Draft, 2017. URL: http://www.mlyearning.org/,/bib/ng/ng2017mlyearning/Ng_MLY01_13.pdf.

[41] Optuna. URL: https://optuna.org/.

[42] Adam Paszke et al. "Automatic Differentiation in PyTorch". In: *NIPS 2017 Workshop on Autodiff* (2017).

[43] Pytorch. URL: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ExponentialLR.html.

[44] Katharina Robb, Oliver Wirjadi, and Katja Schladitz. "Fiber Orientation Estimation from 3D Image Data: Practical Algorithms, Visualization, and Interpre-

tation". In: *7th International Conference on Hybrid Intelligent Systems (HIS 2007)* (2007), pp. 320–325.

[45] B. Lauke S.-Y. Fu and Y. W. Mai. *Science and engineering of short fibre-reinforced polymer composites*. eng. 1. ed., repr. United Kingdom: Oxford: Woodhead Publishing, an imprint of Elsevier, 2019. 352 pp. ISBN: SBN 978-1-84569-649-8.

[46] Matti Schneider. "Lecture notes in Computational Homogenization on Digital Image Data". In: (Nov. 2021).

[47] Matti Schneider. "The sequential addition and migration method to generate representative volume elements for the homogenization of short fiber reinforced plastics". In: *Computational Mechanics* 59 (Feb. 2017). DOI: 10.1007/s00466-016-1350-7.

[48] Ahsan Ali Siddique. *Development and Evaluation of a Micromechanical Approach for Virtual Material Characterization of Short Fiber Reinforced Thermoplastics Under LongTerm Loading*. Jan. 2021.

[49] Simulia. *Abaqus CAE*. 2018.

[50] NumPro-Lite-University of Stuttgart. URL: https://github.com/osolmaz/numpro-lite.

[51] Synopsys Simpleware™. *ScanIP*. Accessed 24 June 2021. URL: https://www.synopsys.com/simpleware/software/scanip.html.

[52] veoptics. URL: https://www.veoptics.com/material-tensile-testing.

[53] Johannes Will and Thomas Most. "Metamodell of optimized Prognosis (MoP) -an Automatic Approach for User Friendly Parameter Optimization". In: (Nov. 2009). DOI: 10.13140/2.1.4946.9122.

[54] Will J. (Dynardo GmbH). *optiSLang - Robust design optimization(rdo) – key technology for resource-efficient product development and performance enhancement*. Accessed 2 November 2021. URL: https://www.ansys.com/de-de/products/platform/ansys-optislang.

[55] T. Zohdi. *Homogenization Methods and Multiscale Modeling*. Nov. 2004. ISBN: 9780470091357.